Sistema de navegación autónoma de micro vehículos aéreos en entornos internos de edificaciones conocidas para cumplimiento de misiones

Yamberla Morales, Danilo Jonathan

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

PhD. Aguilar Castillo, Wilbert Geovanny

17 de marzo del 2021

URKUND

Urkund Analysis Result

Analysed Document:	Tesis_Yamberla_Jonathan.pdf (D98724338)
Submitted:	3/18/2021 5:39:00 AM
Submitted By:	djyamberla@espe.edu.ec
Significance:	3 %

Sources included in the report:

TrabajoTitulaciónVinicioSalcedo.pdf (D35374643)
TESIS_CHAUCA_BRYAN_urkund.pdf (D78121896)
TESIS_SEGUIMIENTO_DRONE_REVISION_1.pdf (D29395619)
Tesis_Heredia_Didier.pdf (D40777211)
Tesis Quisaguano.pdf (D45006163)
TESIS_George_Bryan_Cobeña_Zambrano.pdf (D35797794)
http://redgeomatica.rediris.es/cartoprofesores/Fotogrametria/Apuntes%20de%20Fotogrametr%2592a%20II.pdf
https://1library.co/document/qmw0395z-comparativa-de-algoritmos-deteccion-caracteristicas-para-vision-artificial.html
https://archive.org/stream/ConceptosYMetodosEnVisionPorComputadorEditadoPorEnriqueAlegreGonzaloPajaresYArtu/Conceptos%20y%20Me%CC%81todos%20en%20Visio%CC%81n%20por%20Computador%20Editado%20por%20Enrique%20Alegre,%20Gonzalo%20Pajares%20y%20Arturo%20de%20la%20Escalera%20Junio%202016_djvu.txt
https://intranet.ceautomatica.es/sites/default/files/upload/8/files/ConceptosyMetodosenVxC.pdf

Instances where selected sources appear:

22

WILBERT GEOVANNY
AGUILAR CASTILLO

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Sistema de navegación autónoma de micro vehículos aéreos en entornos internos de edificaciones conocidas para cumplimiento de misiones**" fue realizado por el señor **Yamberla Morales, Danilo Jonathan** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 17 de marzo de 2021

Firma:

WILBERT GEOVANNY
AGUILAR CASTILLO

Ph.D. Aguilar Castillo, Wilbert Geovanny

C. C. 0703844696

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

RESPONSABILIDAD DE AUTORÍA

Yo, **Yamberla Morales, Danilo Jonathan**, con cédula de ciudadanía n° 1003509112, declaro que el contenido, ideas y criterios del trabajo de titulación: **"Sistema de navegación autónoma de micro vehículos aéreos en entornos internos de edificaciones conocidas para cumplimiento de misiones"** es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 17 de marzo de 2021

Firma

Yamberla Morales, Danilo Jonathan

C.C.: 1003509112

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **Yamberla Morales, Danilo Jonathan,** con cédula de ciudadanía n° 1003509112, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Sistema de navegación autónoma de micro vehículos aéreos en entornos internos de edificaciones conocidas para cumplimiento de misiones"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 17 de marzo de 2021

Firma

Yamberla Morales, Danilo Jonathan

C.C.: 1003509112

Dedicatoria

Dedico este trabajo a mi familia, por su constante apoyo a lo largo de toda mi vida, por nunca rendirse, ustedes han sido el mejor ejemplo de unión familiar y siempre los volvería a elegir como familia.

A mis padres, por su lucha constante para darme educación y formarme en valores, por motivarme siempre a seguir adelante y no rendirme, por estar pendientes de mi vida y ser mi apoyo incondicional en los momentos difíciles.

A mi hermano Elvis, por marcar mi camino con su experiencia y darme una idea de las decisiones que debo o no de tomar.

A mis hermanos Andrés e Inty, por la confianza en mis capacidades y darme palabras de aliento que han sido fundamentales para estar más motivado y cumplir mis objetivos.

Al amor de mi vida, Antonella. Por hacer bonito mi mundo, darme tu apoyo y amor incondicional. Tus palabras de aliento me han permitido seguir adelante con más fuerza.

Jonathan Yamberla

Agradecimiento

Agradezco primeramente a Dios por la vida, la salud y la familia que me ha dado, gracias por su infinita bondad y amistad que nunca falla, gracias por llenarme cada día de bendiciones. Y también a la virgencita María, por ser mi madre espiritual e interceder a Dios por mí.

Agradezco a mis padres, Yolanda Morales y Enrique Yamberla, por ser los mejores padres, por hacer que nunca falte nada en casa, gracias a su humildad y valores que han logrado transmitir en mí soy lo que soy.

A mis hermanos, Elvis y Andrés gracias por su compañía, por compartir tantos momentos juntos, cada uno de ustedes es único, me han enseñado tanto y he aprendido mucho.

Agradezco a mi hermano Inty por las desveladas y las veces que me has ayudado en mis proyectos, aunque no lo creas has sido de una enorme ayuda de tu parte sin pedir nada a cambio, de no ser por ti no llegaría tan lejos.

A mis amigos, compañeros y profesores que me acompañaron en esta etapa de mi vida, gracias por compartir su conocimiento y por la paciencia en la enseñanza, y un agradecimiento especial a mi tutor de tesis por la confianza y permitir que este trabajo se realice exitosamente.

Jonathan Yamberla

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El proyecto de investigación consiste en el desarrollo de un sistema de navegación autónoma basada en visión por computador al interior de una vivienda mediante el uso de un micro UAV Parrot Bebop2. El sistema consta de tres partes, la planificación de camino entre dos puntos, los cuales son establecidos mediante la selección de un punto de partida y destino en un plano escalado de pixeles a metros que representa el interior de la vivienda, utilizando algoritmos de búsqueda como RRT (Rapidly-exploring Random Trees) y A* (A-star). La segunda es realizar el modelamiento servo visual de la planta y diseño de los controladores PID para los ejes Pitch (desplazamiento lineal en el eje x) y Yaw (desplazamiento angular sobre el eje z), mediante la extracción de puntos característicos con el algoritmo ORB (Oriented FAST and Rotated BRIEF), correspondencia de puntos y transformaciones geometricas que se traducen a estimación de movimiento mediante fotogramas consecutivos, las cuales son procesadas por la estación terrestre que es una PC. La tercera parte consiste en la implementación del controlador que permite el seguimiento de la ruta generada. Además, para la identificación de zonas accesibles se utiliza marcadores ArUco que sirven a su vez de sistemas de referencias y plataformas de aterrizaje para el UAV. La evaluación del sistema desarrollado se realiza al interior de una vivienda en diferentes condiciones de iluminación y con diferentes rutas.

PALABRAS CLAVE:

- **PLANIFICACIÓN DE RUTA**
- **SEGUIMIENTO DE RUTA**
- **VISION POR COMPUTADOR**
- **NAVEGACION AUTONOMA**

Abstract

The research project consists of the development of an autonomous navigation system based on computer vision inside a home using a micro UAV Parrot Bebop2. The system consists of three parts, the planning of the path between two points, which are established by selecting a starting point and a destination in a plane scaled from pixels to meters that represent the interior of the house, using search algorithms such as RRT. (Quick scan random trees) and A * (A-star). The second is to perform the visual servo modeling of the plant and design of the PID controllers for the Pitch (linear displacement in the x-axis) and Yaw (angular displacement on the z-axis), by means of the extraction of keypoints with the ORB algorithm. (Oriented FAST and Rotated BRIEF), matching of points and geometric transformations that are translated into motion estimation through consecutive frames, which are processed by a PC. The third part consists of the implementation of the controller that allows the tracking of the generated route. In addition, ArUco markers are used to identify accessible areas, which in turn serve as reference systems and landing platforms for the UAV. The evaluation of the developed system is carried out inside a house in different lighting conditions and with different routes.

KEYWORDS:

- **PATH PLANNING**
- **PATH TRACKING**
- **COMPUTER VISION**
- **AUTONOMOUS NAVIGATION**

Capítulo I

Introducción

Antecedentes

Durante los últimos años, el campo de los micro vehículos aéreos ha vuelto significativo en la comunidad de investigación en robótica y en otros campos. El interés proviene de las notables ventajas de estas plataformas voladoras sobre los vehículos aéreos no tripulados (UAV) estándar en términos de seguridad y eficiencia en tamaños pequeños (Huang, et al., 2009). Los micro UAV son una reciente innovación para la recopilación de información visual y otro tipo de información recogida por diferentes sensores, sobrevolada de manera remota (Meythaler, et al., 2016). Al ser dispositivos controlados por un mando remoto, permiten obtener información de lugares inaccesibles (Sani, Tierra, & Robayo, 2015) y es usado en diferentes aplicaciones como la detección de objetivos, operaciones de búsqueda y rescate, monitoreo y vigilancia, seguridad, aplicaciones militares, entre otras (Schnipke, et al., 2015).

Los UAV además de operar a control remoto como se mencionó anteriormente, también pueden operar con instrucciones pre-programadas (de manera autónoma). Diversos sistemas de navegación autónoma han sido desarrollados en los países exportadores de tecnología, los cuales basan su funcionamiento en sistemas de posicionamiento global junto con sistemas de navegación inercial (GPS/INS), algoritmos de procesamiento de imágenes digitales, entre otros (Meythaler, et al., 2016). Algunas de las ventajas de la navegación autónoma es que permite incrementar el rango de vuelo, contar con planes de contingencia que doten al vehículo la capacidad de implementar soluciones automatizadas en caso de fallas, disponer de mayor confiabilidad y versatilidad en planificación y cumplimiento de misiones (Huang, et al., 2009).

El desafío de navegación en entornos internos simples, menos costosos y eficientes ha sido emprendido por muchos grupos de investigación de UAV (Huang, et al., 2009), (Ehsan & McDonald-Maier, 2009), (Zhang, et al., 2010), y se ha realizado muchas aplicaciones de las cuales se mencionan a continuación: “Navegación autónoma de drones y automatización de rutas aplicadas a la limpieza de edificios” (Plaza, 2017), en este trabajo el UAV se desplaza por la superficie a limpiar siguiendo rutas preestablecidas, una vez finalizada la tarea este retorna al punto de partida mediante las coordenadas registradas previamente. Otro trabajo es el descrito en “Development of a Micro Quad-Rotor UAV for Monitoring an Indoor Environment” (Min, et al., 2009), el propósito de este trabajo es desarrollar un micro UAV de cuatro rotores como plataforma para monitorear un ambiente interior, usa estrategias de control basadas en el control PD y es equipado con micro-controladores, varios sensores y una cámara inalámbrica. De la misma manera “Improved Potential Field Method for Unknown Obstacle Avoidance Using UAV in Indoor Environment” (Mac, et al., 2016) es otro trabajo que se basa en la navegación autónoma en entornos internos, la cual propone una solución para la planificación de ruta que evita colisiones usando solo sensores servo visuales e inerciales integrados.

El Centro de Investigación y Desarrollo de la Fuerza Aérea Ecuatoriana (CIDFAE) y el Centro de Investigación de Aplicaciones Militares (CICTE) son centros de desarrollo tecnológico de las Fuerzas Armadas con la finalidad de mejorar la capacidad operativa para la defensa. Los dos centros han desarrollado varios proyectos de entre los cuales se puede mencionar VisualNavDrone, un proyecto de investigación por parte del CICTE desarrollado entre 2016 y 2019, dirigido por el Dr. Wilbert G. Aguilar, en el que se tuvieron varios avances como el desarrollo del algoritmo de evasión de obstáculos y recuperación de trayectoria basado en detección de objetos conocidos tanto fijos como móviles. Otro

trabajo fue el algoritmo que se desarrolló para la detección de personas en tiempo real mediante la combinación de algoritmos HAAR, LBP y regiones salientes para desestimar falsos positivos. De la misma manera el desarrollo de un algoritmo de detección de eventos anómalos basados en el análisis estadístico del movimiento estimado de personas a partir de un conjunto de imágenes.

El presente proyecto pertenece a SmartDrone1, un proyecto aprobado por el Departamento de Eléctrica, Electrónica y Telecomunicaciones de la Universidad de las Fuerzas Armadas que tiene como objetivo fortalecer los departamentos de defensa e investigación como son CIDFAE y CICTE mediante la creación de aplicaciones y sistemas inteligentes de monitoreo y aterrizaje para UAVs tácticos y multirrotores.

Tomando en cuenta los antecedentes mencionados y con el objetivo de superar las dificultades que conlleva la navegación autónoma, se propone en este proyecto de investigación desarrollar un sistema de navegación autónomo basada en visión por computador para micro UAV que sobrevuelen entornos internos de un edificio conocido con el propósito de monitoreo y vigilancia.

Justificación e importancia

Hoy en día el crecimiento delictivo genera la necesidad de contar con servicios de vigilancia que se encuentren muy bien capacitados y/o certificados para atender riesgos mayores y menores (Rojo & Ramos, 2019). Los centros de monitoreo son una necesidad para efectuar las llamadas de emergencia con mayor rapidez, ya sea en seguridad pública o privada. Dada la inseguridad que se vive en la actualidad cada vez son más, tanto las empresas que solicitan esta herramienta como las que ofrecen los servicio, constantemente están en la búsqueda de nuevas y mejores herramientas para su integración y que ofrezcan una solución más completa (Moscoso, 2016).

En el ámbito civil en Ecuador existen muchas empresas que prestan servicios de seguridad como Seprimun, Vigar, G4S, Segdefensa, entre otros los cuales incluyen instalaciones de cámaras de vigilancia en puntos estratégicos, personal de vigilancia las 24 horas del día, alarmas inteligentes y otras herramientas de apoyo para la seguridad. Muchas de estas empresas prestan sus servicios a un costo considerable y esto varía de acuerdo al tamaño de las instalaciones y/o edificaciones, en algunos casos es necesario no solo de una persona para vigilar dicha edificación sino requiere de dos o más dependiendo el caso.

En el ámbito militar se ha desarrollado muchas aplicaciones enfocándose principalmente en entornos externos con UAV, como detección de cultivos ilegales, exploración y reconocimiento de zonas inaccesibles (Valavanis & Vachtsevanos, 2015), misiones de búsqueda y rescate con UAVs (Tomic, et al., 2012), entre otros. A comparación de aplicaciones con UAV en entornos internos no ha sido un tema tan abordado por el sector militar, a pesar de los riesgos que conlleva realizar misiones hacia una edificación por más conocida que sea. Se considera que el uso de UAVs puede ser una alternativa adecuada para la reducción de bajas del personal militar al momento de realizar misiones, haciendo más eficiente y segura el monitoreo interno de edificaciones, u otro tipo de construcciones, en zonas inaccesibles. El objetivo es ofrecer a las entidades públicas y privadas del estado una herramienta ágil, segura y fiable de monitoreo y vigilancia basada en tecnología dron.

Hoy en día, las Fuerzas Armadas necesitan de vehículos aéreos no tripulados para múltiples operaciones militares en el ámbito interno y externo. El objetivo de estas aeronaves es proporcionar al personal militar de herramientas tecnológicas para reducir su exposición a peligros que pueden presentarse. Además, al contar con sistemas que

proporcionen información desde una perspectiva aérea del sector durante una misión, las operaciones aumentan su probabilidad de éxito.

Los centros de investigación CIDFAE y CICTE cuentan con líneas de investigación comunes y desarrollos compatibles en diferentes sistemas que integran una aeronave no tripulada. Para integrar la tecnología y algoritmia desarrollada por los dos centros de investigación de las Fuerzas Armadas se crea el proyecto SmartDrone1, bajo la dirección del Dr. Wilbert G. Aguilar, el mismo que tiene como propósito mejorar la capacidad operativa de reconocimiento, vigilancia, mando y control de las operaciones militares de UAVs desarrollados CIDFAE. Cabe mencionar que el presente proyecto es parte del proyecto SmartDrone1.

El proyecto surge ante la necesidad de monitorear y vigilar edificaciones conocidas, aprovechando los avances tecnológicos como son los UAV y que aporten más valor a empresas que prestan servicios de monitoreo y vigilancia, a su vez entidades de las Fuerzas Armadas como el Centro de Investigaciones CICTE para mantener la seguridad del personal militar al momento del cumplimiento de misiones en el interior de un edificio conocido que por alguna circunstancia el mismo se encuentre en un estado no accesible o críticamente peligroso para el personal militar.

El desarrollo del proyecto apoyará al fortalecimiento del Centro de Investigaciones CICTE y puede ser utilizado como recurso para otras investigaciones y desarrollos tecnológicos. Además de contribuir con la Universidad de las Fuerzas Armadas ESPE en su misión de formar profesionales con visión de futuro, capaces de proponer e implementar soluciones rápidas y eficaces a problemas de la sociedad.

Alcance del proyecto

El proyecto de investigación tiene como propósito desarrollar e implementar un sistema basado en visión artificial para la navegación autónoma de un micro UAV Parrot Bebop 2 en entornos internos de una edificación conocida para cumplimiento de misiones. El sistema estará montado sobre ROS (Robotic Operative System) y para la adquisición de las señales se utilizará visión por computador.

En la primera etapa del proyecto se realizará el estudio del arte sobre las diferentes técnicas utilizadas en visión por computadora, al mismo tiempo instalar el software y librerías necesarias para la comunicación WIFI entre la estación terrestre y el UAV que permitan funciones básicas de control como despegue y aterrizaje, activación/desactivación de la cámara integrada en el UAV.

La segunda etapa consiste en desarrollar un sistema de planificación de ruta para una edificación conocida, para la cual se definirá el espacio de trabajo mediante un plano de la edificación (solo un piso), seguido del desarrollo del algoritmo en base a exploración rápida de árboles aleatorios o RRT (Rapidly Exploring Random Tree). RRT es un algoritmo muy popular porque es bastante fácil de implementar y es fácilmente extensible a sistemas con varios grados de libertad, debido a su versatilidad y eficacia tiene un constante crecimiento en diferentes aplicaciones y algoritmos alternativos mejorados (Montes, 2017). RRT calcula un camino continuo conectando una configuración inicial con una configuración final, utilizando técnicas probabilísticas y teniendo en cuenta los obstáculos y las restricciones.

La tercera etapa del proyecto contempla el desarrollo del sistema de control servo visual. En este tipo de control las consignas de error provenientes del análisis visual de las imágenes se introducen directamente como consignas de control a los actuadores del robot. Para la percepción del entorno se utilizará la cámara monocular integrada en el

Parrot Bebop 2, la misma que proporciona imágenes en el espectro visible de las cuales es necesario determinar los puntos característicos (features points) (Pistarelli, 2013). Existen denominados esquemas clásicos, que permiten llevar a cabo un eficiente procesamiento en la detección y descripción de puntos característicos, tales como los algoritmos SIFT (Zhou, Wang, & Fu, 2016), SURF (Bay, Tuytelaars, & Van Gool, 2006), FAST (Rosten & Drummond, 2006), entre otros, de los cuales se elegirá el que mejor se adapte al proyecto desarrollado. Por otra parte, para el control del UAV el tipo de controlador empleado puede llegar a ser muy diverso, se puede optar desde un control PID (Proporcional, Integral, Derivativo), usado mucho en aviación y basado en la realimentación de sus parámetros, lógica difusa basada en lo relativo a lo observado, hasta un control lineal con movimientos constantes. El control PID es la mejor opción para controlar sistemas que se mueven en 3 dimensiones (Garijo, López, & Pérez, 2009).

Figura 1

Esquema general de funcionamiento

Nota. El gráfico muestra la estación terrestre y la edificación en donde el micro-UAV Parrot Bebop 2 realizará la planificación y seguimiento de trayectoria.

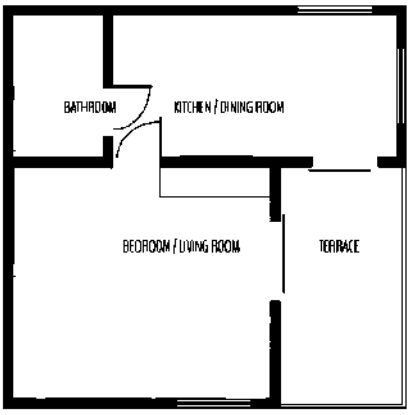
La Figura 1 muestra el esquema general del funcionamiento del sistema, se tiene el UAV posicionado en un entorno interno (dentro de la edificación), el mismo que se

conecta mediante WI-FI con una estación terrestre la cual realiza el procesamiento y envía las señales de control respectivas al UAV que permitan la navegación autónoma.

En la navegación en general es muy útil el uso de planos de los espacios donde se supone navegará y actuará los robots móviles, por lo que se definirá un plano de una edificación de una sola planta como se puede observar en la Figura 2.

Figura 2

Plano de edificación (primera planta)

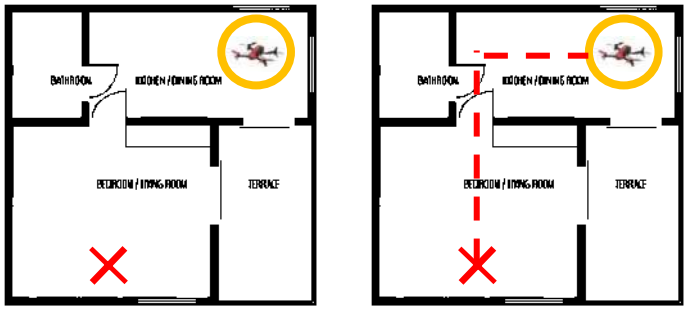


Nota. El gráfico muestra la estructura interna de la edificación (paredes, ventanas, puertas) correspondiente solo a la primera planta.

El UAV será capaz de navegar y explorar autónomamente el entorno mediante la asignación de un punto de exploración en el mapa, generación y asignación de una trayectoria mediante algoritmos de visión por computadora y de exploración ya mencionados, de manera que permitan el desplazamiento seguro del UAV al punto de navegación establecido. En la Figura 3 y Figura 4 se observa de manera gráfica lo descrito anteriormente.

Figura 3

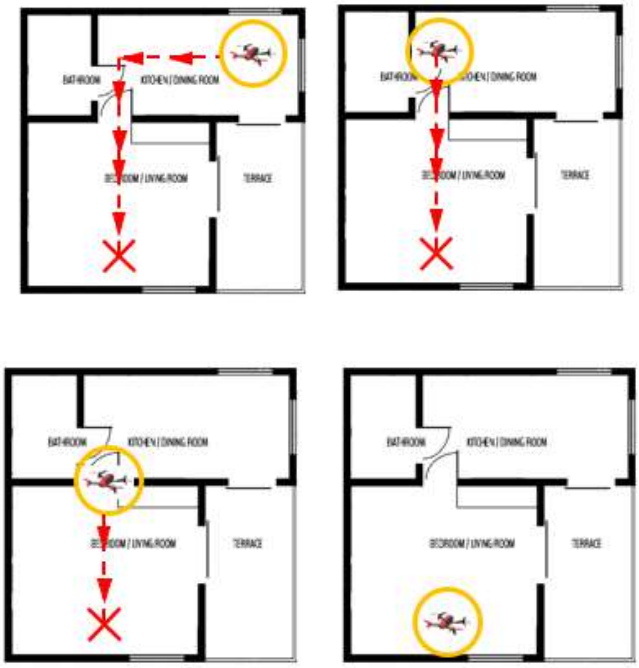
Asignación de punto destino en el mapa y planificación de ruta



Nota. En el gráfico se representa el punto de llegada (cruz roja) y la generación de una trayectoria desde la posición inicial del micro-UAV hasta el punto destino.

Figura 4

Seguimiento a trayectoria hasta punto destino



Nota. El gráfico representa el movimiento del micro-UAV según la trayectoria generada desde su posición inicial hasta llegar a su objetivo.

La etapa final del proyecto consiste en realizar pruebas de funcionamiento al colocar el UAV en el interior de la edificación, señalar en el mapa de la interfaz gráfica un punto de destino y que el UAV sobrevuele el interior del edificio de manera autónoma hasta llegar al punto de destino establecido.

Objetivos

Objetivo general

Desarrollar un sistema de control basado en visión artificial para la navegación autónoma de un micro vehículo aéreo no tripulado en entornos internos de una edificación conocida.

Objetivos específicos

- Realizar un estudio del estado del arte de distintas técnicas y tecnologías que se utilizan para la navegación autónoma de vehículos aéreos no tripulados en entornos internos.
- Establecer la comunicación entre la estación terrestre y el Parrot Bebop 2 mediante WIFI para controlar la navegación desde la estación terrestre.
- Diseñar un sistema de planificación de ruta en base al algoritmo RRT que permita determinar la trayectoria del punto inicial al punto destino.
- Diseñar y modelar un sistema de control servo visual basado en detección de puntos de interés, que permita compensar los movimientos requeridos para la navegación teniendo en cuenta las perturbaciones presentes en entornos internos.
- Realizar pruebas experimentales del sistema implementado utilizando un micro vehículo aéreo no tripulado Parrot Bebop 2 con el fin de validar los resultados y tomar los correctivos en caso de ser necesarios.

Capítulo II

Estado del arte

Vehículos aéreos no tripulados

Los UAV (del inglés Unmanned Aerial Vehicle) conocidos como Vehículos aéreos No Tripulados son uno de los de mayor evolución en los últimos años (Mojica, Cuellar, & Medina, 2015), también son conocidos como drones, se caracteriza por no contar con tripulación humana y puede ser operado por control remoto o ser totalmente autónomo siguiendo una misión pre programada. Originalmente el UAV fue desarrollado con fines militares para tareas de supervisión y de espionaje, portadores de arma de combate (Martínez, et al., 2012), que con el paso de los años ha terminado comercializado por diferentes empresas como medio de entretenimiento e incluso sirven como plataforma de investigación y desarrollo.

Para elegir un UAV es necesario tener en cuenta ciertas características como: autonomía, dimensiones, carga útil transportable, capacidad de despliegue, alcance, tipo de motor, potencia, entre otros. Es decir, las especificaciones de rendimiento en general del UAV. Debido al sin número de aplicaciones que se pueden realizar con un UAV es difícil desarrollar un sistema de clasificación que abarque todos los vehículos aéreos no tripulados, por lo que se puede clasificar dependiendo del aspecto de su misión como: de blanco, reconocimiento, combate, logística investigación y desarrollo, comerciales y civiles (Mojica, Cuellar, & Medina, 2015).

Otro de los métodos de clasificación más empleados en la literatura es la basada en métodos de generación de sustentación, este criterio agrupa los UAVs en aeronaves más pesadas que el aire (aerodinos) y aeronaves cuya suspensión se debe al uso de un

gas más ligero que el aire (aerostatos) (Santana, 2017). En la Figura 5 se representa la clasificación antes mencionada.

Figura 5

Clasificación de los UAVs en función del método de sustentación utilizado

Nota. El gráfico representa una de las clasificaciones más comunes de los UAVs. Tomado de *"Propuesta de sistema multi-UAV para aplicaciones de cobertura de área"*, (Santana, 2017).

Los UAVs más utilizados en el ámbito civil como militar son los aerodinos, por ello solo se referirá a los tres tipos de aerodinos más comunes (izquierda) que se muestra en la Figura 5, los de ala fija, de ala rotatoria e híbridos.

- **UAV de ala fija:** Poseen una estructura simple con una elevada aerodinámica, las mismas que hacen de estas aeronaves tener un alto rendimiento energético y tiempos de vuelos relativamente elevados. Se caracterizan por tener las alas unidas a la aeronave. La complejidad en la ejecución de vuelo se debe a que para realizar el despegue o aterrizaje es necesario de infraestructura externa como se observa en la Figura 6. Este tipo de UAVs es recomendable cuando se requieren vuelos a velocidades y alturas superiores (Addati & Pérez, 2014), es por eso que la maniobrabilidad es mucho menos a las de ala rotatoria, eso implica una restricción a ser utilizados en espacios muy complejos y poco extensos.

Figura 6

UAV de ala fija

Nota. El grafico representa un UAV de ala fija en vuelo, necesita de elementos externos para realizar el despegue y el aterrizaje. Tomado de *"Introducción a los UAV's, Drones o VANTs de uso civil",* (Addati & Pérez, 2014).

- **UAV de alas rotatorias:** Aquellos en los cuales las alas (palas) giran alrededor de un eje, de esta manera se consigue la sustentación en el aire (Bayindir & Sahin, 2007). Esta categoría de UAVs puede subdividirse de acuerdo al número de rotores y su configuración, así se tiene: aeronaves con un rotor principal un rotor en la cola, aeronaves con único rotor, aeronaves con dos rotores en configuración coaxial, aeronaves en configuración tándem y multirrotores (Sahin, et al., 2008). Los últimos son aeronaves que poseen tres o más rotores. De acuerdo a su configuración se denominan tricopteros (aeronaves con tres rotores), quadcópteros (cuatro rotores), hasta configuraciones de 8 rotores llamados octocópteros. A diferencia de los UAV de ala fija, los de ala rotatoria no requieren infraestructura externa para despegar o aterrizar de forma vertical. Presentan un alto nivel de maniobrabilidad con un elevado nivel de precisión, por lo que este tipo de UAVs pueden realizar misiones en interiores como en exteriores. De la misma manera, son capaces de volar a alturas muy bajas respecto al suelo captando imágenes con un nivel de resolución elevado (Santana, 2017).

Figura 7

UAV de alas rotatorias

Nota. UAV multirrotor, configuración de 8 rotores. Tomado de *"DRONES: La tecnología, ventajas y sus posibles aplicaciones",* (Pinto, 2016)

- **UAVs híbridos:** De estructura mecánica y control complejo, esto se debe a que reúnen características del UAV de ala rotatoria como del UAV de ala fija, es decir, puede despegar y aterrizar de forma vertical y realizar vuelos a alta velocidad respectivamente. En la Figura 8 se muestra un ejemplo de un UAV híbrido comercial.

Figura 8

UAV híbrido

Nota. Adaptada de UAV híbrido [fotografía], por *"SONGBIRD: EL DRON HÍBRIDO DE GERMANDRONES"*, 2019, HispaDrones (www.hispadrones.com/ /songbird-dron-hibrido-germandrones/)

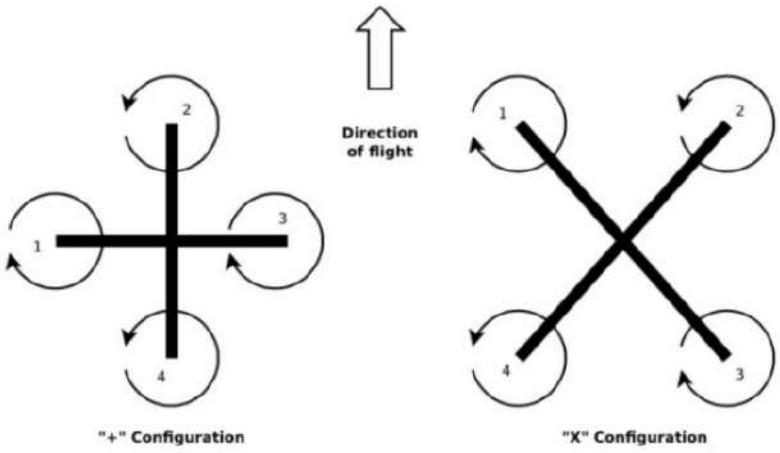
De acuerdo a la clasificación detallada anteriormente y según características como el peso del UAV, altitud normal de operación, alcance y autonomía de vuelo (Gupta, Chonge, & Jawandhiya), este trabajo se enfocará en un micro UAV cuadricóptero (configuración de 4 rotores).

Dinámica de movimiento de un UAV configuración 4 rotores (cuadricóptero)

El cuadricóptero se caracteriza por disponer una configuración de cuatro rotores, cada rotor está compuesto por un motor y una hélice, estos se ubican en los extremos de los brazos que los unen al cuerpo del dron. La configuración de los brazos puede tener un ángulo de inclinación como se muestra en la Figura 9. Los cuatro rotores fijos hacen que el cuadricóptero tenga cuatro fuerzas de entrada, que son básicamente el empuje proporcionado por cada una de las hélices (Thu & Gavrilov, 2017).

Figura 9

Dos tipos principales de configuración de cuadricóptero



Nota. Configuración “+” y configuración “x” de un cuadricóptero. Tomado de “*Designing and Modeling of Quadcopter Control System Using L1 Adaptive Control*”, (Thu & Gavrilov, 2017)

Las dos configuraciones posibles para la mayoría de los diseños de cuadricópteros son positiva “+” y en cruz “x”. Un cuadricóptero en configuración “x” se considera más estable respecto a la configuración “+”, esta última es una configuración mucho más acrobática (Thu & Gavrilov, 2017). Cabe mencionar que para el desarrollo del proyecto de investigación se hace uso de la configuración en cruz.

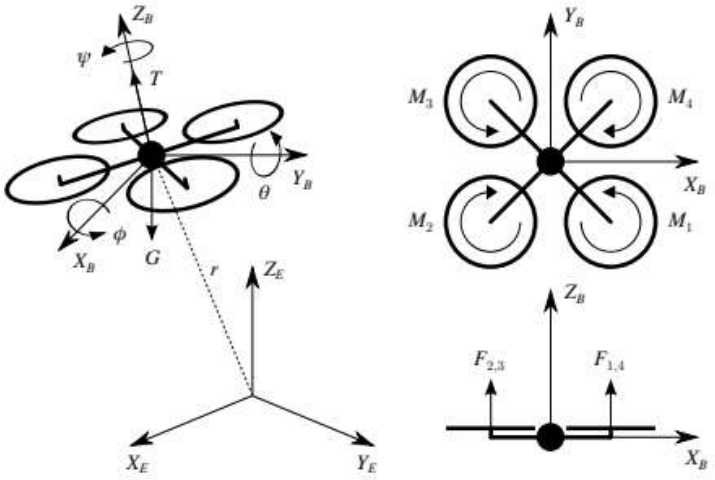
Para sustentarse en el aire, así como para desplazarse un cuadricóptero utiliza el empuje generado por los cuatro rotores. Como se mencionó los rotores están ubicados de forma simétrica respecto al centro de masa del dron en general, cada rotor debe generar un empuje de igual intensidad al momento en que el dron se mantiene en un punto fijo, de manera que el empuje total compense la acción de la gravedad, a esto se lo conoce como hovering (Cancilieri, et al., 2019). Cabe recalcar que cada rotor genera paralelamente un empuje y un torque, de modo que debido a la configuración de los rotores su sentido de giro es alternado (Figura 9), por lo que los torques de igual valor y signo opuesto se cancelan, de esta manera se logra que el dron se mantenga en su posición de hovering sin girar sobre su propio eje vertical (Kharsansky, 2013).

Un cuerpo rígido en tres dimensiones, aislado del entorno y sin restricciones, tiene asociado seis grados de libertad, los cuales corresponden a los tres movimientos de traslación (ejes X, Y, y Z) y tres movimientos de rotación en torno a ellos (angulos Roll, Pitch y Yaw) (Lugo, 2018). El fundamento matemático que sigue se sustenta en (García, et al., 2012) (Hughes, 2012) (Zipfel, 2009).

El sistema de referencia del cuerpo rígido y el global se puede observar en la Figura 10, de igual manera se puede observar la representación del cuadricóptero con las fuerzas y momentos que involucra.

Figura 10

Representación del cuadricóptero en el espacio tridimensional



Nota. El grafico muestra los planos superior y frontal, así como los sistemas de referencia relativo y global. Tomado de "*Sistema para ejecución de trayectorias de drones aéreos*", (Lugo, 2018).

El estado de un cuadricóptero se define por la posición (1) y la orientación (2) con respecto al sistema de referencia global.

$$r = [x \quad y \quad z]^T \tag{1}$$

$$\eta = [\phi \quad \theta \quad \psi]^T \tag{2}$$

Las velocidades lineales y angulares bajo el marco de referencia del cuerpo rígido esta determinadas por (3) y (4) respectivamente.

$$v = [v_x \quad v_y \quad v_z]^T \tag{3}$$

$$w = [w_\phi \quad w_\theta \quad w_\psi]^T \tag{4}$$

De la misma manera cada uno de los rotores i crean una fuerza F_i en dirección al eje del rotor y un momento M_i alrededor del mismo eje. Las ecuaciones (5) y (6)

relacionan dichas variables con la velocidad angular del rotor Ω_i, en donde es k_l el coeficiente de sustentación y k_d es el coeficiente de arrastre de las hélices.

$$F_i = k_l {\Omega_i}^2 \tag{5}$$

$$M_i = k_d {\Omega_i}^2 \tag{6}$$

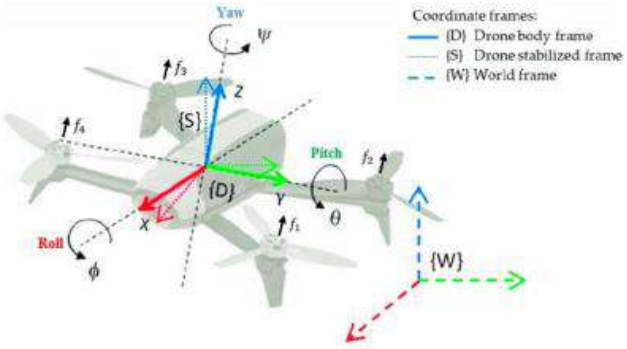
El empuje total producido en el eje Z del cuerpo por las fuerzas combinadas de los rotores viene dado por la suma de cada una de las fuerzas como se muestra en (7). Las fuerzas generan torques en dirección de los ángulos Roll y Pitch del cuerpo ($\tau_\phi \; y \; \tau_\theta$), mientras que el torque en dirección del ángulo Yaw (τ_ψ) solo es afectado por los momentos de los rotores, los torques mencionados en conjunto describen el torque total aplicado sobre el cuadricóptero τ.

$$T_z = F_1 + F_2 + F_3 + F_4 \tag{7}$$

$$\tau = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T \tag{8}$$

Figura 11

Sistemas de referencia asociados al cuadricóptero



Nota. El grafico muestra los sistemas de referencia, así como los movimientos rotacionales Roll, Pitch y Yaw. Tomado de (López, et al., 2017).

Cada uno de los movimientos rotacionales posee un nombre especial (ver Figura 11), los cuales se describen a continuación:

- **Roll:** Esta rotación se encarga del alabeo del cuadricóptero, siendo el eje de rotación el que va del frente hacia la parte trasera del cuadricóptero, permitiendo que este se pueda mover de derecha a izquierda (Chouza, et al., 2020).
- **Pitch:** Esta rotación le permite al cuadricóptero realizar movimientos hacia adelante y hacia atrás, siendo el eje de rotación el que va del lado derecho del cuadricóptero al izquierdo (Chouza, et al., 2020).
- **Yaw:** Esta rotación le permite a un cuadricóptero realizar rotaciones en sentido de las manecillas del reloj y opuesto, por lo cual esta rotación se realiza a través del eje y en el espacio (Chouza, et al., 2020).

Sistemas de comunicaciones inalámbricas para UAVs

Los sistemas y redes habilitadas para vehículos aéreos no tripulados se consideran para diversas aplicaciones que van desde operaciones militares y de seguridad hasta entretenimiento y telecomunicaciones.

Un UAV equipado con una cámara y otros sensores necesarios pueden proporcionar soluciones rentables para vigilancia, inspección y entrega, el equipo de usuario aéreo debe coexistir con los usuarios terrestres y explotar la infraestructura existente como las redes celulares para transferir la información recopilada al operador en tierra con un nivel de confiabilidad, rendimiento y demora (Vinogradov, et al., 2019).

Sistema de Control para UAV

Hoy en día el interés por los vehículos aéreos no tripulados no se centra solamente en desarrollar aplicaciones enfocadas en visión a través de las cámaras integradas, sino que el control de estos sigue siendo un gran desafío para muchos (Garijo, López, & Pérez, 2009). Es por ello que se han desarrollado y siguen desarrollándose muchos proyectos relacionados con este tema, de entre tantos se puede mencionar los siguientes: “Control Basado en Pasividad para un quadrotor UAV” (Guerrero, Lozano, & García, 2015),

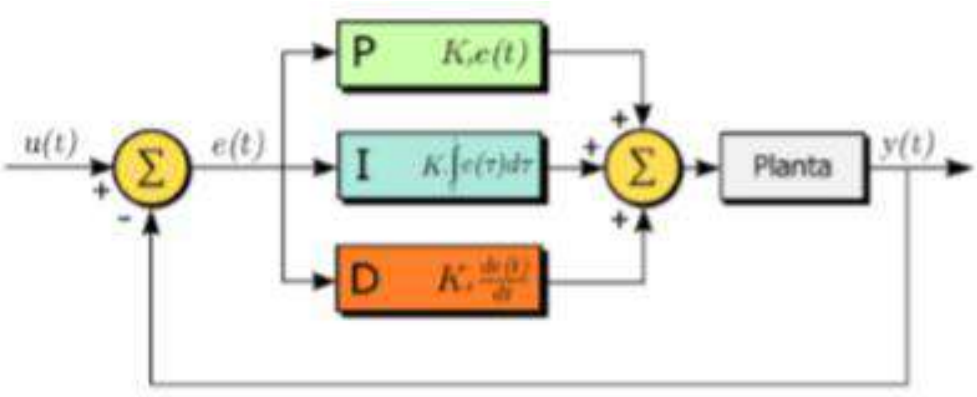
"Robustus control strategies for a QuadRotor helicopter" (Guilherme, 2011), "Ob-board visual control algorithms for unmanned aerial vehicles" (Mondragón, 2011), "Desarrollo de un sistema difuso aplicado al despegue de micro-UAV" (Santamaria & Kemper, 2014), "Adaptive search control applied to Search and Rescue operations using Unmanned Aerial Vehicles (UAVs)" (Chaves & Cugnasca, 2014). De acuerdo a esto podemos decir, que el tipo de control empleado puede llegar a ser muy distinto, puede ir desde el control más clásico y usado PID, lógica difusa, control adaptativo, entre otros. Según la aplicación a desarrollar es necesario determinar el sistema de control más apropiado.

Control PID

Un controlador PID (del inglés Proportional, Integral, Derivative controller) permite controlar un sistema en lazo cerrado para que la variable controlada (set point) alcance el estado de salida deseado. El controlador PID está compuesto de tres elementos que proporcionan una acción Proporcional, Integral y Derivativa. Estas tres acciones son las que dan nombre al controlador PID (Ogata, 2007), como se observa en la Figura 12.

Figura 12

Controlador PID



Nota. El gráfico muestra el esquema de la estructura de un controlador PID. Tomado de "*Control de un vehículo aéreo no tripulado*", (Garijo, López, & Pérez, 2009).

El valor Proporcional determina la reacción del error actual. El Integral produce un cambio proporcional a la integral del error, mientras que el error de seguimiento se reduce a cero. El Derivativo determina la reacción del tiempo en el que el error se produce (Garijo, López, & Pérez, 2009).

Control servo visual

La función principal del servo visual control es controlar el quadrotor utilizando la información obtenida en el espacio de la imagen. El control servo visual se puede dividir en dos categorías: el control servo visual basado en pose y el control servo visual basado en imágenes. El primero implica la reconstrucción o estimación de la pose, y el segundo puede trabajar directamente con los datos de la imagen a través de una cámara, omitiendo el proceso de reconstrucción. En comparación con el primero, el segundo es intrínsecamente robusto a la calibración de la cámara y los errores de modelado de objetivos (Yang, et al., 2017).

Para este proyecto se desarrollará un control servo visual basado en imágenes, las mismas que serán captadas por la cámara del micro UAV, para el modelamiento servo visual se toma como referencia los trabajos realizados por parte de (Aguilar & Angulo, 2014) y (Aguilar, et al., 2017).

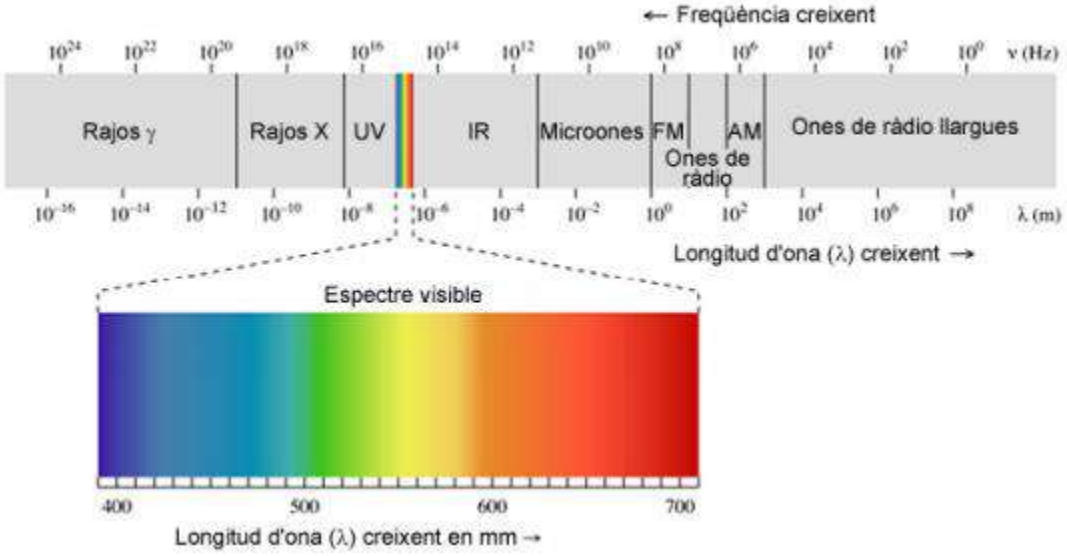
Imagen digital

Antes de definir la imagen digital es necesario diferenciar ciertos conceptos como que es la luz visible. El ojo humano puede ver una parte del espectro de toda la luz que ilumina el universo (Figura 13), el rango de luz que podemos observar se denomina “luz visible”, por consecuencia existe frecuencias de luz que no se pueden ver, como por ejemplo los infrarrojos y los ultravioleta.

La mayoría de las cámaras digitales son sensibles a la luz visible, pero también son sensibles en cierta medida a la luz infrarroja y/o ultravioleta, estos espectros de luz no son útiles para construir una imagen digital, dicho de otra manera no se tienen una representación posible en un color. Por tal razón, la mayoría de las cámaras digitales llevan un filtro anti infrarrojos, para cortar todas las frecuencias por debajo del espectro visible que dicho en otras palabras resultaría en ruido innecesario, y solo permite pasar el rango de luz visible que se desea plasmar con la cámara (García E. , 2017).

Figura 13

Espectrograma de la luz visible y no visible



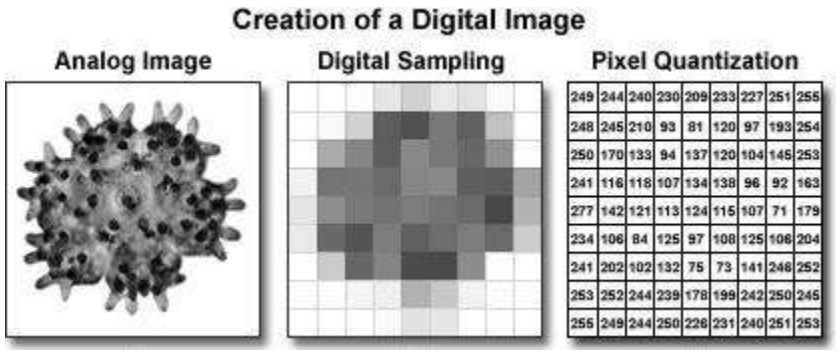
Nota. El grafico muestra todo el espectro de luz, en donde se puede observar que el espectro visible. Tomado de "*Radiación electromagnética*", Wikipedia (http://en.Wikipedia.org/wiki/Electromagnetic_radiation).

La imagen digital es el resultado de convertir datos analógicos en digitales, la luz incide en el sensor digital de la cámara y genera señales eléctricas que un procesador (conversor analógico-digital) convertirá en código digital creando un archivo de imagen.

En el mundo digital las imágenes se representan como una matriz bidimensional de pixeles como muestra la Figura 14, en el que cada pixel puede adquirir valores de codificación de color con tres parámetros RGB (Red, Green, Blue). Cada pixel es una combinación de unos valores de color y brillo en una posición determinada que se registra numéricamente (Costa & Fernandez, 2005).

Figura 14

Creación de una imagen digital



Nota. El grafico muestra el proceso para la creación de una imagen digital, en donde se observa el valor de los pixeles que representan la imagen original. Tomado de "*Concepts in digital imaging technology*", (Spring & Russ, 2018).

La definición de la imagen también es importante, esta vendrá dada por la resolución espacial. La resolución se define como el número de pixeles que componen la imagen, a mayor resolución, mayor detalle de imagen y, por lo tanto, mayor calidad. Esto trae consigo archivos de mayor tamaño, lo que se traduce a mayor espacio de memoria y son menos manejables en su procesamiento (Iglesias, 2004).

Visión artificial

La visión artificial es una rama de la inteligencia artificial, la cual a través del uso de diferentes técnicas posibilita la obtención, procesamiento y análisis de la información

capturada como imagenes. Los procesos de visión artificial son posibles gracias a tecnologías basadas en la captura de imágenes mediante cámaras de video, cámaras web, y todo tipo de cámaras, sumado a ello la capacidad de procesamiento de los ordenadores (García E. , 2017).

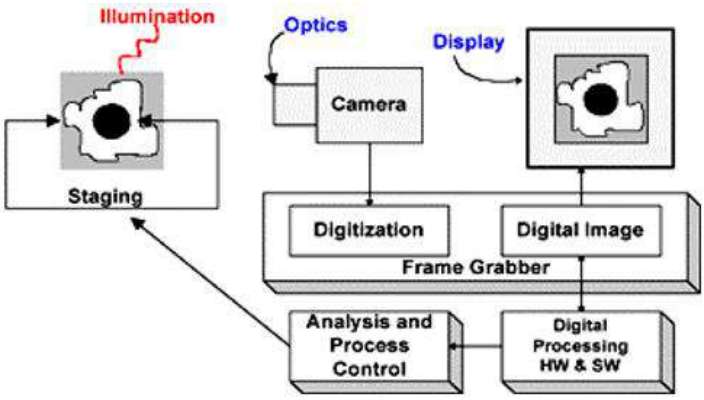
La visión artificial se basa en el sistema de visión humano, el cual permite obtener información del entorno en que se encuentra y requiere del tratamiento de la información obtenida por diferentes técnicas (Sucar & Gomez, 2013).

Componentes de un sistema de vision

Los elementos hardware mínimos necesarios para un sistema de visión artificial son los siguientes (ver Figura 15)

Figura 15

Diagrama de bloques de un Sistema SV



Nota. En la figura se puede ver el objeto observado, la cámara, y el proceso de digitalización que permite la visualización de la imagen digital. Tomado de "*Tecnicas y algoritmos básicos de vision artificial*", (Gonzales, et al., 2006).

Etapas de un proceso de visión artificial

Los pasos fundamentales para llevar a cabo una tarea de visión artificial son las siguientes:

- **Etapa sensorial:** Es necesario sensores y la capacidad para digitalizar la señal producida por el sensor, en otras palabras en esta etapa se realiza la adquisición de imágenes.
- **Etapa de reprocesado:** El siguiente paso consiste en el procesamiento de dicha imagen, mediante transformaciones y filtrado de imágenes, esto se hace con el objetivo mejorar la imagen.
- **Etapa de segmentación:** Aquí se divide la imagen en las partes que la constituyen o los objetos que la forman.
- **Etapa de parametrización:** Los datos de la imagen que salen del proceso de segmentación se deben convertir a una forma que sea apropiada para el ordenador.
- **Etapa de clasificación:** Es necesario especificar un método que extraiga los datos de interés.
- **Etapa de reconocimiento e interpretación:** El reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los descriptores. La interpretación da significado al conjunto de objetos reconocidos (Gonzales, et al., 2006).

Caracterización y Procesamiento de imágenes

Las imágenes captadas por nuestro sistema de visión son interpretadas por nuestro cerebro, este relaciona lo captado en el mundo exterior con información existente gracias a la capacidad del cerebro de caracterizar y diferenciar información visual. De igual manera en los sistemas de visión por computador para el procesamiento de las imágenes estas necesitan ser caracterizadas (Aguilar, Angulo, 2017) (Aguilar & Angulo, 2014), como por ejemplo por su textura, color, orillas, entre otros. Así, se obtiene una

interpretación consistente de las características obtenidas de objetos del dominio de interés, en base al conocimiento y las características se realizan el reconocimiento.

Antes de extraer características de una imagen, esta debe cumplir ciertas propiedades como las que se detallan en la Tabla 1. No necesariamente debe cumplir con todas estas características, depende en gran parte de la aplicación o hacia donde está orientado. Para este caso en particular las imágenes que se van a utilizar deben cumplir con las características mencionadas.

Tabla 1

Propiedades de una imagen

Propiedad	Descripción
Robustez	Ser capaz de detectar la misma ubicación de características halladas en los píxeles que forman la imagen independientemente de la escala, rotación, desplazamiento y ruido.
Repetibilidad	Ser capaz de detectar las mismas características en la imagen sometida a diferentes condiciones de visualización.
Precisión	Localizar con precisión la misma ubicación de características halladas en los píxeles que forman la imagen para tareas de semejanza entre estas.
Generalidad	Ser capaz de detectar características que se pueden usar en diferentes aplicaciones.
Eficiencia	El algoritmo de detección de características debería poder detectar características en nuevas imágenes rápidamente para admitir aplicaciones en tiempo real.
Cantidad	Capacidad de detección de la mayoría o la totalidad de características en los pixeles de la imagen reflejando su contenido para obtener una representación compacta de la imagen.

Nota. La tabla muestra las propiedades que debe cumplir una imagen para que esta se pueda caracterizar. Tomado de (Chasillacta, 2020).

Para poder reconocer los objetos es necesarios describirlos de alguna manera, dicha descripción resume las características obtenidas a partir de la imagen en los procesos de visión. Existen básicamente 3 tipos de descripciones los cuales se pueden combinar: modelos geométricos en 2D o 3D, características globales, características locales (Marin, Sucar, & Morales, 2007).

Modelos geométricos: Se basan como el propio nombre lo describe, mediante la forma del objeto, estos pueden ser en 2 o 3 dimensiones. Se basan en el contorno (superficie) o en la región (volumen). Lo que se busca es que sean invariantes ante cambios de escala, rotación, traslación y robustez ante el ruido y cambios de iluminación (Sucar L. , 2018).

Características globales: Caracterizan la región correspondiente al objeto, normalmente después de un proceso de segmentación, aunque también se puede aplicar a toda la imagen. Se consideran 3 tipos de características: Color, texturas y forma.

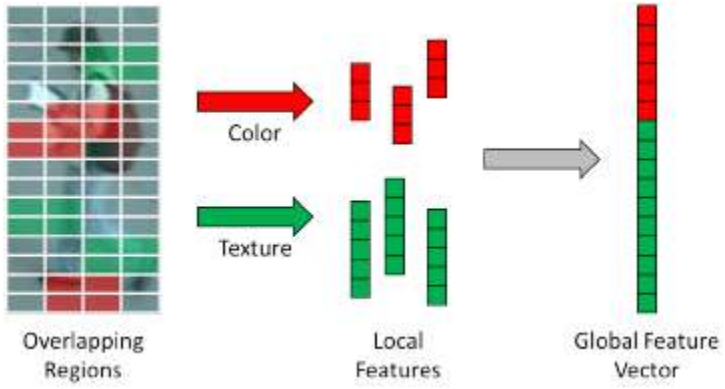
- **Color:** Tratan de caracterizar la distribución de color de un objeto o clases de objetos. Se consideran diferentes modelos de color como RGB, HSI, YIQ, LuV, entre otros.
- **Textura:** Describen la clase de textura del objeto, se busca caracterizar a la textura con pocos parámetros mediante técnicas como la respuesta a un conjunto de filtros a diferentes escala.
- **Forma:** Caracterizan la forma geométrica del objeto (región), normalmente en 2D. Para ello se pueden utilizar las descripciones o características globales que hemos visto para los modelos geométricos (Sucar L. , 2018).

Características locales

Una característica local describe las propiedades de un pixel con relación a los pixeles vecinos, proporcionan información sobre estructuras relevantes de la imagen, tales como regiones o contornos como se puede observar en la Figura 16. En visión por computador una característica muy utilizada es los contornos ya que gracias a ello se puede determinar un objeto en una imagen. Las características locales utilizan descriptores de características locales extraídos de la región de interés llamados keypoints (Hassaballah, Abdelmgeid, & Alshazly, 2016).

Figura 16

Características de una imagen



Nota. El gráfico muestra las características globales y locales de una imagen. Tomado de "*Relaxed Pairwise Learned Metric for Person Re-Identification*", (Hirzer, et al., 2012).

La ventaja de utilizar características locales es que por ejemplo aplicaciones de búsqueda de imágenes a gran escala se tendrá un mejor rendimiento que al usar características globales, debido a solo se procesan los datos más relevantes de la imagen (Hassaballah, Abdelmgeid, & Alshazly, 2016) (Bianco, et al., 2015).

Puntos de interés de una imagen

Los puntos de interés (característicos) o features points se basa en la detección, descripción y correspondencia de puntos característicos los cuales son fundamentales al tratarse de sistemas de visión por computador (Lowe, 2004) (Aguilar & Angulo, 2014). Los features points son la base de muchas aplicaciones como seguimiento de objetos, identificación de objetos, registro de imágenes, reconstrucción 3D, entre otros.

Detección

Los detectores de características se pueden clasificar ampliamente en tres categorías: detectores de una sola escala (single-scale detector), detectores de múltiples escalas (multi-scale detector) y detectores afines invariantes (affine invariant detectors).

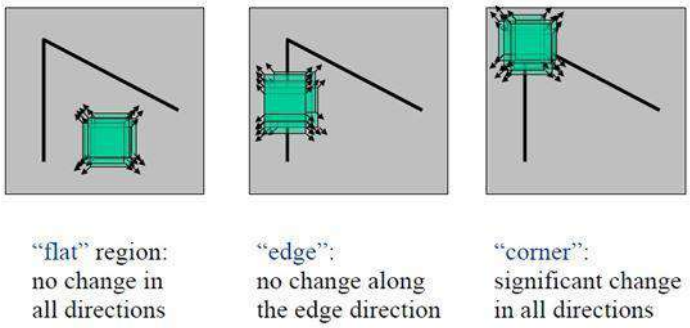
Los detectores en single-scale son invariables a los cambios de la imagen, como la rotación, la traslación, la variación en la iluminación y el aumento de ruido. Mientras que los detectores multi-scale tiene las mismas características que los single-scale, con la diferencia de que este es invariante a transformaciones de imagen en escala. La detección de puntos característicos se ocupa en la extracción de esquinas, cruces, intersección de líneas, entre otros (Mikolajzyk & Tuytelaars, 2005). Para ello se han desarrollado varios algoritmos detectores entre los cuales se puede mencionar los siguientes:

Detector Moravec

El algoritmo de Moravec es uno de los primeros algoritmos para la detección de esquinas (Moravec, 1980). El objetivo del algoritmo es probar cada pixel en una imagen mediante una pequeña ventana y buscar si hay alguna esquina presente (o un gran cambio en intensidad), como se muestra en la Figura 17.

Figura 17

Funcionamiento del detector Moravec



Nota. La figura muestra como la ventana se mueve hasta encontrar un cambio significante de intensidad en todas las direcciones, determinando la esquina de determinada región de interés. Tomado de *"Evaluación comparativa de técnicas de detección y descripción de puntos de interés en imágenes [Tesis Pregrado]",* (Redondo, 2016).

Detector Harris

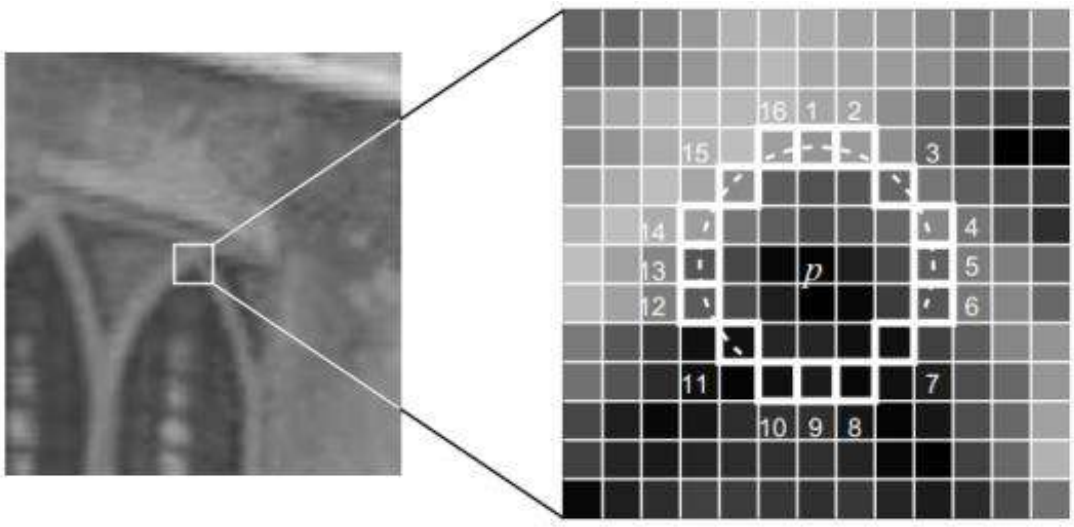
En lugar de usar una ventana local, se usa una ventana circular suavizada para lidiar con el ruido y otras deformaciones de la imagen. Existen varias propuestas modificadas de este algoritmo (Mikolajczyk & Schmid, 2005).

Detector FAST

FAST (Features from Accelerated Segment Test) considera un círculo de 16 píxeles alrededor del candidato de esquina p. Denominaremos n a los píxeles continuos en el círculo que tengan mayor intensidad de brillo que el pixel candidato Ip. En el ejemplo a continuación (Figura 18) se ha escogido n =12 ya que permite una prueba rápida que excluye un gran número de píxeles no esquina (Andrade, 2015).

Figura 18

Representación esquemática del detector Fast



Nota. La figura muestra el área que considera el detector Harris, esta prueba solo examina los píxeles 1, 5, 9 y 13 (los cuatro puntos cardinales). Tomado de *"Correspondencia multi espectral en el espacio de houg [Tesis Pregrado]",* (Andrade, 2015).

Detector ORB

ORB (Oriented FAST and Rotated BRIEF) es una versión ampliada del algoritmo FAST (Rublee, et al., 2011). La extensión se obtiene acelerando el proceso de detección, haciéndolo invariable a los cambios de escala y proporcionando una estimación precisa de las orientaciones de los puntos característicos. El descriptor ORB es una versión extendida e invariante de rotación de BRIEF. La invariancia de rotación se obtiene analizando la varianza de los descriptores BRIEF y calculando la correlación entre los descriptores. Se aplica un método de aprendizaje para des correlacionar los descriptores BRIEF. Esto hace que el descriptor BRIEF sea invariable a los cambios de rotación. ORB es un descriptor binario.

Descripción

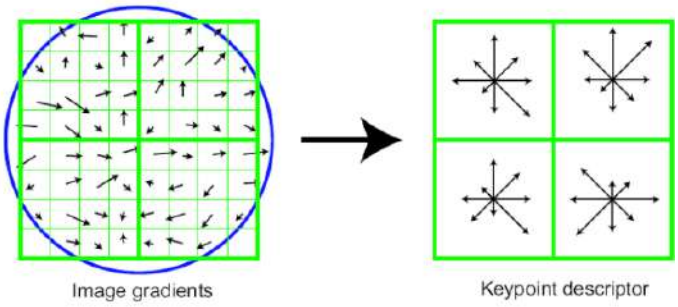
La descripción de puntos característicos es la asignación de distintos vectores descriptores a los puntos característicos obtenidos para que la correspondencia de puntos se puedan establecer de manera confiable entre las imágenes al hacer coincidir los vectores descriptores de puntos característicos (Mikolajczyk & Schmid, 2005). Hay un gran número de descriptores de características en la literatura; alguno de los más usados son los siguientes:

Descriptor SIFT

SIFT (Scale Invariant Feature Transform) es un descriptor que se basa en gradientes de imagen alrededor de los puntos característicos de SIFT. Los gradientes se dividen espacialmente en contenedores de ubicación de 4x44. Para cada ubicación, se calcula un histograma de gradientes orientados como en la Figura 19. Los histogramas se concatenan en todos los contenedores de ubicación para obtener un descriptor SIFT de tamaño 128. El descriptor SIFT se usa ampliamente. Se han propuesto varias modificaciones del descriptor SIFT, como el Histograma de Orientación y Ubicación de Gradiente (GLOH) (Mikolajczyk & Schmid, 2005) y Color-SIFT (Van de Sande, Geves, & Snoek, 2010).

Figura 19

Representación esquemática del descriptor SIFT



Nota. La gráfica muestra como la magnitud y orientación del gradiente son ponderados por una ventana circular Gaussiana (circulo azul) y como la región es dividida en otra matriz. Tomado de (Hassaballah, Abdelmgeid, & Alshazly, 2016).

Descriptor SURF

SURF (Speeded Up Robust Features) (Bay & Tuyelaars, 2006), es un descriptor SURF que se basa en las respuestas de las ondas de Haar. Las respuestas se calculan en la ubicación de cada píxel y luego las respuestas se dividen espacialmente en contenedores de ubicación de 4x4. Se obtiene un vector de cuatro dimensiones para cada ubicación. El vector contiene la suma de las respuestas de las ondas de Haar en direcciones horizontales y verticales y también la suma de los valores absolutos de las respuestas en ambas direcciones. Finalmente, los vectores se concatenan en todos los contenedores de ubicación para obtener un descriptor SURF de 16×4 = 64 dimensiones. El tamaño del descriptor SURF es más pequeño que SIFT.

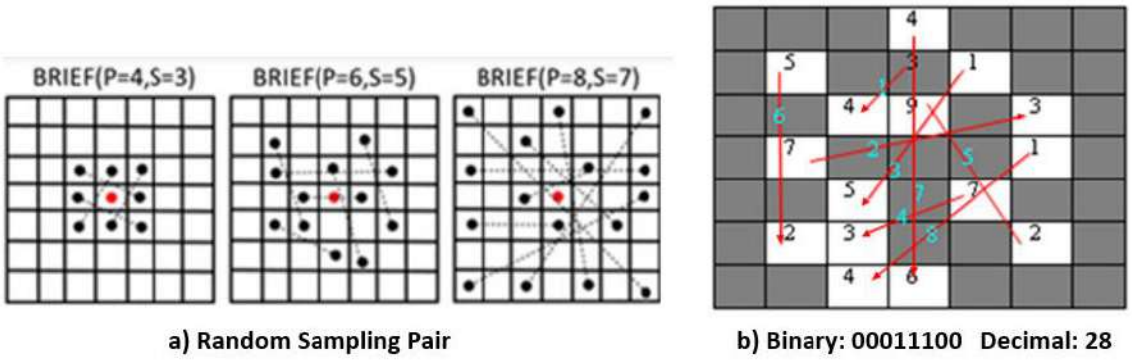
Descriptor BRIEF

Características elementales independientes robustas binarias (BRIEF) se basa en un patrón de muestreo (Candoler, et al., 2010). El patrón de muestreo selecciona al azar

puntos alrededor del punto de característico y las intensidades de píxeles de los puntos seleccionados se comparan. BRIEF es un método de construcción de descriptores de alta velocidad, eficiente en el almacenamiento y la tasa de reconocimiento. Es un descriptor binario y la coincidencia se realiza con la distancia de Hamming, que es computacionalmente eficiente que la distancia euclidiana utilizada para la coincidencia de descriptores SIFT, SURF y CS-LBP. BRIEF no es invariante a los cambios de rotación.

Figura 20

Representación esquemática del descriptor BRIEF



Nota. La gráfica muestra: a) describen el patrón de muestreo aleatorio utilizado por BRIEF y b) muestra el cálculo de BRIEF en una ventana de 7×7. Tomado de (Hassaballah, Abdelmgeid, & Alshazly, 2016).

La Figura 20, P y S se refieren al número de pares de muestras y el tamaño de la ventana, respectivamente. Las celdas blancas, las líneas rojas y el número verde en las líneas rojas se refieren a la intensidad del píxel, los pares para comparar y la secuencia de comparaciones por pares para generar el valor binario.

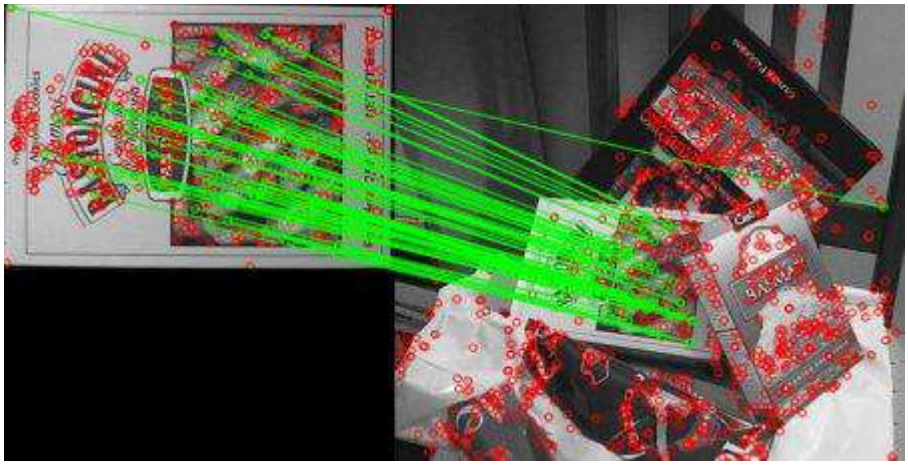
Correspondencia de características

En la correspondencia de puntos de interés (Feature Matching), se comparan los descriptores de las imágenes entre ellos para identificar características similares. Para

dos imágenes se puede conseguir un conjunto de pares, donde es una característica de la primera imagen y su correspondencia en la otra como se muestra en la Figura 21.

Figura 21

Correspondencia de puntos característicos entre dos imágenes



Nota. En la gráfica se observa los puntos característicos (círculos rojos) y la correspondencia entre las dos imágenes mediante la línea verde. Tomado de *"Feature Matching*", (Mordvintsev & Abid, 2013).

Fuerza Bruta

La correspondencia por fuerza bruta (Brute Force) es un algoritmo simple. Toma cada descriptor de la primera imagen y lo corresponde con todas las características de la segunda imagen usando cálculos por distancia y seleccionando el punto más cercano. La desventaja es el tiempo de ejecución, aumenta de forma exponencial con respecto al aumento de conjunto de datos (Narro, 2019).

FLANN

Es una librería que contiene una colección de algoritmos optimizados para búsquedas de puntos más cercanos en grande conjuntos de datos. El rendimiento es

mejor y más rápido en grandes conjuntos de datos frente a los algoritmos de fuerza bruta (Narro, 2019).

Transformaciones geométricas

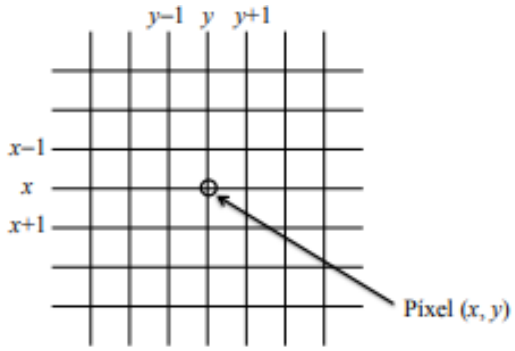
Se trata del manejo de la imagen como una variación en la ubicación de los píxeles mas no alterar su valor original. Estas transformaciones producen un cambio en la distribución de los píxeles de la imagen original respecto a un sistema de coordenadas, dando lugar a una transformación geométrica de la imagen inicial (De La Escalera, 2001) (Pajares, et al., 2003) (Pajares & de la Cruz, 2007). También se debe tener presente que al realizar alguna de las transformaciones conlleva a la alteración de los valores de intensidad como resultado de dicha transformación.

En muchos de los casos, para el análisis de una imagen es necesario centrarse en un área específica de toda la imagen, a esto se lo llama Región de Interés (del inglés Region Of Interest - ROI). La operación geométrica tiene como finalidad convertir los valores de una imagen a otros, tal y como se podría percibirse desde otro punto de vista, dicho de otra manera permite modificar las coordenadas espaciales de la imagen (Hartley & Zisserman, 2003). De tal manera, operaciones como el escalado de una imagen no son más el acercamiento o alejamiento desde un punto de vista, La rotación se traduce como un giro en la perspectiva, y la traslación se interpreta como a desplazamiento de un punto a otro (De La Escalera, 2001).

El primer paso previo a aplicar una operación geométrica radica en observar la distribución de los píxeles en la imagen original y en la transformada Figura 22.

Figura 22

Distribución de píxeles de una imagen

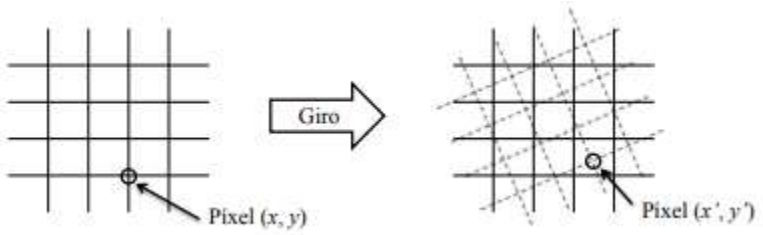


Nota. La gráfica muestra una representación de una imagen mediante pixeles. Tomado de "*Conceptos y Métodos en Visión por Computador*", (De La Escalera, 2001).

Al realizar una transformación en una imagen, ya sea por un desplazamiento, un giro o una aproximación, los nuevos píxeles por lo general quedarán situados sobre puntos intermedios, como muestra la Figura 23, donde la imagen (línea continua) muestra la disposición convencional de los píxeles de la imagen original, mientras que la imagen con rejilla discontinua refleja cómo quedaría la imagen original después de girarla un ángulo determinado. De esta manera los píxeles deben proyectarse sobre los de la imagen final, con una estructura similar a la de la imagen original.

Figura 23

Transformación geométrica aplicada a una imagen



Nota. En la gráfica se muestra una transformación geométrica, la cual suele provocar que los pixeles de la imagen transformada no coincidan con los pixeles de la imagen final. Tomado de "*Conceptos y Métodos en Visión por Computador*", (De La Escalera, 2001).

Una vez aplicado una transformación geométrica a una imagen, es fundamental alcanzar y determinar los valores de intensidad relacionado con la transformación ejecutada, de manera que la imagen inicial sea transformada geométricamente pero con los valores de intensidad correspondientes a la imagen original.

Transformaciones básicas

A continuación se detallan las transformaciones elementales: traslación, rotación y escalado, estas aplicadas a una imagen con sistemas de coordenadas bidimensional asociados a ella. Para la explicación de estas transformaciones se supone que la imagen original tiene coordenadas (i, j), mientras que la imagen resultante posee coordenadas (x, y).

Traslación

Consiste en trasladar el pixel (x, y) en la imagen hasta que alcance la posición $(x + a, y + b)$ y sustituir el valor de la intensidad en $(x + a, y + b)$ por el correspondiente a la posición (x, y) de la imagen original, La transformación se muestra en la siguiente ecuación (Li, et al., 2019).

$$x = x + a$$
$$y = y + b \tag{9}$$

Rotación

La rotación se diferencia de la traslación en que la transformación es un giro. La rotación de un θ con respecto al origen de coordenadas de la imagen. La aplicación de esta transformación es para simular determinadas situaciones de giro de la cámara, o el giro del objeto en situaciones de movimiento, viene dada por la siguiente transformación (Li, et al., 2019).

$$x = \cos(\theta_i) - sen\left(\theta_j\right)$$
$$y = \text{sen}(\theta_i) + cos\left(\theta_j\right) \tag{10}$$

Escalado

Esta transformación consiste en variar el tamaño de la imagen original, esta variación se puede realizar en cualquiera de los ejes de coordenadas x e y. De esta manera se representa mediante factores de escala S_x y S_y en las direcciones x e y respectivamente. Es necesario tener en cuenta que cuando el factor de escala toma valores entre 0 y 1 se produce una reducción de la imagen, mientras que si son mayores a 1 se produce un aumento en la imagen original. La siguiente ecuación muestra la transformación (Li, et al., 2019).

$$x = S_x x$$
$$y = S_y y \tag{11}$$

Transformación rígida

Esta transformación se puede descomponer como una transformación de rotación seguida de una transformación de traslación. Se caracteriza por conservar la distancia euclidiana entre cada par de puntos.

Transformación de similitud

Esta transformación es una composición de transformaciones elementales como la traslación, rotación y escalado, con la característica que se conserva el ángulo. Viene dada por la siguiente representación matricial (12).

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s.\cos(\theta) & -s.\sin(\theta) & t_x \\ s.\sin(\theta) & s.\cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (12)$$

También puede representarse de forma más sencilla en forma de bloque:

$$x' = H_s x = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix} x \quad (13)$$

Donde s representa la escala isotrópica.

Transformación afín

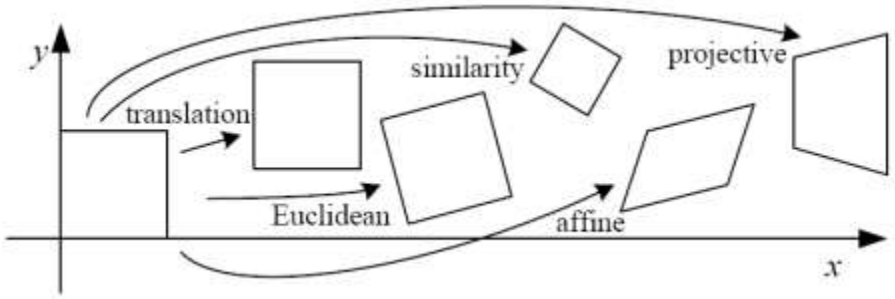
Esta transformación es una composición de transformaciones elementales como la traslación, rotación, escalado e inclinación, con la característica que se conserva el paralelismo. La representación matricial viene dada en la expresión (14).

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (14)$$

Para entender todos los conceptos mencionados sobre transformaciones geométricas y sobre algunas de las muchas transformaciones que existen se presenta la Figura 24, en donde se puede observar el efecto de realizar una transformación geométrica a una imagen, en este caso un cuadrado ubicado en el primer cuadrante del sistema de referencia x, y.

Figura 24

Transformaciones geométricas 2D



Nota. En la gráfica se muestra el efecto que causa al aplicar una transformación geométrica a un objeto 2D como puede ser una imagen capturada por el micro UAV. Tomado de "*2D Geometrical Transformations*", (University of Nevada, Reno, 2018).

Planificador de camino

El planificador de camino o "*path planning*" se define como encontrar un movimiento sin colisiones entre una configuración inicial (inicio) y una final (meta) dentro de un entorno específico. La situación más simple es cuando la ruta debe planificarse en un entorno estático y conocido. Sin embargo, no siempre pasa eso, por ello el problema de planificación de rutas de manera general se puede formular para un entorno dinámico y desconocido (Gasparetto, et al., 2015).

La planificación del camino es una cuestión netamente geométrica, porque se define como la generación de una trayectoria geométrica, sin mencionar ninguna ley de tiempo especificada. Estos planificadores deben de generar rutas que se adapten a la geometría del entorno. Las rutas generadas deben de ser totalmente seguras, manteniendo una distancia de seguridad con todos los obstáculos, minimizando así el riesgo de accidente con el entorno (Gonzales, Monje, & Balaguer, 2015) (Espitia & Sofrony, 2011).

Existen muchas estrategias de planificación de caminos publicadas en la literatura. De los métodos para la planeación de caminos se tienen, en un primer lugar, los basados en grafos, de los cuales se distinguen: grafos de visibilidad, diagramas de Voronoi modelado del espacio libre y descomposición en celdas. Otro tipo de métodos empleados para la planeación de trayectorias parten de un enfoque estocástico, dando lugar al algoritmo de planeación aleatoria de trayectorias (RPP, Randomized Path Planner), al algoritmo de mapas probabilísticos (PRM, Probabilistic Road Map Method) y al algoritmo de árboles de exploración rápida (RRT, Rapidly Exploring Random Tree) (Lindemann & La Valle, 2004) (Elbanhawai & Simic, 2014) (Ismail, Tang, & Khaksar, 2012). Para el desarrollo del proyecto se enfatizará en los algoritmos RRT y A*.

Capitulo III

Generalidades del sistema

Introducción

De entre tantas aplicaciones que se pueden realizar con un UAV, conseguir una navegación autónoma es un reto debido a todos los aspectos que se deben tener en cuenta. Para este proyecto de investigación es necesario definir el hardware y software del sistema que se va desarrollar. Como hardware se refiere a elementos/dispositivos físicos como son la estación terrestre y la estación aérea, mientras que como software se tiene el sistema operativo en el que se desarrolla el proyecto, así también los diferentes entornos de desarrollo como ROS, Python, librerías como OpenCV, numpy, entre otros.

Hardware y software del sistema

Para este proyecto de investigación el hardware que viene a ser el UAV es proporcionado por el Centro de Investigación de Aplicaciones Militares (CICTE), mientras que la estación terrestre es una PC portátil de uso comercial, como se muestra en la Figura 25. El software hace referencia a diferentes programas de código abierto como el sistema Operativo Ubuntu, paquetes de computer visión como OpenCV, ROS, entre otros.

Figura 25

Hardware del sistema

Nota. El gráfico muestra la estación terrestre y aérea (micro UAV) del sistema.

Estación aérea – micro UAV Parrot Bebop 2

Este micro UAV viene a ser la estación aérea, como se mencionó el Parrot Bebop 2 (Figura 26) es uno de los UAV que se usa por el CICTE para desarrollo de diferentes aplicaciones ya sea en el ámbito militar o civil, la gran ventaja de este micro UAV frente a otras es que es una plataforma de código abierto lo cual permite acceder a datos como estado de batería, cámara, datos de edometría, así como enviar acciones de control como ascenso/descenso, acciones de desplazamientos, control de ángulos de inclinación de la cámara, entre otros, los mismos que pueden ser usados a conveniencia para el desarrollo de aplicaciones. Entre las cuales se puede mencionar Planificación de rutas para inspección de infraestructuras (Besada, et al., 2018), evasión de obstáculos basados en computer visión (Chakravarty, et al., 2017), técnicas de control simples y avanzadas (Orozco-Soto, et al., 2018) (Rosaldo-Serrano & Aranda-Bricaire, 2019), técnicas de SLAM (Stumberg, et al., 2017), interacción con el usuario mediante gestos (Monajjemi, Mohaimenianpour, & Vaughan, 2016), entre otros. Todos estos estudios son posibles debido a las características del Bebop y completo software de código abierto (Giernacki, et al., 2020).

Figura 26

Parrot Bebop 2

Nota. El gráfico muestra la estructura física del micro UAV usado para el proyecto. Tomado de "*Support - Parrot Bebop 2 | Parrot Official*", (Parrot, 2020).

Especificaciones generales

Bebop 2 es un micro UAV de tamaño compacto y bajo costo producido por la empresa francesa Parrot Drones, es considerado un dron de categoría media pero bastante superior a los que se consideran drones básicos. Lo más relevante su autonomía de vuelo, que puede llegar hasta los 25 minutos, 4 rotores potentes y hélices eficientes, capaz de sobrevolar en condiciones extremas y con un sistema de seguridad en caso de perder el enlace de conexión con el control de mando. A continuación, en la Tabla 2 se muestra algunas de las características generales de este micro UAV.

Tabla 2

Especificaciones de dron Parrot Bebop 2

Características	Descripción
Sistema de rotores	Cuatro rotores, tres hojas por cada hélice (5 centímetros de diámetro, son de plástico flexible, bastante resistentes y fácilmente reemplazables).
Velocidad máxima horizontal	18m/s. Tarda unos 14 segundos en conseguir su máxima velocidad, sin oposición de viento
Velocidad de ascenso máxima	6m/s. En condiciones favorables llega a una altura de 100 metros en 20 segundos.
Alcance de señal	300 metros.
Autonomía	La batería es de 2.700 mAh, Li-ion. Autonomía, según el fabricante de 25 minutos de vuelo.
Memoria	8GB de memoria interna, accesible vía microUSB
Cámara	Cámara de 14 megapíxeles, Cuenta con estabilización de imagen.

Características	Descripción
Métodos de control	Dos métodos: teléfono móvil (iOS/Android) y mando SkyController.
Peso y dimensiones	Tiene unas dimensiones de 33 x 30 x 10 centímetros, y pesa 480 gramos.

Nota. La tabla muestra especificaciones de manera general del micro UAV Parrot Bebop 2.Tomado de "*Support - Parrot Bebop 2* | *Parrot Official*", (Parrot, 2020).

El Bebop 2, también dispone de diferentes sensores y funciones como: sensor de ultrasonido y barómetro, acelerómetro, magnetómetro, giroscopio, GPS y cámara de estabilización vertical. Todos estos dan ese potencial que se requiere en un dron de gama media, convirtiéndolo en una muy buena opción para aplicaciones de desarrollo basado en el Parrot Bebop 2.

Especificaciones de cámara

Algo que destaca en este dron es la cámara que dispone, su calidad y estabilidad generan una experiencia en captura de fotos y grabación de videos, puede grabar videos en una resolución full HD. El campo de visión es de 180°, centrado en determinada sección de toda la imagen (el resto de secciones están enfocadas para la estabilización digital del video). A continuación, en la Tabla 3 se muestra a detalle las especificaciones técnicas de la cámara. Especificaciones como la resolución, si cuenta con algún tipo de sensor, si dispone de sistemas de establibilizacion para el video, son las más importantes y que se encuentran disponibles para cualquier usuario en las páginas oficiales de los fabricantes de los UAVs

Tabla 3

Especificaciones de la cámara del Parrot Bebop 2

Características	Descripción
Cámara	14 mega pixeles con sensor CMOS
Estabilización de video	Sistema digital de 3 ejes
Resolución de video	1920 × 1080p a (24, 25 o 30 fps)
Codificación de video	H264
Resolución de fotografía	4096 × 3072 pixeles
Formato de fotografía	JPEG, RAW, DNG

Nota. La tabla muestra características técnicas de la cámara que dispone el micro UAV Parrot Bebop 2.Tomado de "*Support - Parrot Bebop 2* | *Parrot Official*", (Parrot, 2020).

Especificaciones de conectividad

El Parrot Bebop 2 genera una red inalámbrica Wi-Fi de 2.4GHz la cual es visible para el resto de dispositivos como ordenadores y smartphones, estos últimos se pueden conectar al dron mediante la propia aplicación desarrollada por Parrot llamada Freeflight previamente instalada en los dispositivos inteligentes. Esta aplicación está disponible tanto en Android como IOS. Puede llegar a tener un alcance máximo de 300 metros siempre y cuando se tenga línea de vista.

Otra forma de conectividad es mediante el mando remoto (Parrot Skycontroller 2) con un alcance máximo de 2 Kilómetros con línea de vista, esto gracias a la integración de una antena. Existe también el driver desarrollado por SDK (Software Development Kit) llamado "bebop_autonomy", este driver permite interactuar con el micro UAV mediante

plataformas de desarrollo como ROS sobre una terminal Linux, este driver es el que se utiliza para el desarrollo del proyecto.

Tabla 4

Especificaciones técnicas de conectividad

Características	Descripción
Tipo de conexión inalámbrica	Wi-Fi IEEE 802.11 a /b /g /n /ac
Network	MIMO Dual band
Antenas Wi-Fi	Antenas dipolares duales de 2.4 y 5 GHz
Rango de señal	300 m con dispositivo móvil 2 km con control remoto Parrot Skycontroller 2

Nota. En esta tabla se muestra las especificaciones de conectividad inalámbrica que dispone el micro UAV Parrot Bebop 2.Tomado de "*Support - Parrot Bebop 2 | Parrot Official*", (Parrot, 2020).

Estación terrestre

Para este proyecto de investigación la estación terrestre viene a ser un ordenador portátil, aquí es en donde se realiza todo el procesamiento de imágenes provenientes del micro UAV en tiempo real. A continuación se detallan las características del ordenador portátil utilizado.

- Marca y modelo: Dell Inspiron 15 5000 Series
- Procesador Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz
- Memoria RAM y tipo de sistema: 8 GB y 64 bits respectivamente
- Tarjeta gráfica: NVIDIA GeForce 920M

Ubuntu

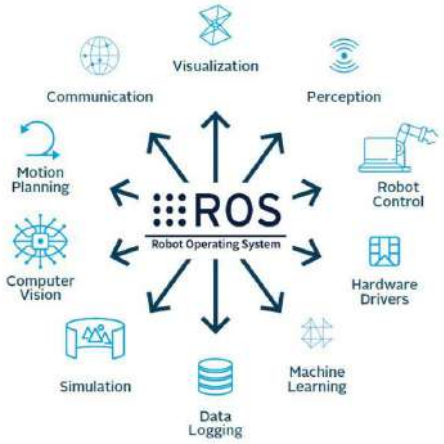
Para esta investigación se dispone de una distribución del Sistema Operativo Linux llamada Ubuntu, específicamente Ubuntu 16.04 LTS. Una de las ventajas de utilizar esta distribución que es libre y de código abierto, además de tener la versión LTS (Long Term Support) que significa soporte a largo plazo, es una de las versiones en donde lo más importante es la estabilidad, también permite integrar diferentes bibliotecas/herramientas para desarrolladores como por ejemplo OpenCV para el procesamiento de imágenes, Python, ROS, librerías de procesamiento matemático como numpy, entre otros que pueden ser agregadas desde repositorios.

Sistema Operativo Robótico – ROS

El Sistema operative robotico ROS (en inglés Robot Operating System, ROS) es una plataforma de código abierto para escribir software de robots (Figura 27). Se trata de una colección de herramientas y librerías las que ayudan a simplificar las tareas de crear estructuras de robot complejas (Serrano, 2011) (Salcedo, 2018). Al tener una arquitectura característica distribuida donde el procesamiento toma lugar en “nodos”, permite separar procesos complejos en simples, de esta manera se reduce los tiempos de ejecución y mejora el rendimiento de las aplicaciones porque emplea una topología punto a punto (Quigley, Gerkey, & Smart, 2015). Gracias a que permiten la reutilización de códigos en nuevos programas se puede agregar fácilmente librerías desde los repositorios, también es importante mencionar los diferentes lenguajes de programación que permite ROS como: C++, Python, Octave, LISP, LUA, y muchas otras como java que aún en fase experimental pero apoyada por google (García A. , 2013).

Figura 27

Sistema Operativo Robótico ROS



Nota. El gráfico muestra las diferentes aplicaciones que dispone ROS. Tomado de "*AutomationWare es Robótica Colaborativa*", Economía Del Hoy (https://www.economiadehoy.es/automationware-es-robotica-colaborativa)

Componentes de ROS

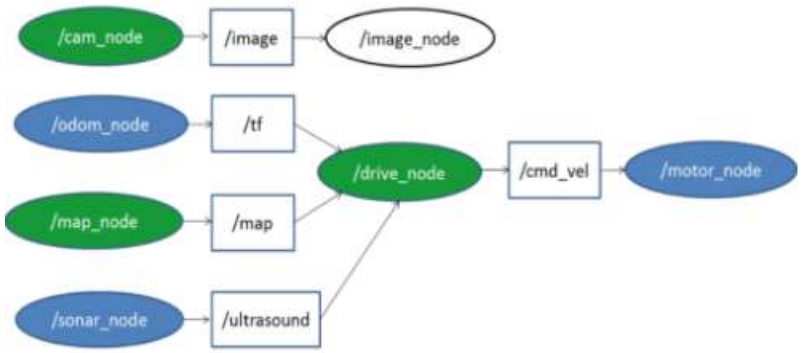
- **Tópicos** (topic): Son canales de información entre los nodos, no es más que un mensaje que se envía (existen distintos tipos de clases de mensajes). Un nodo puede emitir o suscribirse a un tópico.
- **Paquete (package):** ROS está estructurado en paquetes, estos pueden incluir nodos, librerías, datos, o módulos.
- **Nodos:** Son los procesos que ejecutan la comunicación de manera independiente, aunque todos estén interconectados entre sí. Los nodos se combinan dentro de un grafo, compartiendo información entre ellos para ejecuciones complejas.
- **Pila (stack):** Conjunto de nodos que juntos proporcionan alguna funcionalidad.

- **Mensajes:** Los nodos se comunican mediante el paso de mensajes, los cuales son una estructura simple de datos como integer, floatinf, point, boolean, entre otros.
- **Servicios:** La petición y respuesta se lo realiza atraves de los servicios, se usa cuando se requiere una comunicación síncrona bidireccional (Tormes, García, & Benítez, 2007).

A continuación se muestra un ejemplo de implementación de nodos, nodos que controlan los actuadores y nodos que leen la información de los sensores, ver Figura 28.

Figura 28

Grafos de nodos y tópicos de ROS



Nota. La gráfica muestra los nodos y tópicos para el control de velocidad de un robot. Tomado de "*Implementación de sistema operativo robótico en una plataforma de robot móvil*", (Miguélez Machado, et al., 2020).

Instalación de ROS

ROS puede ser ejecutado sobre algunas plataformas basadas en UNIX, principalmente se lo ha probado en Ubuntu. La versión de ROS varía según la distribución y versión del sistema operativo, entre los cuales tenemos: ROS Jade Turtle, ROS Indigo Igloo, ROS Kinetic Kame, entre otros.

Para este proyecto ROS será ejecutado sobre Ubuntu 16.04 LTS, por lo que la distribución de ROS disponible es ROS Kinetic Kame. La guía de instalación está disponible en la página oficial de ROS (ROS.org), se recomienda seguir los pasos indicados en: http://wiki.ros.org/kinetic/Installation/Ubuntu

OpenCV

OpenCV (Open Source Computer Vision) es una biblioteca libre de visión artificial y machine learning desarrollada originalmente por Intel, que cuenta con más de 2500 algoritmos. Estos algoritmos permiten identificar objetos, rostros, clasificar acciones humanas en vídeo, hacer tracking de movimiento de objetos, encontrar imágenes similares, eliminar ojos rojos, reconocer escenarios, entre otros.

Una de las ventajas de esta biblioteca es que es multiplataforma, es decir, se puede ejecutar en diferentes sistemas operativos como Linux, Windows, Mac OS X, Android e iOS. También se puede utilizar en diferentes lenguajes de programación como Java, C, Python, entre otros. Para el desarrollo de este proyecto se utiliza el lenguaje de programación Python, debido a que es de código abierto, así como por la sencillez que presenta, haciendo muy corta la curva de aprendizaje de este lenguaje de programación.

Instalación de Python 2.7 con paquetes adicionales

Al instalar el sistema operativo Ubuntu 16.04 LTS en el ordenador, por defecto viene instalado una versión de Python, para este caso la versión 2.7. También se puede instalar otras versiones como la 3.0, esto de acuerdo a la versión de OpenCV que se desea instalar.

Para el desarrollo de aplicaciones en visión artificial no es suficiente con tener instalado el lenguaje de programación, para utilizar OpenCV es necesario ciertos

paquetes como Numpy, sciPy, Matplotlib, Pip, entre otros. Estos paquetes son bibliotecas de código abierto que facilitaran el trabajo.

- **NumPy:** Da soporte a vectores y arrays para Python.
- **SciPy:** Contiene herramientas y algoritmos matemáticos para Python.
- **Matplotlib:** Se utiliza para la generación de gráficos a partir de vectores y arrays.
- **Pip:** Gestor de paquetes para Python.

A continuación los comandos para instalar los paquetes mencionados, estos comandos deben ser ejecutados desde una terminal de Ubuntu, el gestor de paquetes pip ya viene instalado por defecto al instalar Python.

```
python -m pip install numpy
python -m pip install scipy
python -m pip install matplotlib
```

Instalación de OpenCV

Gracias al gestor de paquetes ya mencionado “pip” la instalación en muy sencilla, el siguiente comando instala la versión compatible de OpenCV con python. Si se dispone de Python 3 solo cambiar *pip* por *pip3* en la línea de comandos.

```
pip install opencv-python
```

Para comprobar que se ha instalado correctamente ejecutar el siguiente comando. Si la instalación fue exitosa se podrá importar OpenCV sin ningún problema.

```
Python – c “import cv2”
```

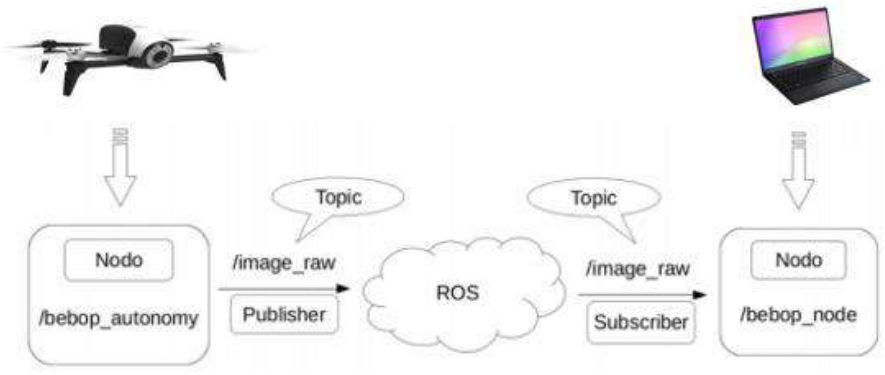
Es recomendable utilizar un entorno de desarrollo para programar en Python, como Sublime Text 3, el cual es un editor de código multiplataforma y ligero que permite estructurar el código mediante el coloreado de la sintaxis.

Comunicación entre estación terrestre y Parrot Bebop2

El sistema operativo robótico ROS al disponer de librerías para sistemas de comunicación de distintas plataformas robóticas, permite la comunicación entre la estación terrestre y el micro UAV Bebop2 mediante la librería *bebop_autonomy* de código abierto. Para establecer la comunicación de ROS con la estación terrestre (PC) y la estación aérea (Bebop2) lo primero que se debe realizar es establecer la conexión inalámbrica entre la PC y la red Wi-Fi generada por el micro UAV (esto se puede comprobar realizando un *ping* desde el terminal de la PC a la dirección IP 192.168.42.1), luego es necesario definir un nodo Maestro que este a su vez permita la comunicación entre los diferentes nodos que puedan existir, esto debido a ROS trabaja con una arquitectura Maestro-Esclavo (Valero, 2017). A su vez los nodos se comunican entre sí transmitiendo parámetros mediante tópicos o servicios (Figura 29).

Figura 29

Comunicación entre Bebop2 y PC mediante ROS



Nota. El gráfico muestra el esquema de comunicación mediante ROS del micro UAV Bebop2 y de la PC que viene a ser la estación terrestre. Tomada de (Valero, 2017).

Control de movimiento del UAV mediante "Bebop_autonomy"

Bebop_autonomy es un driver controlador de ROS para los cuadricópteros Bebop1 y Bebop 2, este controlador está basado en ARDroneSDK3 oficial de Parrot y fue desarrollado en los laboratorios de la Universidad Simon Fraser llamada Autonomy Lab. En la página oficial de ROS se puede encontrar más información al respecto, así como pasos para su instalación (Monajjemi M. , 2015).

Al trabajar con el controlador "bebop_autonomy", este genera diferentes nodos y tópicos a los cuales se pueden acceder mediante suscriptores (subscribers), de esta manera mediante la PC se puede acceder por los sensores del micro UAV Bebop2, también es posible enviar acciones de control al Bebop2 mediante publicadores (publishers) con el fin de maniobrar el micro UAV desde la estacón terrestre.

Ejecución de bebop_autonomy y comandos de ROS

Una vez que se ha establecido la comunicación inalámbrica entre la PC y el Micro UAV Bebop2, lo siguiente es generar un espacio de trabajo mediante el siguiente comando en la terminal de Ubuntu:

```
source ~/bebop_ws/devel/setup.[bash|zsh]
```

Luego se ejecuta el nodo maestro mediante (Si todo es correcto en la terminal debe aparecer el siguiente mensaje "started core service [/rosout]…"):

```
roscore
```

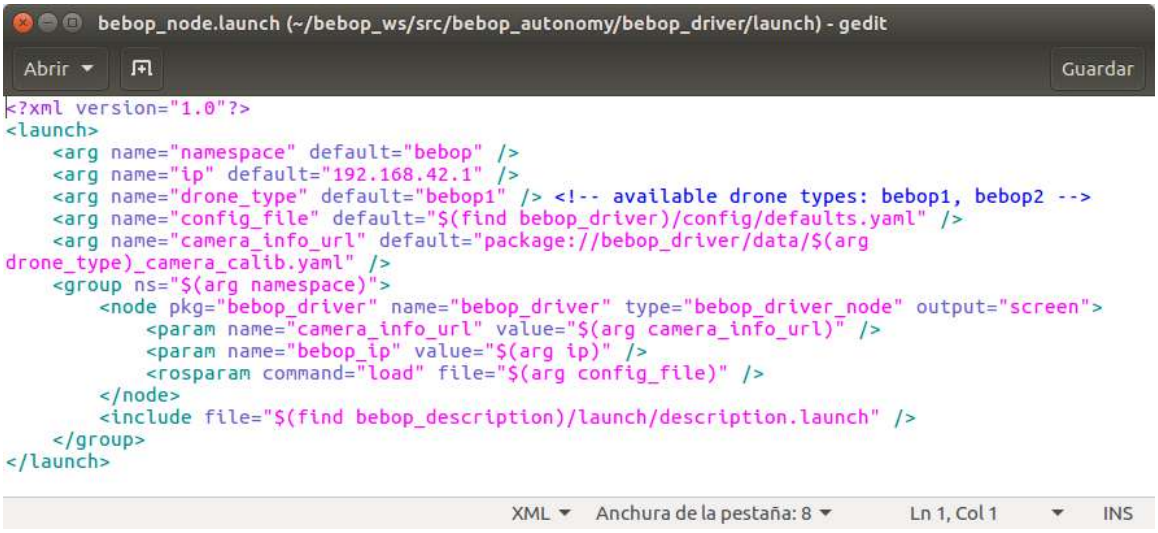
Se puede ejecutar la unidad ROS de Bebop un nodo ROS o como un nodo ROS independiente. Para ello se debe abrir una nueva pestaña y ejecutar el siguiente comando, el nodo se llama "bebop_driver_node" y existe en el paquete "bebop_driver":

```
roslaunch bebop_driver bebop_node.launch
```

Se recomienda usar la configuración predeterminada, aunque si se requiere es posible modificar esta configuración, como el nombre del nodo, la dirección IP del dron, entre otros.

Figura 30

Configuración de "bebop_node.launch"



Nota. El gráfico muestra la configuración general del archivo "bebop_node.launch", estos parámetros se pueden modificar.

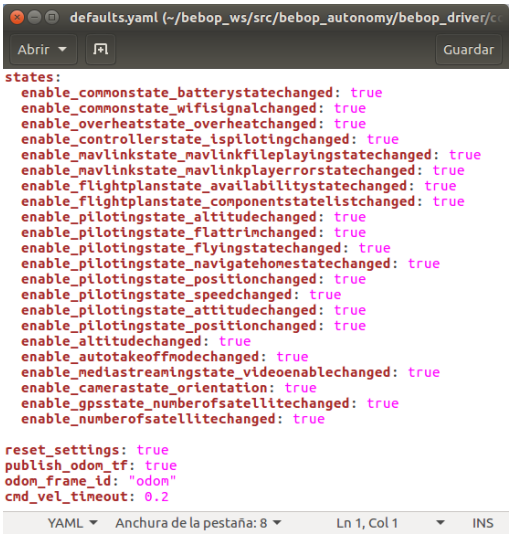
Otra manera de establecer la comunicación es mediante un Nodelet ROS, la diferencia al anterior es que aquí se puede visualizar el flujo de video del Bebop2, es una manera rápida de verificar si el controlador *bebop_autonomy* está funcionando correctamente. Para ejecutarlo se usa el siguiente comando:

```
roslaunch bebop_tools bebop_nodelet_iv.launch
```

Otro de los ficheros importantes de configuración es el archivo .yaml (Figura 31) en el cual se configura todos los parámetros iniciales del dron, este archivo se encuentra dentro de la carpeta “bebop_autonomy/bebop_driver/config/”. Es importante tener en cuenta que una vez ejecutada el driver los parámetros de este archivo no podrán ser cambiados, en el caso de necesitar algún cambio en la configuración inicial, se debe detener el driver y volverlo a ejecutar con los nuevos parámetros.

Figura 31

Archivo de configuración defaults.yaml



Nota. El gráfico muestra el archivo de la configuración inicial defaults.yaml, los parámetros mostrados son por defecto pero se pueden añadir más de acuerdo a los requerimientos.

Algunos de los parámetros que se pueden agregar a este archivo son: altitud mínima, altitud máxima, grados máximos de inclinación al momento del avance, parámetros de pilotaje, red, imagen, GPS, entre otros (Valero, 2017) (Grijalva, 2019) (Salcedo, 2018).

Tabla 5

Parámetros de configuración para el micro UAV Bebop2

Tipo de parámetro	Descripción y tipo de variable	Nombre del parámetro
Pilotaje	Altitud máxima en [m] (tipo Double)	*PilotingSettingsMax AltitudeCurrent*
	Angulo máximo de inclinación en [°] (tipo Double)	*PilotingSettingsMax TiltCurrent*
	Distancia máxima de separación desde el punto de partida en [m] (tipo Double)	*PilotingSettingsMax DistanceValue*
	Altitud mínima en [m] (tipo Double)	*PilotingSettingsMinAltitude Current*
Movimiento	Velocidad máxima vertical en [m/s] (tipo Double)	*SpeedSettingsMaxVertical SpeedCurrent*
	Velocidad de rotación máxima en [°/s] (tipo Double)	*SpeedSettingsMaxRotation SpeedCurrent*
	Velocidad de rotación máxima en pitch y roll en [°/s] (tipo Double)	*SpeedSettingsMaxPitchRoll RotationSpeedCurrent*
Red	Tipo de red Wi-Fi (tipo Int, donde 0: Selección automática y 1: Selección manual)	*NetworkSettingsWifi Selection Type*
Imagen	Número de fotogramas por segundo (tipo Int, donde 0: 23.967 fps, 1: 25 fps y 2: 29.97fps)	*PictureSettingsVideo FramerateFramerate*
	Resolución del video (tipo Int, donde 0: Grabación a 1080p – Streaming a 480p y 1: Grabación a 720p – Streaming a 720p)	*PictureSettingsVide ResolutionsType*

Nota. La tabla muestra todos los parámetros permitidos por el Bebop2 y son configurables desde el archivo .yaml.

Una vez ejecutado el driver se generan una serie de tópicos que permiten manejar todas las variables del UAV, nivel de batería, orientación de la cámara, recepción de imágenes captadas por la cámara integrada del UAV, controla las velocidades para tele

operarlo, entre otras. En la Tabla 6 se puede observar una lista de los tópicos mencionados anteriormente y que son usados para el desarrollo del proyecto.

Tabla 6

Principales Tópicos publicados y suscritos de "bebop_autonomy"

Suscriptor o Publicador	Tópico	Mensaje	Acción
Pub_Land	/bebop/land	std_msgs/Empty	Aterrizar
Pub_Takeoff	/bebop/takeoff	std_msgs/Empty	Despegar
Pub_Twist	/bebop/cmd_vel	geometry_msgs/Twist	Movimientos lineales en x,y,z del dron
Pub_CamTwist	/bebop/camera_control	geometry_msgs/Twist	Movimiento de la cámara virtual
Sub_Image	/bebop/image_raw	sensor_msgs/Image	Transmisión de imágenes
Sub_Odom	/bebop/odom	nav_msgs/Odometry	Obtiene los fotogramas de odometría
Sub_Battery	/bebop/states/common /CommonState /BatteryStateChanged	CommonCommonState BatteryStateChange	Estado de batería

Nota. La tabla muestra los tópicos más importantes, así como el tipo de mensaje y la acción que realiza cada uno de ellos en el micro UAV Bebop2.

Para hacer uso de estos tópicos es necesario de publicadores y subscriptores, como por ejemplo: en el caso de que se desea datos de estado de la batería o imágenes, es necesario de suscriptores, y para enviar acciones de control como el despegue, aterrizaje, avance al micro UAV (ver Tabla 7) es necesario de publicadores.

Tabla 7

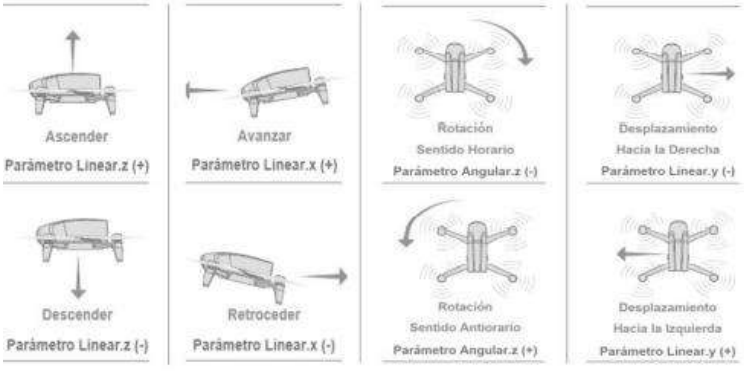
Acciones de navegación del tópico cmd_vel

Parámetro	Valor	Acción
Linear.x	Positivo	Pitch adelante
	Negativo	Pitch atrás
Linear.y	Positivo	Roll izquierda
	Negativo	Roll derecha
Linear.z	Positivo	Ascender
	Negativo	Descender
Angular.z	Positivo	Yaw Rotar antihorario
	Negativo	Yaw Rotar horario

Nota. La tabla muestra todas las acciones que dispone el tópico cmd_vel, el rango de velocidades que permiten van desde [-1.0 …1.0]. Si se desea poner en estado estático solo se debe poner a cero aquellos parámetros.

Figura 32

Movimientos generados por el tópico cmd_vel



Nota. La gráfica muestra todos los movimientos que el Parrrot Bebop2 permite. Así como los parámetros que requiere para conseguir ese movimiento. Tomada de (Chauca, 2020).

De manera gráfica se puede visualizar la acción de cada uno de los mensajes como ilustra en la Figura 32. Para el desarrollo de este proyecto las utilizadas son los movimientos a lo largo del eje x, es decir Pitch y movimientos rotacionales sobre el eje z que viene a ser Yaw.

Obtención de imágenes del Bebop2

Una parte fundamental es la obtención de imágenes del micro UAV, para esto es necesario una conversión de datos dado que ROS publica imágenes como un tipo de mensaje y OpenCV trata a las imágenes como matrices. Para realizar esta conversión se utiliza una función llamada *cvBridge* la que se encarga de realizar conversiones de mensajes de ROS e imágenes de OpenCV.

La imagen como un tipo de mensaje es publicado constantemente en el tópico *"/bebop/image_raw",* luego se realiza una llamada a la función *callback()* la que se encarga de codificar el mensaje de ROS a un objeto OpenCV y se puede tratar con normalidad como cualquier imagen que es llamada por OpenCV. La resolución de la imagen es de 856x480 pixeles a una tasa de 30 Hz.

Capítulo IV

Navegación autónoma en entornos internos

La navegación autónoma implica varios aspectos, de entre las cuales la primera es la selección del ambiente (entorno) adecuado de acuerdo a los requerimientos y especificaciones del proyecto. Seguido de esto se realiza la delimitación del ambiente o establecimiento de zonas navegables, con ello la construcción del plano del área de trabajo. Luego se tiene la determinación e identificación de los marcadores de referencia para las zonas accesibles, posterior a ello la planificación de camino en donde se revisaran los algoritmos RRT y A*. Otra fase del proyecto de investigación es la estimación de movimientos en base a putos característicos de una imagen y transformaciones geométricas. Como parte final se tiene el modelamiento servo visual respecto a los ejes de interés (Pitch y Yaw) y el diseño del controlador.

Selección, descripción del ambiente

Inicialmente se estableció trabajar al interior de una edificación grande, algunas de las opciones era el Edificio de Postgrados de la Universidad de Las Fuerzas Armadas - ESPE o alguna de las plantas de los bloques destinados como aulas, debido a que estas contaban con amplias áreas en los pasillos como en aulas. Por motivos de fuerza mayor no fue posible realizar el proyecto en estas edificaciones por lo que se optó por una de las plantas de una vivienda (Figura 33).

El ambiente seleccionado es una vivienda, que cuenta con dos pisos (planta baja y alta) de las cuales se seleccionó la planta alta debido a que su diseño favorecía de cierta manera a las condiciones que debía cumplir el ambiente para que el micro UAV pueda sobrevolar el interior de esta.

Figura 33

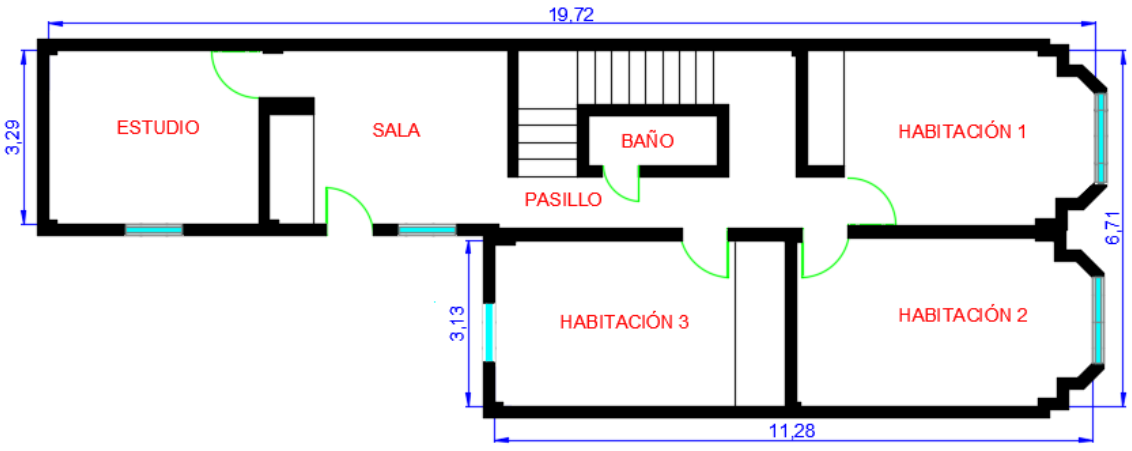
Vivienda seleccionada

Nota. La gráfica muestra la vivienda en donde se ejecutara el proyecto.

La planta alta tiene un área de construcción aproximada de 238 m^2, el mismo que está distribuido en diferentes zonas como se puede observar en la Figura 34, está planta cuenta con tres habitaciones, sala, estudio, baño, pasillos y escaleras que permiten el acceso tanto a la planta baja como a la terraza (ver Figura 35).

Figura 34

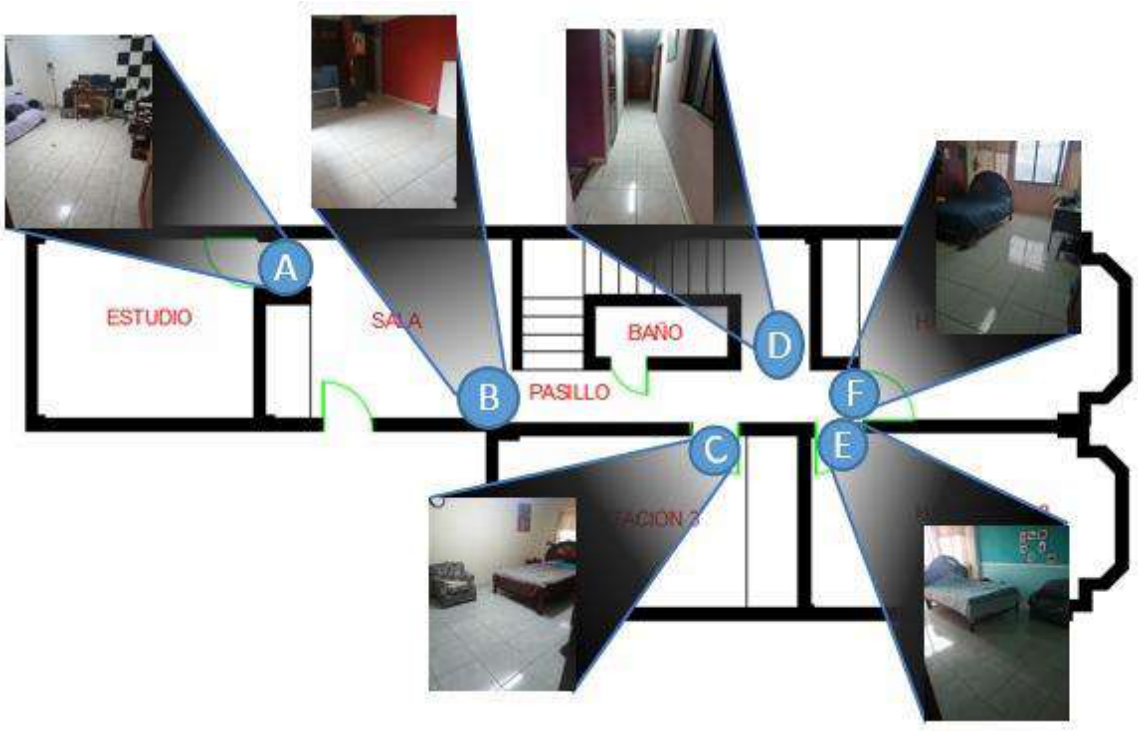
Plano de construcción de la planta alta



Nota. La gráfica muestra el plano general de la planta alta, se puede observar la distribución de las diferentes zonas. Las medidas están expresadas en metros.

Figura 35

Interior de la planta seleccionada



Nota. La gráfica muestra los diferentes ambientes existentes al interior de la planta alta de la vivienda.

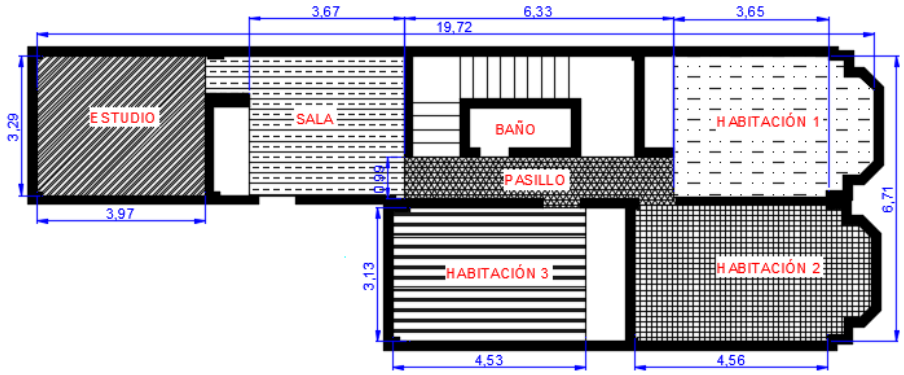
Plano del entorno conocido

Delimitación del entorno

En toda vivienda hay lugares a los que no se puede acceder, ya sea por privacidad o porque el espacio es muy reducido. Para este caso hay ciertas restricciones, el micro UAV Bebop2 solo tendrá acceso a las siguientes zonas: habitación 1, habitación 2, habitación 3, sala, estudio y pasillo (Ver Figura 36). El baño, las escaleras y el pequeño pasillo que conducen al mismo no serán tomados en cuenta por cuestiones de privacidad y por el limitado espacio que existen en los mismos.

Figura 36

Distribución de áreas en la planta alta



Nota. La gráfica muestra las diferentes áreas (con textura) que pueden ser utilizadas para la navegación del micro UAV.

Es muy importante tener en cuenta el espacio donde el dron va ejecutar la navegación, para ello es necesario conocer el área de dichas zonas. En base a las medidas tomadas se procede a determinar el área respectiva, dichas áreas son un aproximado y se pueden observar en la Tabla 8.

Tabla 8

Área de espacios disponibles

Lugar	**Área (aproximada)** $[m^2]$
Habitación 1	12
Habitación 2	14.27
Habitación 3	14.17
Estudio	13
Sala	12
Pasillo	6.33

Nota. En la tabla se detallan las zonas accesibles para el micro UAV Bebop2 así como sus respectivas áreas.

En la mayoría de casos los espacios físicos no siempre están libres (Figura 37). En esta planta se encontraron muebles y espacios ocupados a los que no se tiene acceso, así como una puerta que conectaba con el exterior la cual también se ha considerado como inaccesible debido a que la navegación es netamente al interior de la vivienda.

Figura 37

Zonas inaccesibles

Nota. La gráfica muestra todas las zonas inaccesibles existentes en la planta alta.

Una vez que se han determinado las zonas inaccesibles, procedemos a rellenar estos espacios para que los algoritmos de búsqueda (RRT y A*) no generen rutas por dichas zonas (ver Figura 38). De esta manera se asegura el acceso solo a las zonas autorizadas o accesibles.

Figura 38

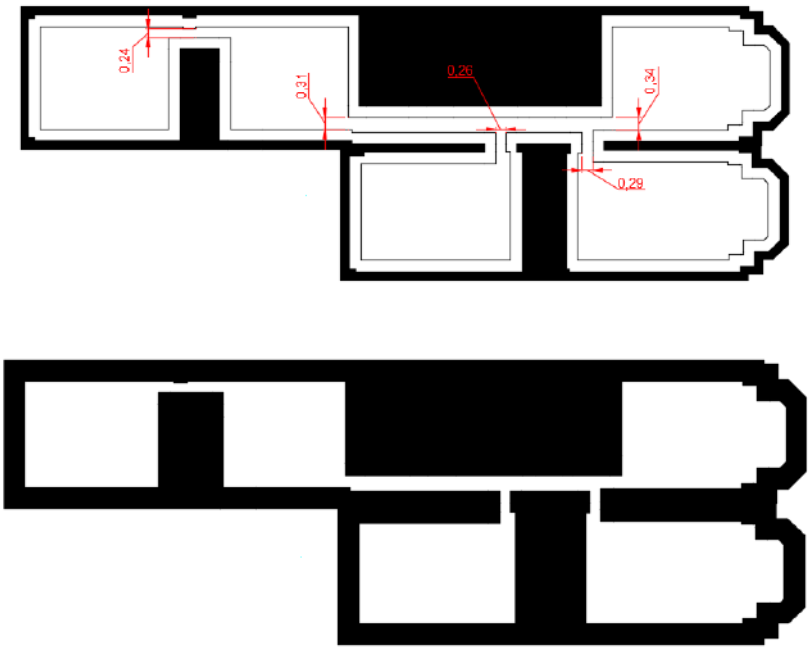
Zonas accessibles

Nota. En la gráfica se observa como se ha rellenado las zonas inaccesibles (espacio en negro): a) identificación de zonas inaccesibles, b) zonas accesibles (espacios en blanco)

Uno de los primeros retos que se presentan en entornos internos es la configuración que estas tienen. Para este caso en particular se tiene accesos reducidos (puertas), tienen un aproximado de 86 cm de ancho lo cual es muy normal en muchas de las viviendas residenciales. Para este proyecto debido a que el micro UAV Bebop2 debe sobrevolar en el interior y pasar por los accesos reducidos tiene una probabilidad de chocar, para evitar de cierta manera el riesgo de choque se ha optado por procesar la imagen y conseguir algo ideal como sería la mostrada en la parte inferior de la Figura 39.

Figura 39

Espacio necesario en los accesos



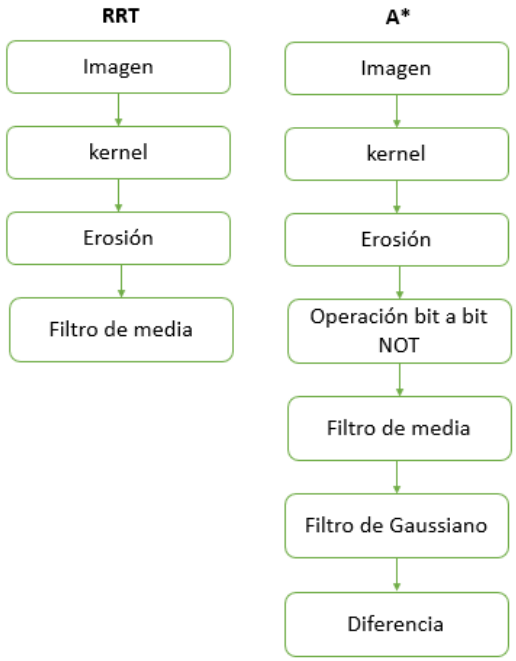
Nota. En la gráfica se observa las zonas a las cuales se tiene acceso con el mínimo espacio para evitar colisiones del micro UAV.

Para conseguir el efecto mencionado en las paredes se optó por las aplicaciones de filtros, operaciones y técnicas básicas de procesamiento de imágenes. Filtros como el de mediana, gaussiano, superposición, sustracción de imágenes, operaciones bit a bit

son algunas de las usadas para procesar el plano y generar un espacio seguro para la navegación del micro UAV. En la Figura 40 se muestra un el proceso de las técnicas utilizadas para usar el algoritmo RRT como para el A*, cada una tiene un tratamiento especial.

Figura 40

*Procesamiento de plano para los algoritmos RRT y A**



Nota. La gráfica muestra las diferentes operaciones y técnicas de procesamiento de imágenes para el plano de navegación del micro UAV Bebop2.

La mayoría de las técnicas son muy conocidas y usadas en el área de procesamiento de imágenes, a continuación se describen algunos de los que se usaron para este proyecto:

- **Kernel:** Es un elemento estructurante de dimensiones pequeñas (matriz). Dependiendo de los valores que este tome, el kernel puede producir una serie de efectos en la imagen procesada (Shapio & Stockman, 2001).

- **Erosión:** Por lo general se lo realiza sobre imágenes binarias. Este recibe como parámetro la imagen original y el núcleo (kernel) que es el encargado de decidir el resultado. Un pixel en la imagen original (ya sea 1 o 0) se considera 1 solo si todos los pixeles debajo del núcleo son 1, caso contrario lo erosiona (se reduce a cero) (OpenCV, 2019).
- **Filtro de media (Median Blur):** Esto realiza un suavizado en una imagen. El elemento central de la imagen se reemplaza por la mediana de todos los pixeles en el área del kernel. Esta operación procesa todos los bordes mientras elimina ruido.
- **Filtro Gaussiano:** Este filtro mezcla ligeramente los colores de los píxeles que estén vecinos el uno al otro en un mapa de bits (imagen), lo que provoca que la imagen pierda algunos detalles minúsculos y, de esta forma, hace que la imagen se vea más suave (aunque menos nítida) respecto a que los bordes presentes en la imagen se ven afectados. Se genera un efecto similar al de una fotografía tomada con una cámara fotográfica desenfocada.

En el caso del plano para el algoritmo RRT, la búsqueda de los valores y parámetros de los filtros aplicados se realiza experimentalmente variando el Kernel y el filtro de media (ver Tabla 9). Por otra parte el plano para el algoritmo A* se trata con los procesos mostrados en el diagrama de la Figura 40, la búsqueda de los valores y parámetros de los filtros aplicados se realiza de igual manera, variando el Kernel, el filtro de media y el filtro Gaussiano como en la Tabla 9. A su vez en la Figura 44 se muestran los resultados de variar y aplicar dichos parámetros.

Tabla 9

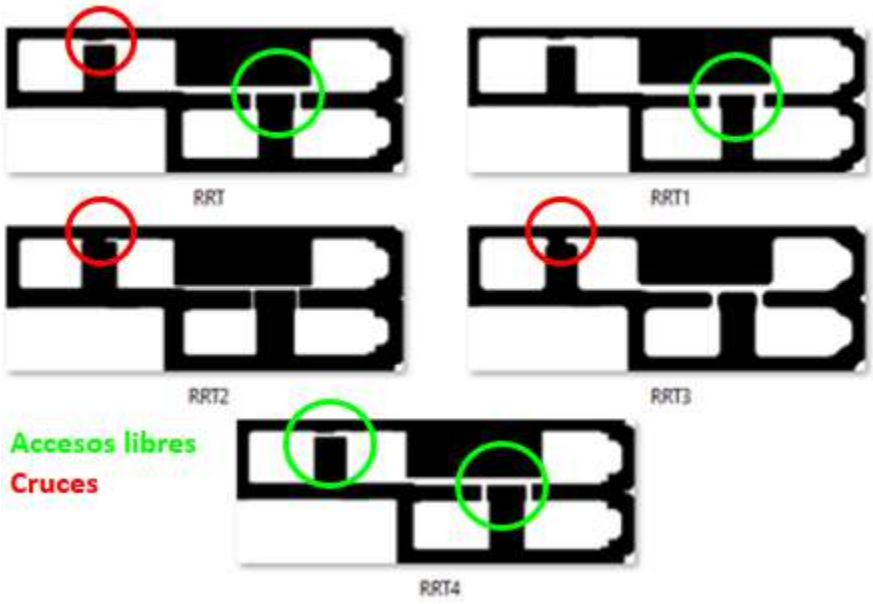
Variación de parámetros en los filtros para plano RTT

Imagen Resultante	Kernel	Median Blur
RRT	[20,20]	5
RRT 1	[15,15]	5
RRT 2	[25,25]	5
RRT 3	[20,20]	15
RRT 4	[20,20]	1

Nota. La tabla muestra las diferentes configuraciones del kernel y el filtro de media para que generan las imágenes respectivas.

Figura 41

Procesamiento de imágenes usados para el buscador RRT



Nota. La gráfica muestra el resultado de aplicar el procesamiento de imágenes modificando alguno de sus parámetros como kernel y filtro de media. Los planos que destacan como mejores son el RRT1 y RRT4.

Se puede notar ciertas diferencias en los planos resultantes de la Figura 41, lo primero es que a medida de incrementar el kernel y el filtro de media existen cruces, eso quiere decir que los accesos están cerrados. Mientras que con un kernel y filtro de media reducido se obtiene buenos resultados sin ningún cierre en los caminos y con una buena estructura en toda la planta.

Por otra parte el plano para el algoritmo A* se trata con los procesos mostrados en el diagrama de la Figura 40, la búsqueda de los valores y parámetros de los filtros aplicados se realiza de igual manera, variando el Kernel, el filtro de media y el filtro Gaussiano (ver Tabla 10). A su vez en la Figura 42 muestra los resultados de variar y aplicar dichos parámetros.

Tabla 10

*Variación de parámetros en los filtros para plano A**

Imagen Resultante	Kernel	MedianBlur	GaussianBlur
A	[15,15]	15	[15,15]
A1	[20,20]	15	[15,15]
A2	[20,20]	5	[15,15]
A3	[20,20]	5	[22,22]
A4	[22,22]	1	[15,15]
A5	[22,22]	1	[15,15]

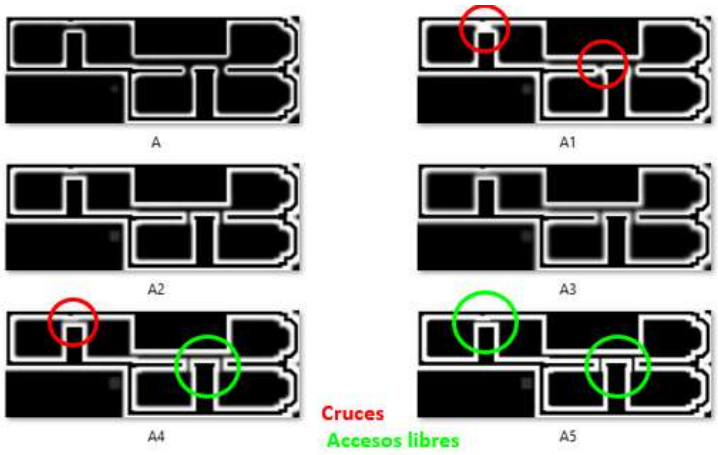
Nota. La tabla muestra las diferentes configuraciones del kernel y el filtro de media para que generan las imágenes respectivas.

Se puede notar ciertas diferencias en los planos resultantes de la Figura 42, como ya se comentó anteriormente el usar parámetros no adecuados para el procesamiento de

imagen trae sus consecuencias, para el caso de la Figura 42 (A1) se usó dos filtros con un valor relativamente alto. No obstante con el uso de más de dos filtros se pueden conseguir resultados favorables como en Figura 42 (A, A2, A3 y A5) donde se logra planos sin cierres de camino.

Figura 42

*Procesamiento de imágenes usados para el buscador A**

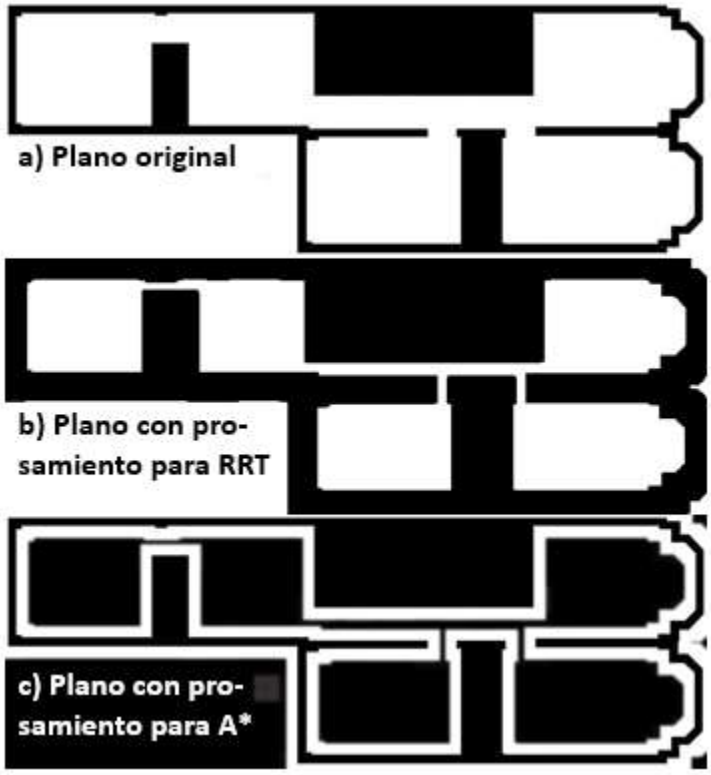


Nota. La gráfica muestra el resultado de aplicar el procesamiento de imágenes modificando alguno de sus parámetros como kernel, así como el valor de los filtros de media y Gaussiano. Los planos que destacan como mejores son: A5 y A2

En la Figura 43 se realiza una comparación de la imagen original sin procesar con las dos maneras de procesar el mismo plano. El plano es el mismo solo que se lo ha tratado de diferente manera. En el b) las áreas en blanco son las zonas accesibles, mientras que líneas negras representan las paredes del interior. Para c) es lo opuesto, para desplazarse de un espacio a otro se lo hace por las líneas negras.

Figura 43

Plano original vs planos procesados



Nota. La gráfica muestra la comparativa entre los planos a los que se aplicaron procesamiento de imágenes con imagen original.

Marcadores de Referencia

ArUco

ArUco es una biblioteca de código abierto para detectar marcadores fiduciales cuadrados en imágenes. Además, si la cámara está calibrada, puede estimar la pose de la cámara con respecto a los marcadores (Muñoz R. , 2018). Es importante mencionar que aquí hay varios tipos de marcadores, cada uno de ellos perteneciente a un diccionario. Cada biblioteca ha propuesto su propio conjunto de marcadores como: ArToolKit +, Chilitags, AprilTags y el diccionario ArUco.

Está basado en OpenCV y permite la detección de varios tipos de etiquetas. Su proceso de detección es mucho más efectivo y rápido que otras librerías, lo que lo hace especialmente adecuado para aplicaciones del análisis de movimientos en tiempo real (Garrido, et al., 2014).

ArUco Markers

Los marcadores tienen bordes negros que facilita su rápida detección en la imagen y la codificación binaria permite su identificación Dependiendo del diccionario, hay marcadores con más o menos bits. Cuantos más bits, más palabras hay en el diccionario y menor es la posibilidad de confusión. Sin embargo, más bits significan que se requiere más resolución para una detección correcta.

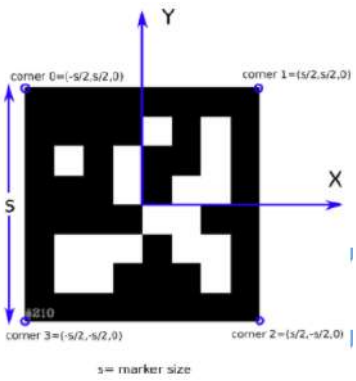
Es utilizado como la aplicación de técnicas de detección y corrección de errores. Los marcadores se pueden utilizar como puntos de referencia en 3D para la estimación de la pose de la cámara. El tamaño del marcador determina el tamaño de la matriz interna (Garrido, et al., 2014). Los marcadores tienen identificación única. Puede ser utilizado para controlar diferentes acciones en un sistema alimentado por cámaras (Mahadeva, 2020).

Debido a la utilidad y facilidad de uso de los marcadores ArUco esta es utilizada para desarrollar diferentes aplicaciones de las cuales se mencionan algunas como: *"Validación de los marcadores ArUco para el análisis de movimientos humanos"* (Jara, et al., 2019*), "Autonomous flying with quadrocopter using fuzzy control and ArUco markers"* (Bacik, et al., 2017), *"Visual localization of mobile robot using artificial markers"* (Babinec, et al., 2014), *"ArUco markers pose estimation in UAV landing aid system"* (Marut,

Wojtowicz, & Falkowski, 2019)*, "Accurate landing of unmanned aerial vehicles using ground pattern recognition"* (Wubben, et al., 2019).

Figura 44

Esquema de un marcador ArcUco



Nota. El gráfico muestra un marcador de ArUco de 6x6. Se puede generar desde su página oficial. Tomada de "*ArUco: An efficient library for detection of planar markers and camera pose*", (Muñoz R. , 2018).

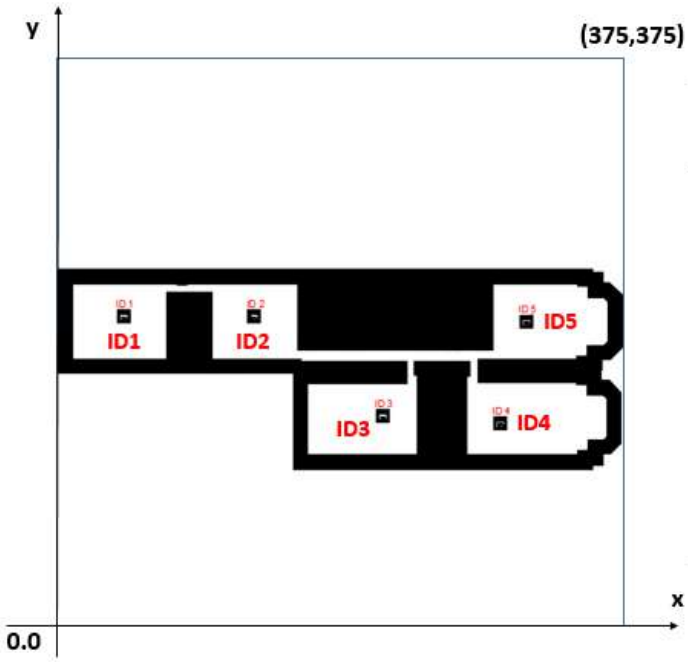
Ubicación y Asignación de ArUco Markers

En el transcurso del desarrollo del proyecto se dio la necesidad de recurrir a elementos que puedan servir de referencia para ubicar al micro UAV de manera efectiva, ya que la navegación al ser netamente visual, la estimación de movimientos no es precisa. El uso de marcadores permitirá no solo conocer la posición del dron, sino que también servirán para identificar a las zonas accesibles de la planta alta. Por otra parte las zonas accesibles no son accesibles en su totalidad, esto porque al tratarse de una planta que pertenece a una vivienda residencial está ocupada (en su mayoría por muebles como: escritorio, cama, sofá, closet, aparatos electrónicos, entre otros) y solo tiene pequeñas áreas libres.

Estos marcadores van a ser ubicados en puntos específicos de las zonas accesibles como se muestra en la Figura 45, en espacios libres que tengan el área suficiente para que el miro UAV ejecute sus acciones de despegue, aterrizaje y navegación sin ningún problema.

Figura 45

Distribución de los marcadores ArUco



Zona	Marker	(x,y)
Estudio	ID1	(46,230)
Sala	ID2	(130,231)
Habitación 3	ID3	(218,164)
Habitación 2	ID4	(291,164)
Habitación 1	ID5	(305,213)

Nota. En la gráfica se muestra la ubicación de los marcadores en las diferentes zonas accesibles mediante los identificadores ID1, ID2, ID3, ID4, ID5.

Se optó por los marcadores ArUco por la facilidad y la variedad de bibliotecas que posee, así como la integración por parte de la librería abierta OpenCV que cuenta con módulos ya establecidos para encontrar este tipo de marcadores en imágenes. Para este proyecto se utilizó los cinco primeros marcadores del diccionario de ArUco 4x4 (desde ID1 hasta 1D5, ver Tabla 11), con una dimensión de 15 x 15 centímetros que el micro UAV pueda detectar fácilmente, esto es muy importante tener en cuenta porque una mala

selección del tamaño hará que el dispositivo de captura de imágenes no la distinga, el tamaño del marcador viene dada en función de la altitud de vuelo y los parámetros de la cámara

Tabla 11

Asignación de ArUco markers a zonas

ID	ID1	ID2	ID3	ID4	ID5
ArUco Markers					
Zona	Estudio	Sala	Habitación 3	Habitación 2	Habitación 1

Nota. En la gráfica se muestra la apariencia de cada uno de los marcadores que se usaron como puntos de referencia en cada zona.

Calibración de la cámara del Parrot Bebop2

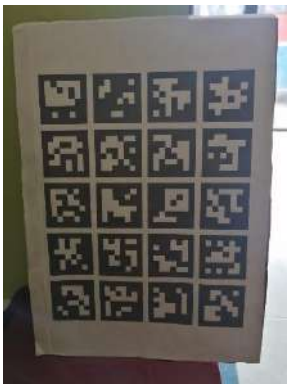
Para este proceso se apoya de la plataforma GitHub, el cual es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador. La plataforma está creada para que los desarrolladores suban el código de sus aplicaciones y herramientas.

Antes de realizar los pasos de calibración es necesario acceder a la cámara del micro UAV mediante la suscripción a los tópicos ya mencionados en el anterior capítulo. Los pasos para calibrar la cámara son los siguientes:

1. Imprimir el tablero de marcadores ArUco.

Figura 46

Tablero para calibración



Nota. La gráfica muestra el tablero impreso y pegado en una superficie plana.

2. Previo a la generación de los datos primero se debe establecer la ruta completa en donde se almacenan las imágenes.
3. Tomar alrededor de 50 imágenes del tablero. Las imágenes deben tomarse desde diferentes ángulos. Utilizar el archivo *"data_generation.py"*

Figura 47

Calibrando la cámara el Parrot Bebop2

Nota. La gráfica muestra la captura de imágenes del tablero con el micro UAV desde diferentes ángulos.

4. Medir la longitud del lado del marcador individual y el espacio entre dos marcadores.
5. Ingresar los datos anteriores (longitud y espaciado) en *"camera_calibration.py"*.
6. Establecer la ruta a las imágenes capturadas anteriormente.
7. Establezca el indicador *"calibrate_camera"* en verdadero, es decir "*calibrate_camera=True*".
8. Ejecutar el archivo *"camera_calibration.py"*

Ejecutando todos los pasos correctamente obtendremos los parámetros de la cámara, las matrices *mtx* de 3x3 y la matriz de distorsión *dst* de 1x5.

$$mtx = \begin{bmatrix} 537.292878 & 0.000000 & 427.331854 \\ 0.000000 & 527.000348 & 240.226888 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix} \quad (15)$$

$$dst = [0.004974 \quad -0.000130 \quad -0.001212 \quad 0.002192 \quad 0.000000] \quad (16)$$

Detección de ArUco Markes con el Bebop2

Para la detección de los marcadores usamos las bibliotecas de OpenCV e implementamos un script en Python. Es necesario definir el diccionario de ArUco markers con el que se está trabajando, para este proyecto especificamos el diccionario de esta manera:

```
aruco_dict = aruco.Dictionary_get(aruco.DICT_4X4_50)
```

Ahora para estimar la Pose de los marcadores usamos la función "aruco" de OpenCV, previamente hay que importarla en el script como *"import cv2.aruco as aruco"*. La pose de los markers viene definida por los vectores de rotación y traslación "*rvec"* y *"tvec" respectivamente.*

```
rvec, tvec ,_ = aruco.estimatePoseSingleMarkers(corners, 0.05,
mtx, dist)
```

Para obtener lo que se ve en la Figura 48 es necesario usar ros para suscribirse al tópico de la cámara y capturar imágenes desde el micro UAV, permitiendo así capturar los marcadores y asignar las zonas correspondientes a cada identificador.

Figura 48

Detección de marcador ArUco

Nota. La gráfica muestra la detección de un marcador ArUco con el Parrot Bebop2, específicamente el ID:5. También se puede ver la estimación de pose del marcador con su respectivo sistema de coordenadas.

Planificador de camino

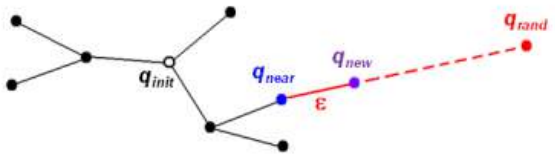
Algoritmo RRT

El algoritmo de Árboles de Exploración Rápida RRT (Rapidly-exploring Random Trees) se trata de una técnica probabilística. RRT es un algoritmo de planificación de caminos que tiene en cuenta los obstáculos y restricciones, el objetico de este algoritmo es dirigir la exploración hacia zonas inexploradas libres de colisiones (Márquez, Maza, & Ollero, 2015) (Gómez, et al., 2017).

Se generan estados aleatorios q_{rand} en el espacio de estado, luego se encuentra el estado del árbol más cercano q_{near} al generado aleatoriamente y se expande el árbol desde q_c hacia q_{rand}, hasta que el estado q_{new} es alcanzado, finalmente el este estado q_{new} se añade al árbol de exploración como se ve en la Figura 49 (Márquez, Maza, & Ollero, 2015) (Wilbert G. & Morales, 2016).

Figura 49

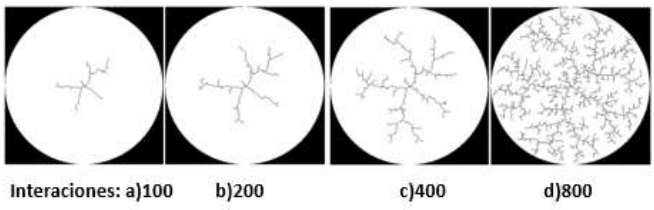
Generación de un nuevo estado en el algoritmo RRT



Nota. La gráfica muestra el inicio del algoritmo RRT, parte de un estado inicial y se van generando ramas (nuevos estados) mientras no colisione. Tomada de "*Comparación de planificadores de caminos basados en muestro para un robot aéreo equipado con brazo manipulador*", (Márquez, Maza, & Ollero, 2015).

Figura 50

Evolución del algoritmo RRT en un entorno circular



Nota. En la gráfica se puede observar cómo según el número de iteraciones del algoritmo este va generando nuevos puntos aleatorios. Tomada de "*Nuevas aportaciones en*

algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos [Tesis Doctoral]", (López D. , 2012).

Para la obtención de q_{new} se tiene en cuenta si hay alguna colisión en dicho desplazamiento, devolviendo "verdadero" o "falso" según sea el movimiento posible o, por el contrario, colisione con algún obstáculo. Si no se ha detectado colisión, se agrega el nuevo punto al árbol distinguiendo entre dos casos.

En el caso de la Figura 50, el espacio libre consiste en un círculo. Significa que desde el punto central no hay ninguna preferencia en cuanto a la dirección de crecimiento. Efectivamente, puede verse que las ramas iniciales del árbol surgen en direcciones aleatorias y crecen sin predominio entre ellas (López D. , 2012).

Algoritmo RRT aplicado a plano de entorno conocido

El pseudocódigo para el algoritmo RRT se describe a continuación (Figura 51). La función *RANDOM STATE* toma una muestra aleatoria del mapa para su posterior análisis. La función *NEAREST NEIGHBOR* encuentra el nodo del árbol más cercano a la muestra. La función *DETECT OBSTACLE* detecta si hay un obstáculo entre ambos. La función *GET PATH* se encarga de computar la trayectoria a partir de las casillas inicial y final y las ramas del árbol que los unen (Muñoz J. , 2019).

Figura 51

Pseudocódigo del algoritmo RRT

Algoritmo RRT

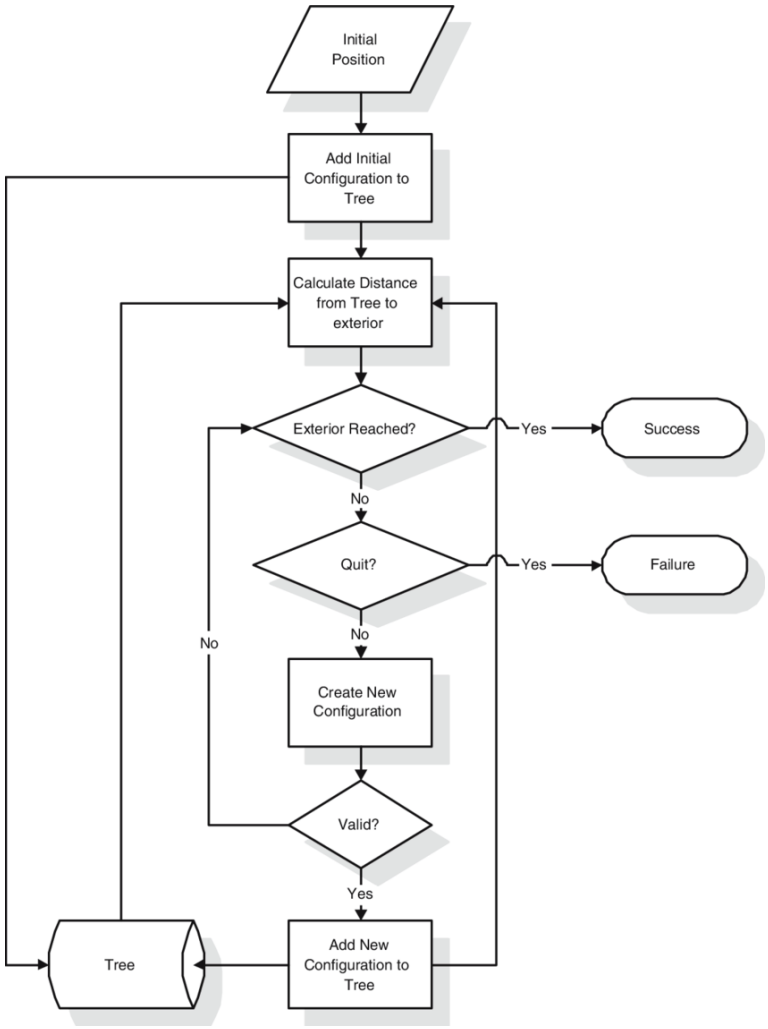
```
1: procedure PATH(x_init, x_goal)
2:     Añadir x_init al árbol
3:     while destino no alcanzado do
4:         x_rand = RANDOM_STATE(Mapa)
5:         x_near = NEAREST_NEIGHBOR(x_rand, árbol)
6:         obstáculo = DETECT_OBSTACLE(x_rand, x_near, Mapa)
7:         if not obstáculo
8:             x_new = x_rand
9:             Añadir x_new al árbol
10:    path = GET_PATH(árbol, x_init, x_goal)
11:    return path
```

Nota. La gráfica muestra el pseudocódigo del algoritmo RRT que se ha implementado. Tomado de (Muñoz J. , 2019).

Para una mejor comprensión se presenta un diagrama de flujo de cómo funciona el algoritmo RRT. Se parte de una posición inicial, esta es la que añade la configuración al árbol, el algoritmo calcula la distancia del árbol hacia el exterior hasta encontrar una nueva configuración válida, la cual es agregada al árbol. Y se reitera nuevamente a partir del nuevo punto.

Figura 52

Diagrama de flujo RRT



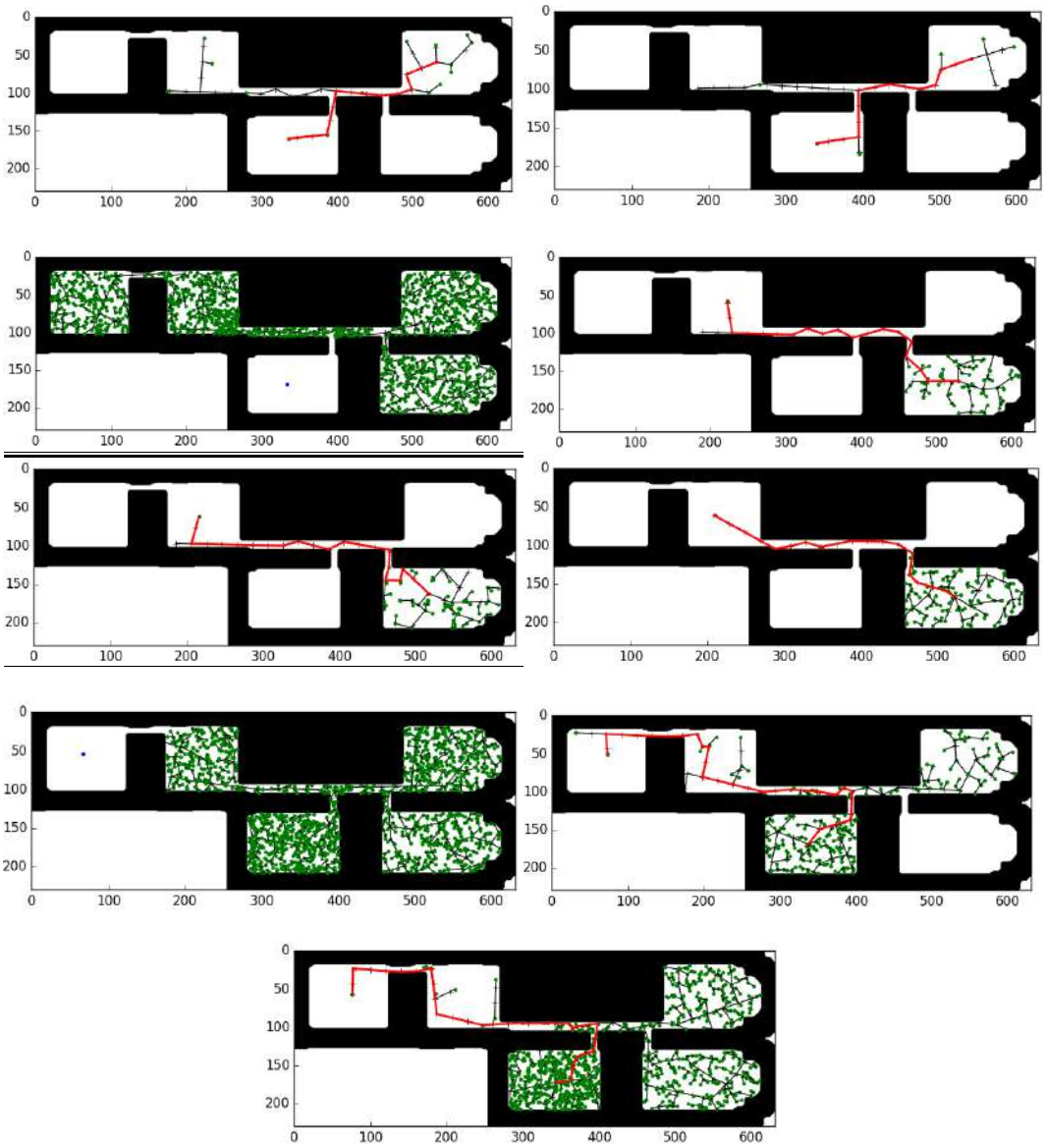
Nota. La gráfica muestra un diagrama de fulo en donde se detalla la lógica de este algoritmo. Tomado de "*Parallel RRT-based path planning for selective disassembly planning*", (Aguinaga, Borro, & Matey, 2008).

Una vez que se ha logrado desarrollar el script para la búsqueda de camino o ruta se procede a realizar diferentes pruebas de funcionamiento como se puede observar en la Figura 53. Se observa que de 9 pruebas realizadas una resulto errónea al no encontrar el punto de destino. También otro punto a tener en cuenta es que este algoritmo al ser

probabilístico genera rutas diferentes para los mismos puntos de inicio y de llegada. Como se observa en las dos primeras imágenes superiores de la Figura 53. No obstante en algunos casos el algoritmo genera rutas en pocas iteraciones.

Figura 53

Pruebas algoritmo RRT



Nota. Resultados de las pruebas de búsqueda de camino con el algoritmo RRT. Las líneas en verde son las ramas aleatorias generadas por el algoritmo.

Algoritmo A*

El algoritmo A* (A-start o A-asterisco), es un algoritmo de búsqueda heurística el cual trata de hallar el camino más corto entre dos puntos (Hart, Nilson, & Raphel, 1986). La función heurística para el algoritmo de búsqueda A* se define como:

$$f(n) = h(n) + g(n) \tag{17}$$

$h(n)$ estima el costo mínimo desde un nodo n hasta el nodo objetivo, esta estimación se hace comúnmente utilizando el concepto de distancia euclidiana y distancia de manhattan, dependiendo de la aplicación donde se valla a implementar el algoritmo A* (Gonzales E. , 2015).

$g(n)$ Estima el costo de desplazarse desde el nodo inicial hasta cualquier otro nodo, tratando que el camino encontrado tenga el menor costo en desplazamiento (Gonzales E. , 2015).

Esta función heurística es aplicada a cada uno de los estados sucesores del estado actual y se escoge como siguiente mejor paso al que contenga el menor valor de $f(n)$. Este algoritmo puede llegar a ser muy valioso dentro de algoritmos de búsqueda debido a que la ruta encontrada se puede considerar óptima en términos de menos distancia recorrida.

$h(n)$ Está definida como la distancia euclidiana entre un par de nodos:

$$h(n) = d_E(n', n_g)\sqrt{\left(n_{gx} - n'_x\right)^2 + \left(n_{gy} - n'_y\right)^2} \tag{18}$$

$g(n)$ Se define como:

$$g(n) = \begin{cases} C(n_i, n') & Para\ i = 1 \\ g(n) + C(n_i, n') & \forall i > 1 \end{cases} \tag{19}$$

Donde n_i con $i = 1$ representa que la búsqueda acaba de empezar, es decir el agente se encuentra en el nodo inicial y $\forall i > 1$ representa cualquier otro nodo que hace parte de la ruta.

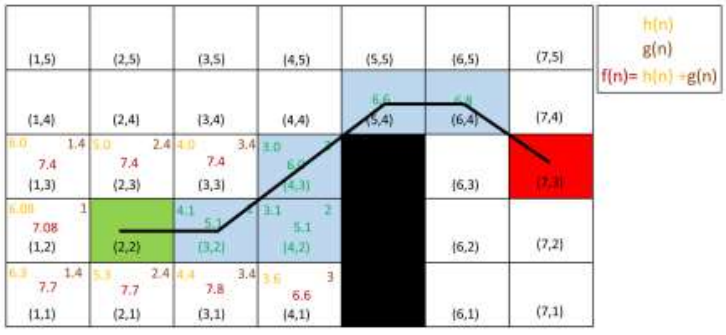
La función $C(n_i, n')$ es la función de costo de desplazar el agente de un nodo actual n a un nodo n' adyacente. Se puede asignar con un valor constante o como el resultado de una función matemática (Gonzales E. , 2015). Para este proyecto en específico considerando el entorno un espacio bidimensional se utilizó la función distancia euclidiana definida para un par de nodos $n_i = (n_x, n_y)$ y $n' = (n'_x, n'_y)$, entonces $C(n_i, n')$ viene definida por:

$$C(n_i, n') = d_E(n_i, n')\sqrt{(n'_x - n_x)^2 + \left(n'_y - n_y\right)^2} \qquad (20)$$

Para una mejor comprensión del funcionamiento de este algoritmo se presenta la Figura 54. Se observa un entorno conformado por una cuadricula, cada casilla corresponde a un nodo y está representado con sus respectivas coordenadas cartesianas, la casilla marcada en verde representa el nodo donde iniciará el agente, las casillas en negro son nodos ocupados y la casilla en rojo es el nodo objetivo.

Figura 54

*Algoritmo A**



Nota. En la gráfica se puede observar la ruta generada por el algoritmo A* para llegar

desde el punto inicial (verde) hasta el punto destino (rojo). Tomado de "*Desarrollo de un algoritmo planificador de rutas con capacidad de implementación en diversas aplicaciones de la robótica móvil",* (Gonzales E. , 2015).

Algoritmo A* aplicado a plano de entorno conocido

Este algoritmo es muy utilizado en el ámbito de la planificación de trayectorias y búsqueda en grafos debido a su eficiencia y precisión. La propiedad más destacada de este algoritmo es que, si existe, es capaz de encontrar la trayectoria de menor coste entre dos puntos.

Para comenzar la planificación se crean dos listas, la lista abierta y la lista cerrada. La lista cerrada maneja las celdas pertenecientes a la trayectoria y la lista abierta las celdas vecinas que se están analizando. La búsqueda parte de la celda inicial, analizando las celdas vecinas y sus costes. A continuación se elige la celda con menor $f(n)$ como sucesora, se añade a la lista cerrada y se realiza el mismo análisis. Si en alguna iteración del algoritmo se analiza una celda ya analizada previamente desde otra celda padre, su $f(n)$ se actualiza. Si esta $f(n)$ es menor que la que tenía previamente, se actualizan las coordenadas de su celda padre y se obtiene una trayectoria de menor coste. La búsqueda de la trayectoria termina cuando la búsqueda alcanza la celda final. La sucesión de celdas desde la celda inicial con $g(n) = 0$ hasta la celda final con $h(n) = 0$, pertenecientes a la lista cerrada, conforman la trayectoria final (Muñoz J. , 2019). El pseudocódigo para el algoritmo A* se muestra en la Figura 55.

Figura 55

*Pseudocódigo algoritmo A**

Algoritmo A*

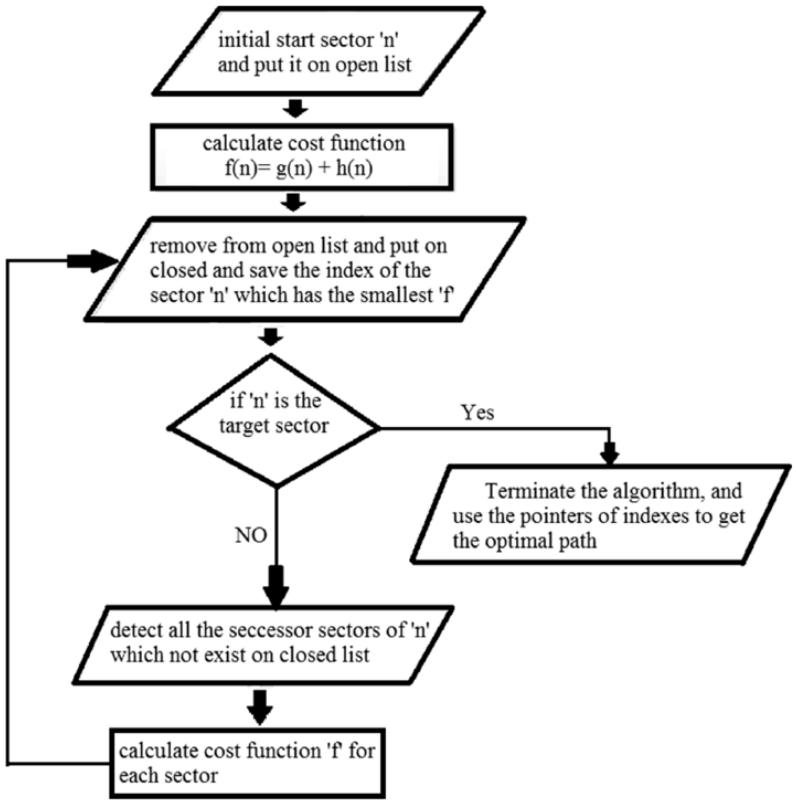
```
1: procedure PATH(x_init, x_goal)
2:     Inicializar Listas
3:     Añadir x_init a la Lista Cerrada
4:     while Lista Abierta no vacía do
5:         NuevaCelda = celda con mínima f en la Lista Abierta
6:         Mover NuevaCelda de la Lista Abierta a la Lista Cerrada
7:         Hijas = celdas adyacentes a NuevaCelda
8:         for Hijas de NuevaCelda que no son obstáculos
9:                 Calcular f(n)
10:                if f(n) < anterior f(n)
11:                        Padre = NuevaCelda
12:                Explorar siguiente vecina
13:    if Lista Abierta vacía
14:            No existe solución
15:    if NuevaCelda = x_goal
16:            path = path entre x_init y x_goal mediante las celdas de la Lista Cerrada
17:    return path
```

Nota: La gráfica muestra el pseudcódigo del algoritmo A*, se diferencia del resto de algoritmos de búsqueda por su encontrar el camino con el menor coste posible. Tomado de: *"Implementación de algoritmos de planificación de trayectorias para robots móviles en entornos complejos",* (Muñoz J. , 2019).

Inicia en un sector "n" y ponerlo en una lista abierta, luego se calcula la función de costo. Hecho esto se remueve de la lista abierta y los ponen en cerrado guardando los índices del sector inicial "n", los cuales son más pequeños que "f". Si el "n" es el sector objetivo termina el algoritmo y usa los punteros de los índices para obtener el camino óptimo. Caso contrario detecta todos los sectores vecinos de "n" los cuales no existen en la lista cerrada, posterior a ello se calcula la función "f" para cada sector.

Figura 56

*Diagrama de flujo A**



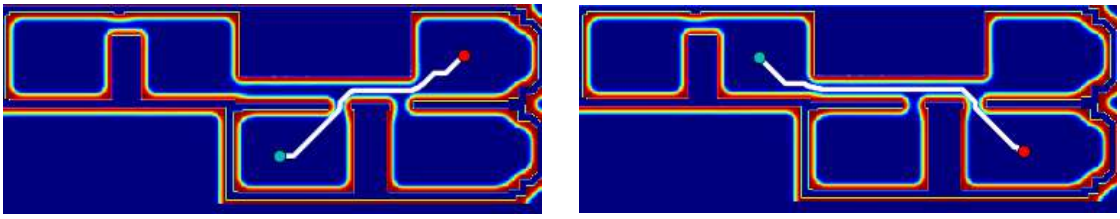
Nota: La gráfica muestra de manera resumida el funcionamiento del algoritmo A* utilizado en la generación de caminos con un coste muy bajo. Tomado de *"Wavefront and A-Star Algorithms for Mobile Robot Path Planning"* (Zidane & Ibrahim, 2018)

Una vez desarrollado el script correspondiente a la búsqueda de camino por el algoritmo A* se realizan pruebas en el plano ya tratado mediante procesamiento de imágenes, en donde se aplicaron filtros y otras operaciones para conseguir un escenario en donde se desempeñe de mejor manera este algoritmo. Como se puede observar en la Figura 57 los puntos rojo y azul son el punto de partida y el punto de llegada respectivamente, las zonas en azul limitadas por las líneas rojas representan las zonas navegables para el micro UAV. Como se ilustra en la figura mencionada efectivamente

genera una ruta mucho más estable y segura que el micro UAV pueda seguir. En las pruebas realizadas, no se tuvo complicaciones como en el caso del algoritmo RRT que no encontraba el punto de llegada.

Figura 57

*Pruebas Algoritmo A**

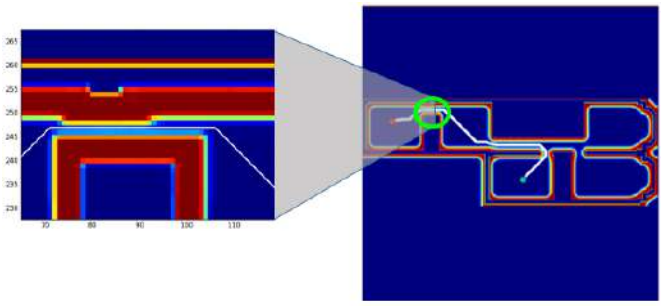


Nota. Resultados de las pruebas de búsqueda de camino con el algoritmo A*. La línea blanca representa la trayectoria generada.

Es necesario observar la ruta de manera más profunda, es por eso que en la Figura 58 se realiza un acercamiento a una de las zonas más complicadas de cualquier generador de rutas, así es, los accesos como puertas o pasillo estrechos, es uno de los retos de este proyecto, es por eso que se ha procesado el plano a tal punto de se garantice que el micro UAV pase por el centro de los accesos o los pasillos. .

Figura 58

Vista cercana de un acceso



Nota. La gráfica muestra un acercamiento de un acceso (puerta), se observa que el algoritmo A* genera la ruta que pasa por el centro del acceso.

Algoritmo RRT vs A*

Para analizar estos dos algoritmos de búsqueda se realizó varias pruebas, en donde se establecieron los mismos puntos de inicio y de destino. La métrica que se usó para comparar estos algoritmos fue los tiempos de procesamiento (ver Tabla 12), es decir el tiempo que se demora en generar una ruta dados los puntos ya mencionados. El algoritmo RRT si bien es cierto, en muchos casos generó la ruta en menor tiempo que el A*, pero la ruta generada no es la más conveniente para la aplicación que se desea en este proyecto. Es por eso que para el desarrollo y ejecución del proyecto se lo hará con el algoritmo de búsqueda A*. Las pruebas realizadas demuestran que este algoritmo funciona correctamente generando rutas estables en un tiempo promedio de 0.38 segundos aproximadamente.

Tabla 12

*Tiempos de procesamiento los algoritmos RRT y A**

Trayectoria	RRT [seg]	A*[seg]
G1	0.35	0.71
G2	14.09	0.59
G3	0.51	0.17
G4	0.07	0.23
G5	0.07	0.27
G6	0.14	0.27
G7	0.07	0.29
G8	0.85	0.59
G9	2.82	0.22
G10	0.47	0.54
Total (Seg)	19.47	3.88
Tiempo Promedio [seg]	**1.947**	**0.388**

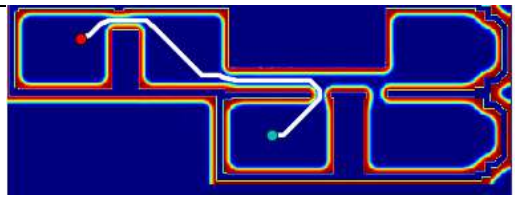
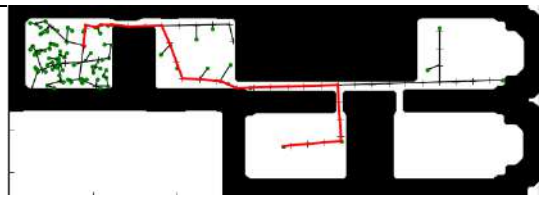
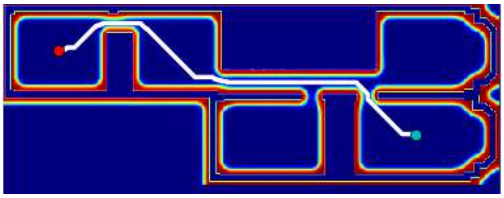
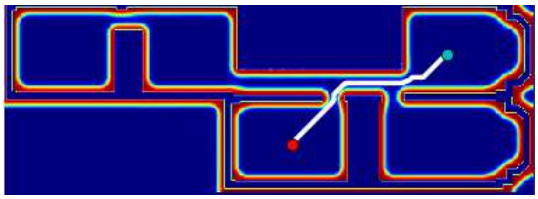
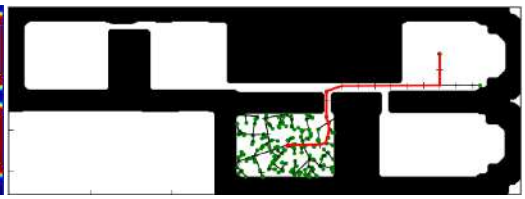
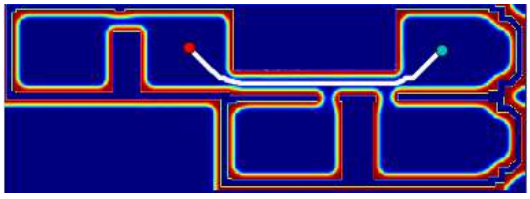
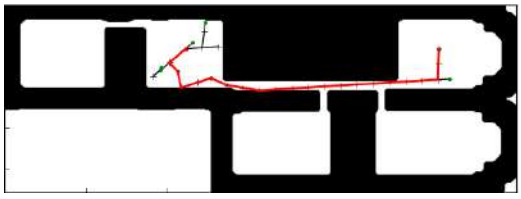
Nota. En la tabla se muestra el tiempo que tarda los algoritmos RRT y A* en generar una ruta entre dos puntos (inicio y destino).

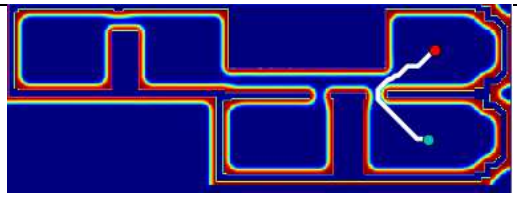
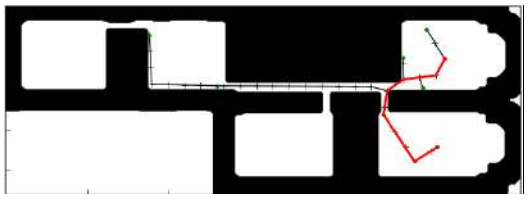
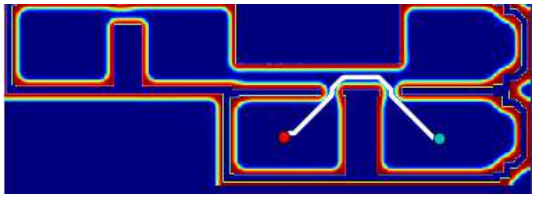
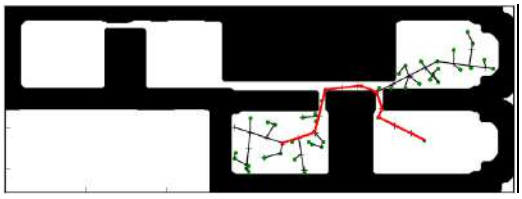
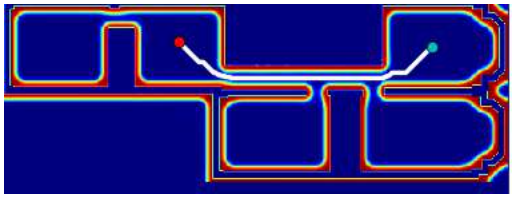
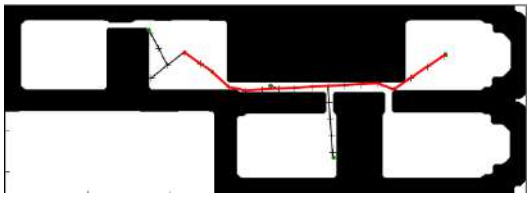
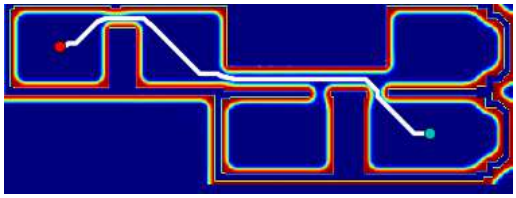
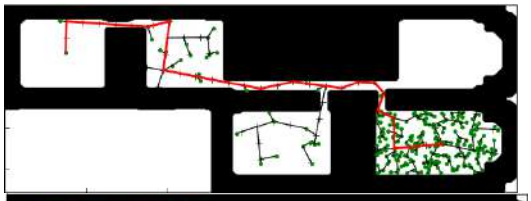
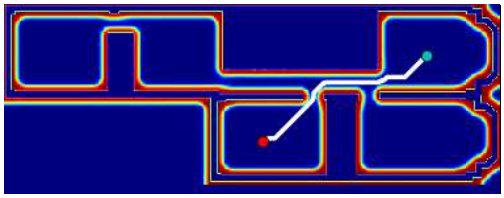
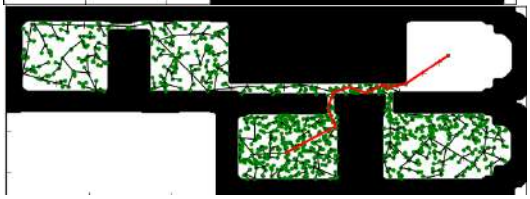
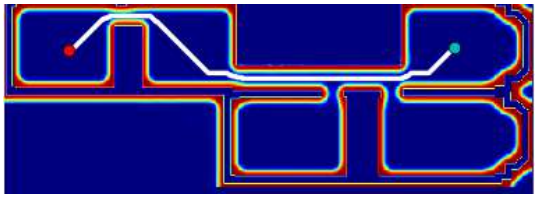
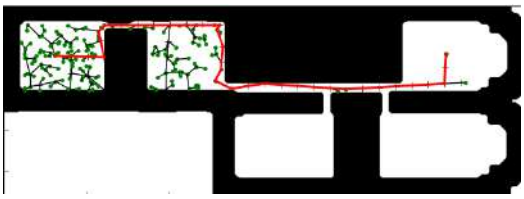
Como antes se mencionó que el algoritmo RRT es probabilístico, eso se puede comprobar en las figuras mostradas en la Tabla 13. En la mayoría de casos las rutas

generadas por RRT tienen trayectorias muy cercadas a las líneas de color negro que viene a ser las paredes, es cierto que se estableció y proceso el plano original para evitar esto, pero aun así es muy arriesgado usar este algoritmo para la el desarrollo de este proyecto. A diferencia de A*, se puede notar donde todas las rutas generadas van por el centro ya sea de las habitaciones o de los pasillos, reduciendo las probabilidades de que el micro UAV choque con las paredes.

Tabla 13

*Comparación algoritmos de búsqueda RRT y A**

Path	RRT	A*
G1		
G2		
G3		
G4		

Path	RRT	A*
G5		
G6		
G7		
G8		
G9		
G10		

Nota. En la tabla se muestra las rutas generadas por cada algoritmo. El A* genera una ruta más adecuada para el micro UAV.

Extracción de puntos característicos

Una parte importante del proyecto consiste en la interpretación y reconocimiento de una imagen, es por ello que para esta fase se realiza una comparativa de algunos algoritmos de detección y descripción, posterior a ello se elige uno de los algoritmos que mejor se adapten al proyecto y finalmente se procede a lo que se conoce como correspondencia de puntos característicos o matching.

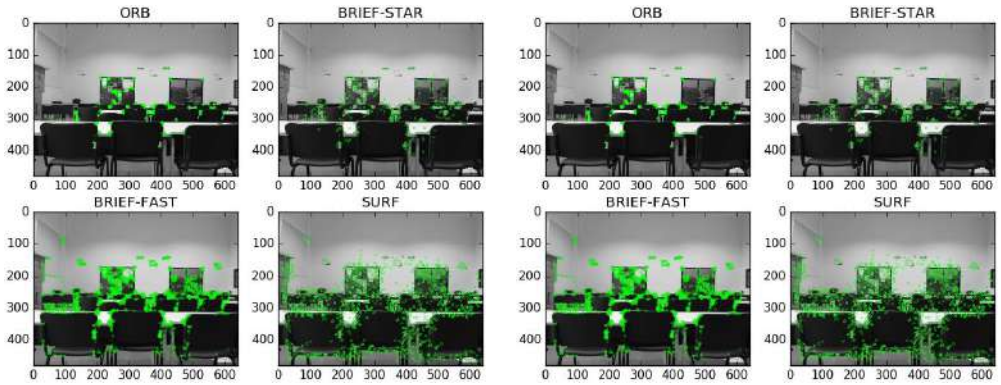
Detectores y descriptores

Algoritmo SURF, FAST, ORB

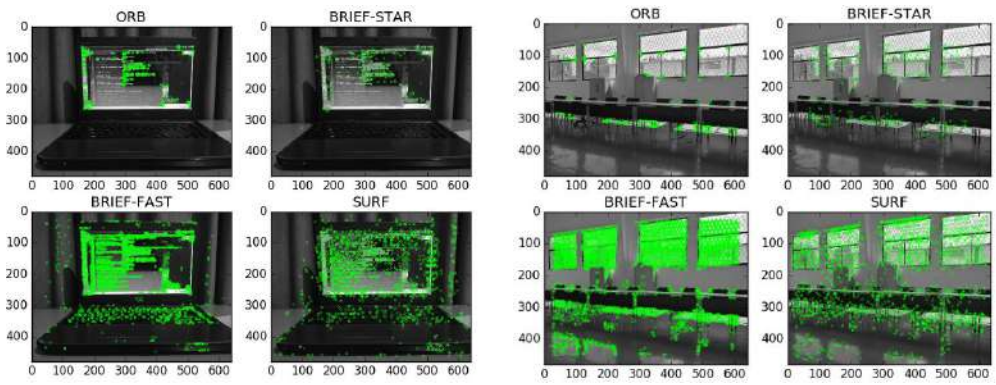
Para la elección del algoritmo que servirá para detectar y extraer los puntos característicos se realiza una prueba de los algoritmos ORB, BRIEF-START, BRIEF-FAST Y SURF como se muestra en la Figura 59. Se ha seleccionado estos algoritmos para realizar estas pruebas según la revisión de literatura y se ha establecido 4 imágenes que varían en detalles, texturas, iluminación, las mismas que se usan para la aplicación de dichos algoritmos.

Figura 59

Algoritmos ORB, BRIEF-STAR, BIEF-FAST y SURF



FT1 **FT2**



FT3 FT4

Nota. En la gráfica se observa las diferentes pruebas realizadas, se pone a prueba cuatro algoritmos de detección y extracción de características aplicadas a diferentes imágenes y exposiciones de luz.

Como resultado de las pruebas realizadas se obtuvieron los siguientes datos: Los Keypoints (puntos característicos) y el tiempo de procesamiento (Tabla 14), estas son las métricas que se usaron para realizar una comparativa y posteriormente la selección del algoritmo con el que se trabajará. Visualmente podemos decir que el mejor algoritmo es el que presenta la mayor cantidad de puntos característicos, en ese caso sería BRIEF FAST. En cuanto al tiempo de procesamiento los tres algoritmos ORB, BRIEF-STAR y BRIEF-FAST tienen un procesamiento muy rápido. Se debe tener presente que en el caso del ORB, los puntos característicos lo definimos previamente, es por ello que siempre tendremos 500 keypoints con un tiempo de procesamiento similar al de algoritmos modificados presenta un buen rendimiento. Otro punto a tener en cuenta es la etapa de correspondencia de puntos, es por ello que no es muy recomendable tener cientos o miles de puntos que describan lo que con una decena de puntos se pueden lograr. Con lo mencionado y las pruebas realizadas y resultados mostrados se determina ORB como el algoritmo de extracción de puntos característicos del sistema que se desarrolla.

Tabla 14

Comparación de algoritmos para detección y extracción de características

		Keypoints	Time Process
FT1	Orb	500	0.01181094
	Brief - Star	269	0.01436846
	Brief – Fast	1507	0.01534797
	Surf	1266	0.10916090
FT2	Orb	500	0.01617209
	Brief - Star	241	0.00951505
	Brief – Fast	1481	0.01555103
	Surf	1066	0.09937382
FT3	Orb	500	0.01031702
	Brief - Star	200	0.01078979
	Brief – Fast	2641	0.01325598
	Surf	1647	0.13814783
FT4	Orb	500	0.01356999
	Brief - Star	377	0.01904898
	Brief – Fast	5500	0.01851296
	Surf	2452	0.17927909
Promedio	**Orb**	**500**	**0.01296751**
	Brief - Star	**271**	**0.01343057**
	Brief – Fast	**2782**	**0.01566698**
	Surf	**1607**	**0.13149041**

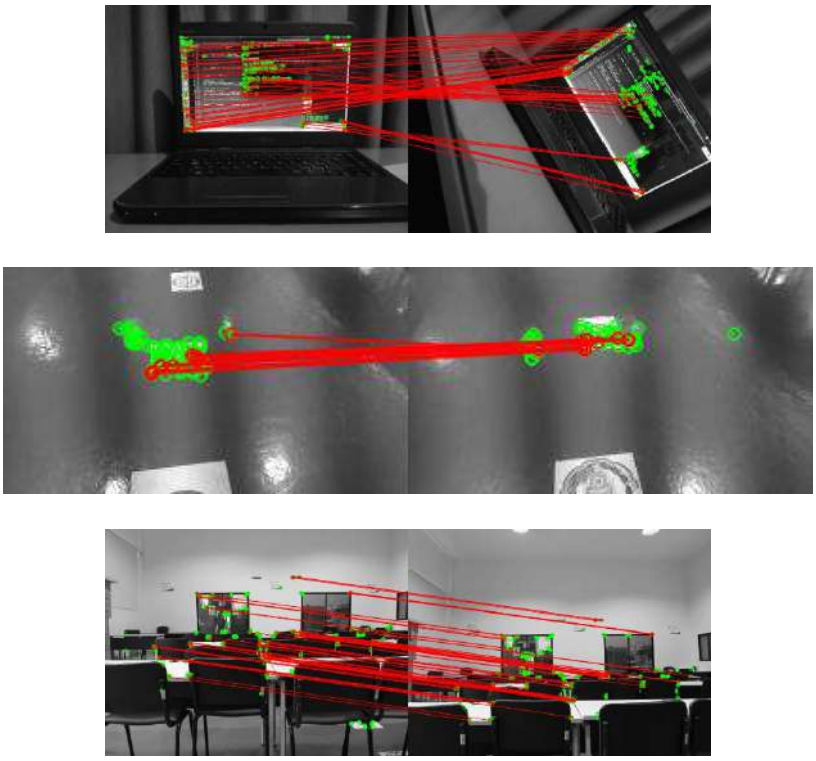
Nota. En la tabla se puede apreciar las métricas que se usaron para determinar el algoritmo a utilizar en el proyecto. OBR destaca entre los mejores algoritmos de extracción de puntos característicos.

Correspondencia de puntos

Una vez seleccionado el algoritmo extractor de características lo segundo es realizar la correspondencia de puntos entre dos imágenes, las mismas que deben ser similares, como se puede observar en la Figura 60.

Figura 60

Correspondencia de kerypoints entre dos imágenes



Nota. La gráfica muestra la correspondencia (líneas rojas) de keypoints (círculos verdes) entre dos imágenes en escenarios diferentes.

Al realizar este proceso se debe tener en cuenta que no siempre la correspondencia de puntos es la correcta, a esto se lo conoce como falsos positivos. Es decir, cuando un punto característico de una imagen es correspondida a otro punto totalmente diferente. Estas correspondencias falsas son las que generan problemas en lo que posteriormente se explicará, la etapa de estimación de movimientos mediante transformaciones geométricas de una imagen.

Control servo visual

La función principal del servo visual control es controlar el quadrotor utilizando la información obtenida en el espacio de la imagen. Para este proyecto se desarrollará un control servo visual basado en imágenes, las mismas que serán captadas por la cámara del micro UAV, para el modelamiento servo visual se toma como referencia los trabajos realizados por parte de (Aguilar & Angulo, 2014) y (Aguilar, et al., 2017).

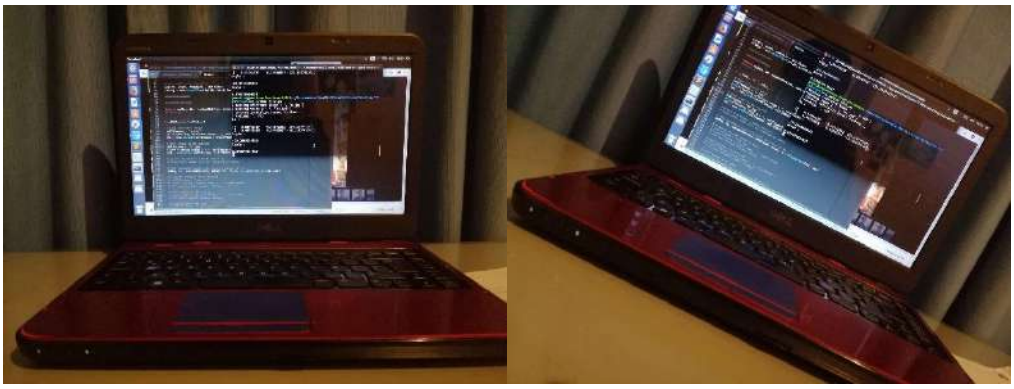
Estimación de movimiento del UAV

Transformaciones geométricas

La principal utilidad o enfoque de las conocidas como transformaciones geométricas son la de estimar las deformaciones en las imágenes tomadas por una cámara. Otra de las aplicaciones es la de estimar como una imagen se movió en dos instantes de tiempo, en otras palabras si se aplicó alguna rotación o traslación a la imagen como muestra la Figura 61.

Figura 61

Imágenes tomadas en diferentes instantes de tiempo



Nota. En la gráfica se puede observar dos imágenes tomadas en diferentes instantes de tiempo y desde diferentes ángulos. Frame 1 (izquierda) y frame2 (derecha).

Para determinar la matriz de transformación primero se realiza la extracción de puntos de interés (keypoints), luego la correspndencia de puntos entre las dos imágenes y finalmente mediante las librerías de OpenCV seleccionamos que tipo el tipo de transformación de acuerdo a nuestro interés. Para este caso como solo se desea conocer si la imagen tuvo una rotación, traslación o las dos escogemos la transformación de similitud (Similitary Transformation). Matemáticamente la matriz de similitud viene dada por (12) en capítulos anteriores. Para el ejemplo de la Figura 61 la matriz de transformación de similitud viene dada por (21).

$$H_s = \begin{bmatrix} 0.91 & -0.46 & 170.94 \\ 0.46 & 0.91 & -144.80 \\ 0 & 0 & 1 \end{bmatrix} \tag{21}$$

Donde $T_x = 170.94$ y $T_y = -144.80$ y el ángulo de rotación es:

$$\theta = arcTan\left(\frac{T_y}{Tx}\right) \tag{22}$$

La transformación de una imagen implica la creación de una nueva a partir de dos imágenes (frame 1 y frame 2, ver Figura 61). El área oscura representa pixeles negros (Figura 62) y se interpreta como la transformación para pasar del frame 1 al frame 2.

Figura 62

Transformación de la imagen

Nota. En la gráfica se puede observar dos imágenes tomadas en diferentes instantes de tiempo y desde diferentes ángulos. Frame 1 (izquierda) y frame2 (derecha).

Para estimar el movimiento basta con obtener la matriz de trasformación, determinar la rotación y traslación, finalmente realizar una aplicación con los datos obtenidos. Para este caso en particular es necesario realizar una transformación de pixeles a metros, esto se lo hace mediante una relación. De esta manera se puede estimar el movimiento del UAV a través de fotogramas tomados en instantes de tiempo.

Modelamiento servo visual

Para el diseño de cualquier tipo de controlador es importante conocer la planta y su modelo que describe su comportamiento, para este proyecto la propuesta de modelamiento es servo visual utilizando visión por computador, el sistema de visión es la cámara integrada en el Bebop2 la cual nos provee los fotogramas que en conjunto con los procesos descritos anteriormente se puede estimar el movimiento del micro UAV. Se opta por este tipo de modelamiento debido a que en varios proyectos realzados como (Grijalva, 2019), (Salcedo, 2018), (Chasillacta, 2020), (Chauca, 2020) obtienen buenos resultados utilizando este método. El modelamiento servo visual puede llegar a ser mejor que un modelo matemático basado en estimación de movimientos basados en Unidad de Medida Inercial (IMU) como (Aguilar, Casaliglla, & Pólit, 2017), (Khamseh, Lotfi, & Taghirad, 2019) o (Yang, Lee, Jeon, & Lee, 2017), esto porque los tópicos del UAV Bebop2 tienen una tasa de refresco bajo, aproximadamente de 5 Hz lo que dificulta el cálculo del modelo matemático estimado para el movimiento del micro vehículo aéreo no tripulado (Salcedo, 2018) (Grijalva, 2019).

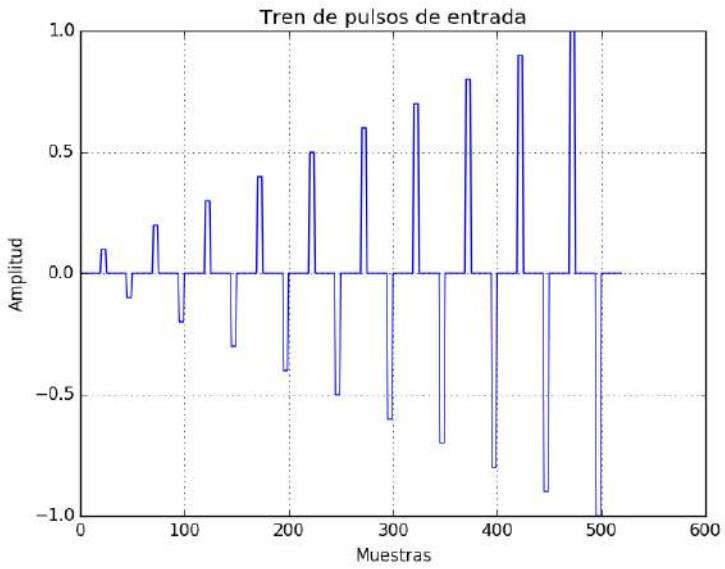
Solo se realizará el modelamiento respecto a dos ejes del Bebop2, en los ejes Pitch (eje X) y Yaw (eje Z). Sobre el eje Pitch porque permite el avance o retroceso en línea recta, mientras que sobre el eje Yaw para realizar giros sobre su propio eje ya sea a la derecha o izquierda. Es importante conocer el comportamiento sobre cada eje al aplicar cierta velocidad, para eso se define una entrada tipo tren de pulsos con amplitudes

que están entre [-1.0 y 1.0] como se muestra en la Figura 63, este rango de velocidades son las que permite el micro UAV y son de tipo flotante.

Al aplicar el tren de pulsos al tópico de control de velocidad, se tendrá un efecto. Los valores positivo como negativo son interpretados por el miro UAV y hacen que este se desplace de adelante hacia atrás, o gire de izquierda a derecha dependiendo el tópico utilizado.

Figura 63

Tren de pulsos de entrada



Nota. La gráfica indica el tren de pulsos de entrada para el modelamiento servo visual del micro UAV Bebop Parrot 2. La amplitud tiene un rango de [-1.0 a 1.0].

Paralelamente al aplicar el tren de pulsos se captura fotogramas consecutivos, esto con el propósito de aplicar los algoritmos de extracción de características, correspondencia de puntos característicos y finalmente mediante las transformaciones geométricas estimar los cambios de traslación (para el eje Pitch) o rotación (Para el eje

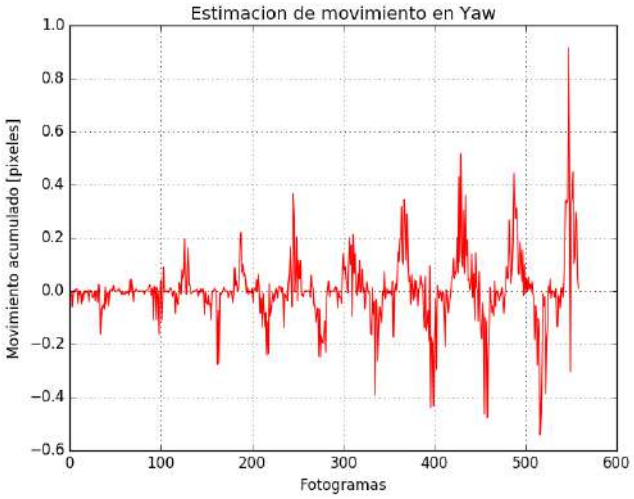
Yaw) entre cada fotograma permitiendo estimar el desplazamiento realizado por el miro UAV.

Modelamiento en el eje Yaw

Para este modelamiento es necesario del tópico “/bebop/cmd_vel” aplicada al eje Yaw, se envía los valores del tren de pulsos de manera secuencial generando movimiento rotacional del drone sobre su propio eje de izquierda a derecha. Se capturan los fotogramas y se realiza el proceso de estimación de movimiento y se obtiene lo mostrado en la Figura 64.

Figura 64

Estimación de movimientos en Yaw



Nota. La gráfica muestra el comportamiento entre cada fotograma tomado a través de la cámara del Bebop2. El eje “y” representa el movimiento acumulado entre pixeles mientras que el eje “x” el número de fotogramas que fueron capturadas en cada instante de tiempo.

Para la identificación de la planta se utilizó la herramienta ident de Matlab que es usada para la obtención de modelos matemáticos de un sistema a lazo abierto. Los

parámetros que necesita son: La señal de entrada, la señal de salida y el tiempo de muestreo.

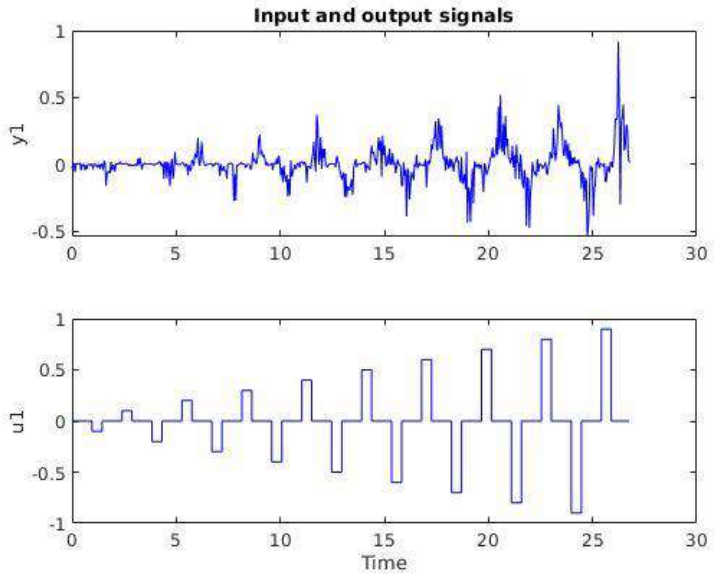
El tiempo de muestreo se determina de la siguiente manera:

$$Ts = \frac{Tiempo\ total\ transcurrido}{Número\ total\ de\ frames\ procesados} \tag{23}$$

El tiempo de muestreo determinado es de 48 milisegundos y el número total de frames total es de 559 dando un total de 27 segundos aproximadamente como se muestra en la Figura 65. Esto quiere decir que el sistema tarda 27 segundos en capturar y procesar todos los frames.

Figura 65

Señales de entrada y salida para Yaw



Nota. La gráfica muestra las señales de entrada y salida, u1 y y1 respectivamente. Estas son generadas mediante la herramienta ident de Matlab antes de estimar el modelo matemático que mejor se adapte a la señal de salida.

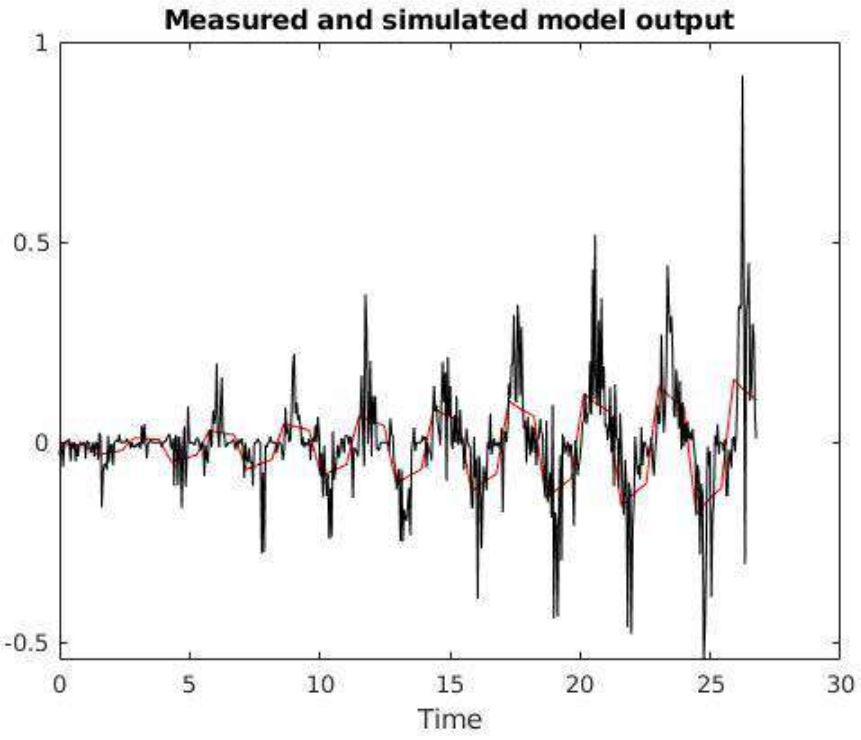
Con los parámetros ingresados basta con seleccionar el tipo de identificación que se requiera, esto da como resultado una función de transferencia de primer orden que de manera general viene dada por la siguiente expresión:

$$G_1(s) = \frac{K_p}{1 + T_p s} \tag{24}$$

Donde $K_p = 1.3998$ y $T_p = 2.1739$. Como resultado se tiene la Figura 66 la gráfica del modelo estimado.

Figura 66

Identificación de la planta en Yaw



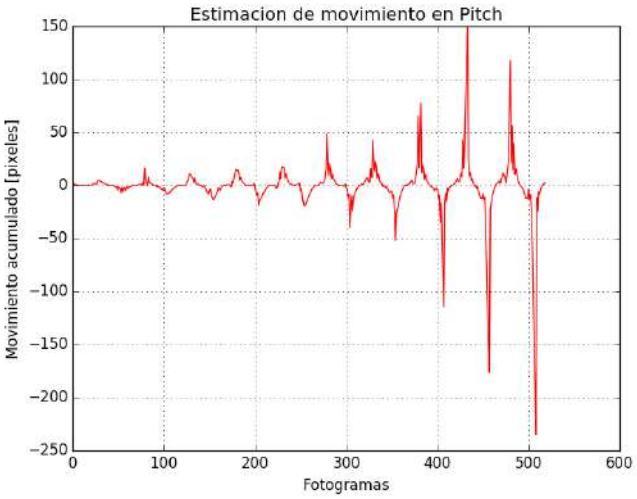
Nota. En la gráfica se observa la señal de salida (línea negra) obtenida mediante la estimación de movimientos en Yaw, así como la señal estimada mediante ident de Matlab.

Modelamiento en el eje Pitch

La metodología es la misma que se dio para el modelamiento en el eje Yaw, al aplicar el tren de pulsos de la Figura 63 se obtiene la estimación de movimientos para Pitch, donde se nota el cambio en el eje “y” de la Figura 67. En este caso el movimiento es lineal por lo que se desplaza una mayor cantidad de pixeles ya sea adelante o atrás. A medida que aumenta la amplitud de la señal de entrada el dron gana más velocidad generando sobre picos que representan cambios bruscos entre frames, lo que da lugar a malas interpretaciones por parte de los algoritmos utilizados especialmente en la etapa de matching generando falsos positivos, por consecuencia una mala estimación de la matriz de transformación y movimiento.

Figura 67

Estimación de movimientos en Pitch



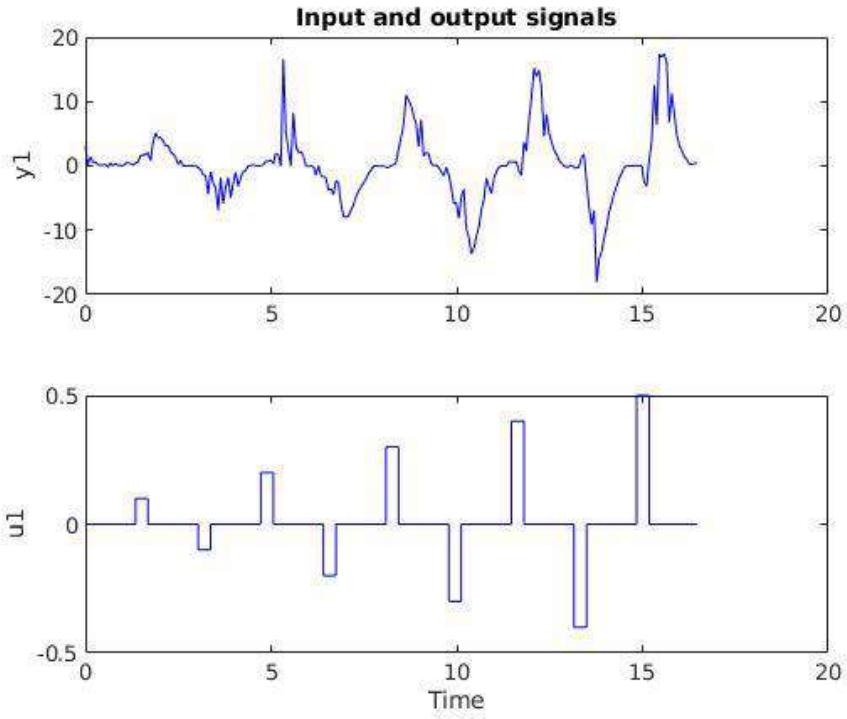
Nota. La gráfica muestra el comportamiento entre cada fotograma tomado a través de la cámara del Bebop2. El eje “y” representa el movimiento acumulado entre pixeles mientras que el eje “x” el número de fotogramas que fueron capturadas en cada instante

de tiempo. Los picos a partir de los 350 fotogramas se generan por el movimiento brusco del Bebop2 al estar a altas velocidades.

Para este proyecto cabe recalcar que no se trabajará a velocidades elevadas porque la navegación es en el interior de una vivienda. Por esta razón se omite los datos generados a partir de las amplitudes mayores a 0.5 y -0.5 en el tren de pulsos.

Figura 68

Señales de entrada y salida para Pitch



Nota. La gráfica muestra las señales de entrada y salida, u1 y y1 respectivamente. Estas son generadas mediante la herramienta ident de Matlab antes de estimar el modelo matemático que mejor se adapte a la señal de salida. Se ha recortado la señal debido a que para este proyecto se manejarán bajas velocidades.

El tiempo de muestreo determinado es de 67 milisegundos y el número total de frames total es de 245 dando un total de 16.7 segundos aproximadamente como se muestra en la Figura 68. Esto quiere decir que el sistema tarda 16.7 segundos en capturar y procesar todos los frames.

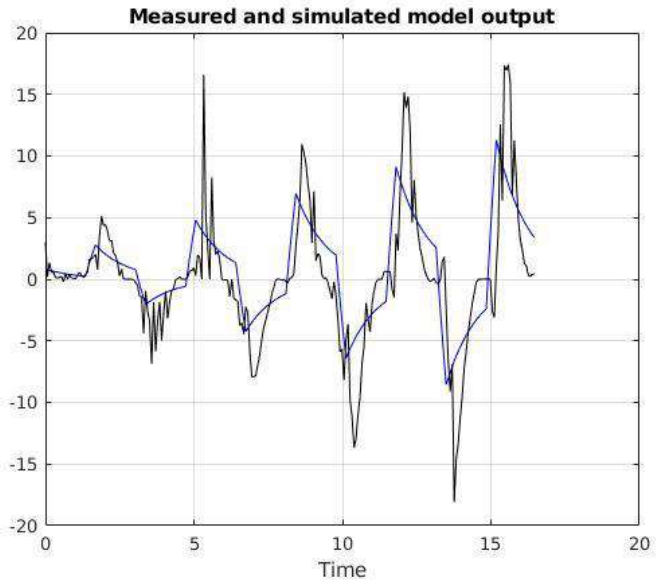
Con los parámetros ingresados se tiene como resultado una función de transferencia de primer orden que de manera general viene dada por la siguiente expresión:

$$G_2(s) = \frac{K_p}{1 + T_p s} \tag{25}$$

Donde $K_p = 95.959$ y $T_p = 1.063$. Como resultado se tiene la Figura 69 la gráfica del modelo estimado.

Figura 69

Identificación de la planta en Pitch



Nota. En la gráfica se observa la señal de salida (línea negra) obtenida mediante la estimación de movimientos en Yaw, así como la señal estimada mediante ident de Matlab.

Diseño de Control PID para seguimiento de camino

Para diseñar este controlador es necesario conocer la función de trasferencia de la planta que viene a ser el micro UAV Bebop2, la identificación de la planta se lo realizó mediante modelado servo visual utilizando visión por computador aplicado a frames consecutivos tomados con la cámara del drone. Se necesitan de dos controladores independientes, uno para el eje Z (Yaw) y otro para el eje X (Pitch). Matlab proporciona herramientas potentes para el diseño y sintonización de controladores como el pidTuner.

Controlador PID para Yaw

El controlador para este caso debe tener ciertas características de rendimiento, debe ser capaz de garantizar un error en estado estable cercano a cero y compensar el movimiento requerido para posicionar al drone en posición de aterrizaje asó como en trayectorias donde el dron debe girar sobre su propio eje hasta alcanzar el ángulo deseado.

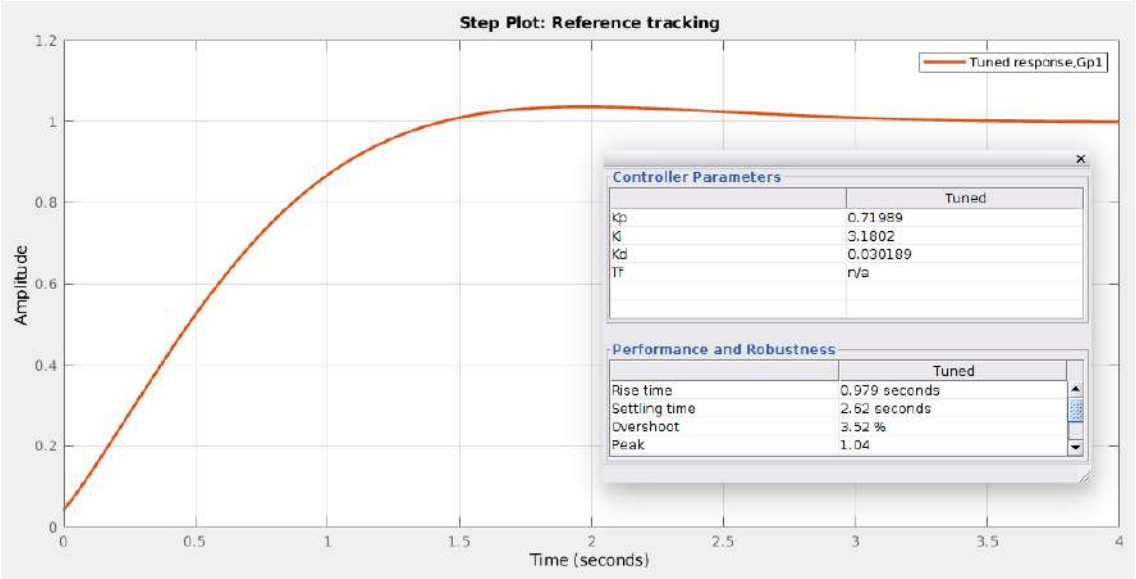
- Overshoot: $M_p < 5\%$
- Tiempo de asentamiento: $t_s < 3\ seg$

Los parámetros resultantes después de haber sintonizado el controlador son las constantes K_p, K_i y K_d mostrados en la Figura 70. Los valores de las constantes correspondientes son:

- $K_p = 0.71989$
- $K_i = 3.1802$
- $K_d = 0.030189$

Figura 70

Respuesta al escalón unitario del controlador en Yaw



Nota. La gráfica muestra los resultados del controlador diseñado para el eje Yaw. Se puede observar que se cumple con las condiciones de desempeño propuestas.

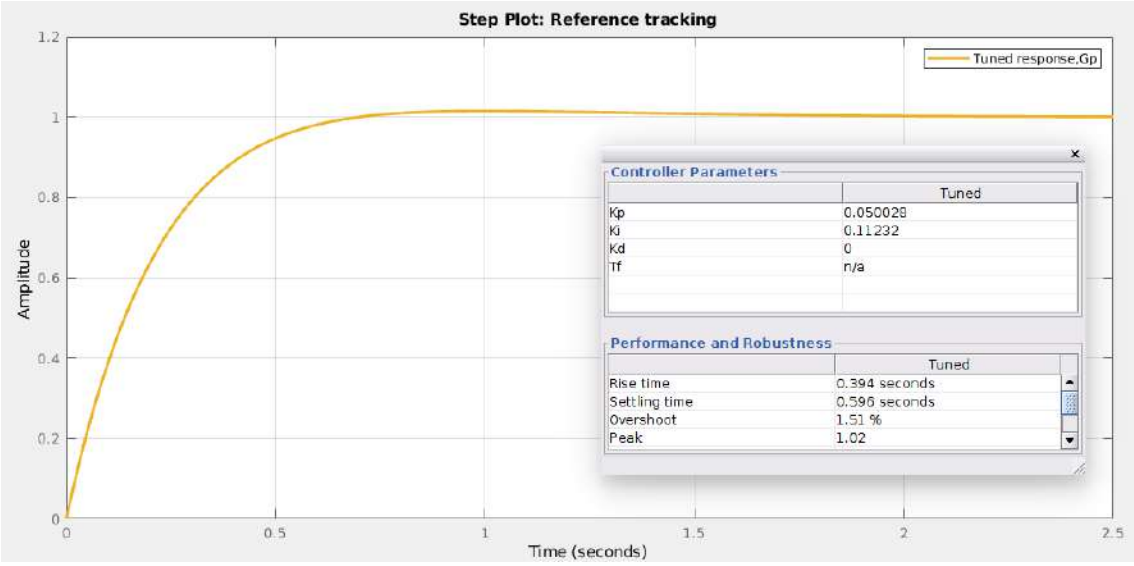
Controlador para Pitch

Siguiendo los mismos pasos de diseño del controlador para el eje Yaw se lo hace para el eje Pitch. El controlador para este caso debe tener ciertas características de rendimiento, a diferencia del controlador para Yaw el controlador para el eje Pitch debe tener una respuesta rápida y sobre picos mínimos. Este controlador debe compensar el movimiento requerido para posicionar al drone en cada instante de tiempo de acuerdo a la ruta definida por el algoritmo de búsqueda A*. Utilizando pidTuner de Matlab y especificando los parámetros de rendimiento se obtiene las constantes del controlador PID.

- Overshoot: $M_p < 2\%$
- Tiempo de asentamiento: $t_s < 1\ seg$

Figura 71

Respuesta al escalón unitario del controlador en Pitch



Nota. La gráfica muestra los resultados del controlador diseñado para el eje Pitch. Se puede observar que se cumple con las condiciones de desempeño propuestas.

Los parámetros resultantes después de haber sintonizado el controlador son las constantes K_p, K_i y K_d mostrados en la Figura 71. Los valores de las constantes correspondientes son:

- $K_p = 0.050028$
- $K_i = 0.11232$
- $K_d = 0.0$

Capítulo V

Conclusiones y recomendaciones

Conclusiones

Se estableció la comunicación entre la estación terrestre y el micro UAV Bebop Parrot2, así como el control de movimientos básicos: despegue, aterrizaje, desplazamientos lineales en x (PITCH), y (ROLL), así como desplazamientos angulares sobre el eje z (YAW) mediante los tópicos proporcionados por el controlador "*bebop_autonomy*" para ROS, montado en un sistema operativo Linux, distribución Ubuntu 16.04 LTS.

Se evaluó diferentes algoritmos para la detección y extracción de puntos caracteristicos como ORB, BRIEF-START, BRIEF-FAST Y SURF, estos fueron aplicados a diferentes imágenes cada una con variaciones en textura, iluminación y rotaciones. La métrica para determinar la elección del algoritmo para esta etapa consistía en la cantidad de keypoints encontrados y el tiempo de procesamiento para encontrar dichos puntos. ORB a pesar de que no es un algoritmo compuesto como BRIEF-STAR o BRIEF-FAST presenta un buen desempeño en cuanto al tiempo de procesamiento y keypoints encontrados, es por ello que ORB es seleccionado como algoritmo extractor de características.

El uso del algoritmo de búsqueda RRT para una edificación conocida no es recomendable, porque es un algoritmo probabilístico y genera rutas aleatorias para los mismos puntos de partida y de destino. En algunos casos no genera llegaba al punto destino y se quedaba iterando de manera indefinida en un mismo punto por lo que no generaba la ruta. Como segunda opción se consideró al algoritmo A* presentado muy buenos resultados debido a su método de nodos vecinos y por tener una función de coste

baja, este algoritmo siempre genera la misma ruta y siempre encuentra una siempre y cuando no exista obstáculos o cierres que interfieran en la búsqueda de la ruta. Por las cuestiones mencionadas se elige el algoritmo A* como algoritmo buscador de camino para este proyecto.

El modelamiento servo visual se lo realizó para dos ejes: para el eje “X” y otra para el eje “Z”. Para el modelamiento servo visual se estimó el movimiento del UAV en base a la captura de fotogramas consecutivos, mediante los procesos de detección y extracción de puntos características, correspondencia de puntos, y transformaciones geométricas se logra estimar la matriz de transformación entre dos fotogramas la cual viene representada en forma matricial compuesta por una matriz de rotación y otra de traslación. Realizando eso de forma iterativa se logró estimar el movimiento para los diferentes ejes mencionados y posterior a ello mediante la herramienta ident de Matlab se logra estimar la función de transferencia de cada planta.

Se diseñó dos controladores para Pitch y para Yaw, que cumplen las condiciones de desempeño establecidas, para ello se utilizó la función de transferencia ya obtenida mediante ident. La sintonización de los dos controladores se lo realizó en PID Tuner, otra de las herramientas de Matlab. Dando como resultado las constantes K_p, K_i y K_d para cada controlador.

La inserción de marcadores ArUco al proyecto permitió identificar las zonas accesibles del mapa con un punto específico donde el dron tiene que llegar. De esta manera se logra determinar la posición del dron en el plano y comprobar si efectivamente llego al destino, así como también se logró establecer los marcadores como puntos de aterrizaje.

Recomendaciones

Realizar una elección adecuada de puntos de interés para reducir el coste computacional. Para la extracción de características utilizar no más de 500 puntos de interés, dado que a medida que aumentan el número de fotogramas o los fps (frames per second) el coste computacional incrementa. Además al tener muchos puntos de interés estos pueden generar falsos positivos en la etapa de matching y por consecuencia generar una matriz de transformación errónea los cuales influyen en la estimación de movimientos.

Una vez obtenido el plano del interior de la planta, realizar la etapa de procesamiento para bordear las zonas posibles de colisión. Lo ideal del procesamiento de imágenes es conseguir un plano con bordes (paredes) lo más anchos posibles sin que exista cierres de caminos. La elección de un tamaño de kernel, el valor de los filtros de media, gaussianos y la aplicación de otras técnicas ayudan a conseguir este efecto.

Una buena manera de establecer sistemas de referencia son los marcadores, como los usados en este proyecto. ArUco marker posee muchos diccionarios y cada diccionario decenas y cientos de marcadores, gracias a la librería abierta de OpenCV es muy sencillo la identificación y estimación de pose de un marcador mediante visión por computador. También es importante realizar la calibración de la cámara para el reconocimiento de estos marcadores. Existe una guía paso a paso en la plataforma de GitHub que explica cómo se realiza esta calibración.

En cuanto a la comunicación inalámbrica entre el dron y la PC, estos deben tener una buena intensidad de señal de manera que no se pierda la comunicación entre los dos al momento de realizar alguna prueba de vuelo o ejecutar un programa que implique el movimiento del dron. En el caso de perder la comunicación en pleno movimiento del UAV y no poder controlarlo se procede a tomar con una mano desde la parte superior sin topar

las hélices y se voltea el dron, la acción realizada hace que el dron se apague automáticamente.

Para la estimación de movimientos en el eje X (Pitch) y posteriormente la identificación de la planta, es recomendable tomar fotogramas a cierta tasa de muestreo, una vez generada los n fotogramas crear un script que permita procesar dichos fotogramas, o también se puede grabar un video desde cuándo empieza a actuar la señal de entrada aplicada hasta que termine. Si el procesamiento de imágenes se lo realiza paralelamente mientras el dron esta en movimiento esto provoca un cambio significativo en las respuestas o señal de salida que se llegan a tener.

Se recomienda usar como estación terrestre un PC de alto rendimiento que incluya una tarjeta de video dedicada, esto sobre todo si se requiere aplicaciones en donde se tenga procesamiento de imágenes en tiempo real. Con una PC normal los tiempos de procesamiento son lentos, llegan a sobrecalentar el hardware ocacionando un desgaste en los componentes físicos e internos.

. Para utilizar el UVA en entornos internos determinar zonas amplias y con una textura del piso que sea variable y no uniforme, también no debe reflejar luz porque esto ocasiona que el micro UAV pierda el control y desplace de manera inesperada e inestable. Si este fuera el caso se recomienda cubrir el piso con materiales u objetos de texturas y colores que se distingan del piso.

Antes de realizar pruebas de navegación con el UAV se debe tomar en cuenta ciertos aspectos como: Verificar si las hélices están en buenas condiciones y correctamente ajustadas, comprobar que la batería este cargada de ser posible al 100% y que esté sujeta muy fija al UAV.

Referencias

Addati, A., & Pérez, G. (2014). Introducción a los UAV's, Drones o VANTs de uso civil. *ECONSTOR*. Obtenido de http://hdl.handle.net/10419/130802

Aguilar, W. G., & & Angulo, C. (2014). Robust video stabilization based on motion intention for low-cost micro aerial vehicles. *IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*.

Aguilar, W. G., Angulo, C., & Pardo, J. A. (2017). Motion intention optimization for multirotor robust video stabilization. *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies*.

Aguilar, W. G., Casaliglla, V., & Pólit, J. (2017). Obstacle Avoidance Based Visual Navigation for MIcro Aerial Vehicles. *Electronics, 6*(1).

Aguilar, W., & Angulo, C. (2014). Real time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing, 1*, 1-13.

Aguilar, W., Salcedo, V., Sandoval, D., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation. *Computational Neuroscience*, 94-105.

Aguinaga, I., Borro, D., & Matey, L. (2008). *Parallel RRT-based path planning for selective disassembly planning.* International Journal of Advanced Manufacturing Technology.

Andrade, G. (2015). *Correspondencia multiespectral en el espacio de houg [Tesis Pregrado].* Escuela Superior Politécnica del Litoral, Guayaquil.

Babinec, A., Jurisica, L., Hubinský, P., & Duchon, F. (2014). Visual localization of mobile robot using artificial markers. *Procedia Engineering*.

Bacik, J., Durovsky, F., Fedor, P., & Perdukova, D. (2017). Autonomous flying with quadrocopter using fuzzy control and ArUco markers. *Intelligent Service Robotics*.

Bay, H., & Tuyelaars, T. V. (2006). SURF: Speeded up robust features. *Europen Conference on Computer Vision*, 404-417.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up robust features. *Proceedings of the 9th European conference on Computer Vision - Volume Part I*, 404-417. doi:10.1007/11744023_32

Bayindir, L., & Sahin, E. (2007). A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering & Computer Sciences, 15*(2), 115-147.

Besada, J., Bergesio, L., Campana, I., Vaquero-Melchor, D., Lopez Araquistain, J., Bernardos, A., & Casar, J. (2018). Drone Mission Definition. *Sensors, 18*, 1170.

Bianco, S., Mazzini, D., Pau, D., & Schettini, R. (2015). Local detectors and compact descriptors for visual search: a quantitative comparison. *Digital Signal Process*, 1-13.

Cancilieri, R., Chiapparo, J., Vommaro, D., & Verrastro, S. (2019). Caracterización de Planta para un Cuadricóptero. *Revista Tecnología Y Ciencia, 34*, 54-79. doi:https://doi.org/10.33414/rtyc.34.54-79.2019

Candoler, M., Lepetit, V., Strecha, C., & Fua, P. (2010). BRIEF: Binary Robust independient Elementary Feature. *Eur Conf Comput Vis*, 778-792.

Chakravarty, P., Kelchtermans, K., Roussel, T., Wellens, S., Tuytelaars, T., & Van Eycken, L. (2017). CNN-based Single Image Obstacle Avoidance on a Quadrotor. *IEEE International Conference on Robotics and Automation (ICRA)*, 1-9.

Chasillacta, M. (2020). *Reconocimiento y seguimiento de plataformas para el aterrizaje automático de un vehículo aéreo no tripulado basado en inteligencia artificial y odometría visual [Tesis Pregrado].* Universidad de las Fuerzas Armadas-ESPE.

Chauca, B. (2020). *Seguimiento y búsqueda de objetivos en entornos complejos usando micro vehículos aéreos con cámaras monoculares para aplicaciones militares [Tesis Pregrado].* Universidad de las Fuerzas Armadas-ESPE.

Chaves, A., & Cugnasca, P. (2014). Adaptive search control applied to Search and Rescue operations using Unmanned Aerial Vehicles (UAVs). *IEEE Latin America Transactions, 12*(7), 1279-1283.

Chouza, A., Hernandez, R., Jimenez, J. L., & Orozco-Rosas, U. (2020). Navegación autónoma de un vehículo aéreo por referencia visual. *Número Especial de la Revista Aristas: Investigación Básica y Aplicada, 8*(15).

Costa, A., & Fernandez, J. (2005). La imagen digital. *Fotografía digital*, 255-266.

De La Escalera, A. (2001). Visión por Computador: fundamentos y métodos. *Prentice Hall*.

Ehsan, S., & McDonald-Maier, K. (2009). On-Board Vision Processing for Small UAVs: Time to Rethink Strategy. *NASA/ESA Conference on Adaptive Hardware and Systems*, 75-89. doi:10.1109/AHS.2009.6

Elbanhawai, M., & Simic, M. (2014). Sampling-based robot motion planning. *A review Access IEEE, 2*, 56-77.

Espitia, H., & Sofrony, J. (2011). ALGORITMO PARA LA PLANEACIÓN DE TRAYECTORIAS DE ROBOTS MÓVILES EMPLEANDO ENJAMBRES DE PARTÍCULAS BROWNIANAS. *Visión Electrónica*.

García, A. (2013). *ROS: Robot Operating System.* Universida politécnica de Cartagena, Cartagena.

García, E. (2017). Visió artificial. *FUOC. Fundació per a la Universitat Oberta de Catalunya*.

García, L., Dzul, A., Lozano, R., & Pégard, C. (2012). Quad rotorcraft control: vision-based hovering and navigation. *Springer Science & Business Media*.

Garijo, D., López, J., & Pérez, I. (2009). *Control de un vehículo aéreo no tripulado [versión PDF].* Madrid. Obtenido de https://eprints.ucm.es/id/eprint/9477/1/documentacion.pdf

Garrido, S., Muñoz, R., Cuevas, D., & Marín, M. (Junio de 2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recogn*.

Gasparetto, A., Boscariol, P., Lanzutti, A., & Vidoni, R. (2015). Path Planning and Trajectory Planning Algorithms: A General Oveview. *Mechanisms and Machine Science*, 3-27.

Giernacki, W., Kozierski, P., Michalski, J., Retinger, M., Madonski, R., & Campoy, P. (2020). Bebop 2 Quadrotor as a Platform for Research and Education in Robotics and Control Engineering. *International Conference on Unmanned Aircraft Systems (ICUAS)*.

Gómez, F., Ollero, A., López, D., Cuesta, F., & del Toro, M. (2017). *Planificación distribuida de caminos basada en la informacion de una red de sensores wireless.* Universidad de Sevilla, Departamento de Ingeniería de Sistemas y Automática.

Gonzales, A., Martinez, F., Pernía, A., Alba, F., Castejon, M., Ordieres, J., & Vergara, E. (2006). *Técnicas y algoritmos básicos de visión artificial.* Universidad de la Rioja.

Gonzales, E. (2015). *Desarrollo de un algoritmo planificador de rutas con capacidad de implementación en diversas aplicaciones de la robótica móvil [Tesis Pregrado].* Universidad Tecnológica de Pereira.

Gonzales, V., Monje, C., & Balaguer, C. (2015). Planificación de trayectorias con vehículos aéreos no tripulados en un entorno aeroportuario. *Actas de las XXXVI Jornadas de Automática*.

Grijalva, S. (2019). *Sistema de navegación autónomo basado en visión para pistas de carreras de drones con marcas de superficie [Tesis Pregrado].* Universidad de las Fuerzas Armadas-ESPE.

Guerrero, M., Lozano, B., & García, C. (2015). Control basado en pasividad para un quadrotor UAV. *Congreso Nacional de Control Automático. AMCA*, 81-86.

Guilherme, R. (2011). Robust control strategies for a QuadRotor helicopter. *DialNet*.

Gupta, S., Chonge, M., & Jawandhiya, P. (s.f.). Review of Unmanned Aircraft System. *International Journal of Advanced Research in Computer Engineering & Technology, 2*(4), 1646-1658.

Hart, P., Nilson, N., & Raphel, B. (1986). A formal basis for the heuristic determination of minimun cost paths. *Systems Science and Cybernetics, IEEE Transactions*, 100-107.

Hartley, R., & Zisserman, A. (2003). *Multiple View Geometry in Computer Vision.* Cambridge university press.

Hassaballah, M., Abdelmgeid, A., & Alshazly, H. (2016). Image Features Detection, Description and Matching. *Studies in Computational Intelligence*, 11-45.

Herrera, P., Guijarro, M., & Guerrero, J. (2016). Operaciones de transformación de imágenes. *Conceptos y Métodos en Visión por Computador* , 61-76.

Hirzer, M., Roth, P., Kostinger, M., & Bischof, H. (2012). Relaxed Pairwise Learned Metric for Person Re-Identification. *Research Gate*.

Huang, H., Hoffmann, G., Waslander, S., & Tomlin, C. (2009). Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering. *IEEE International Conference on Robotics and Automation*. doi:10.1109/ROBOT.2009.5152561

Hughes, P. (2012). Spacecraft attitude dynamics. *Courier Corporation*.

Iglesias, D. (2004). La gestión de la imagen digital. *Hipertex.net*(2).

Ismail, N., Tang, S., & Khaksar, W. (2012). A review on robot motion planning aproaches. *Pertanika Journal of Science and Technology, 20*(1), 15-29.

Jara, A., Abad, J., Navrro, J., & Page, A. (Julio de 2019). Validación de los marcadores ArUco para el análisis de. *11º Simposio CEA de Bioingeniería*.

Khamseh, H. D., Lotfi, F., & Taghirad, H. H. (2019). Position Estimation for Drones based on Visual SLAM and IMU in GPS-denied Environment. *International Conference on Robotics and Mechatronics (ICRoM)*.

Kharsansky, A. (2013). Diseño e implementación de un sistema embebido de control de actitud para aeronaves no tripuladas. *[Tesis de pregrado]*, 113.

Li, Y., Zhu, J., Hoi, S. C., Song, W., Wang, Z., & Liu, H. (2019). Robust Estimation of Similarity Transformation for Visual Object Tracking. *AAAI Conference on Artificial Intelligence, 33*(01), 8666-8673. doi:https://doi.org/10.1609/aaai.v33i01.33018666

Lindemann, S., & La Valle, S. (2004). Steps Toward Derandomizing RRTs. *IEEE Fourth International Workshop on Robot Motion and Control*.

López, D. (2012). *Nuevas aportaciones en algoritmos de planificación para la ejecución de maniobras en robots autónomos no holónomos [Tesis Doctoral].* Huelva.

López, M., Garcia, S., Barea, R., Bergasa, L., Molinos, E., Arroyo, R., & Pardo, S. (2017). A Multi-Sensorial Simultaneous Localization and Mapping (SLAM) System for Low-Cost Micro Aerial Vehicles in GPS-Denied Environments. *Sensors 2017, 17*(4).

Lowe, D. (2004). Distinctive image features drom scale-invariant keypoints. *Int J comput Vis, 60*(2), 91-110.

Lugo, R. (2018). Sistema para la Ejecución de Trayectorias. *[Tesis de grado]*, 18.

Mac, T. T., Copot, C., Hernandez, A., & De Keyser, R. (2016). Improved potential field method for unknown obstacle avoidance using UAV in indoor environment. *IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 345-350. doi:10.1109/SAMI.2016.7423032

Mahadeva, M. (7 de Egosto de 2020). *Marcadores ArUco: uso en Visión por Computadora usando OpenCV - python.* Obtenido de muralimahadeva.medium.com:

https://muralimahadeva.medium.com/aruco-markers-usage-in-computer-vision-using-opencv-python-cbdcf6ff5172

Marin, H., Sucar, L., & Morales, E. (2007). Automatic image annotatio using a semi supervised ensemble of classifiers. *Computer Science*, 487-495.

Márquez, F., Maza, I., & Ollero, A. (2015). Comparación de planificadores de caminos basados en muestro para un robot aéreo equipado con brazo manipulador. *Jornadas de Automática*.

Martínez, B., Pineda, L., Martínez, M., De Ávila, D., & Hernández, L. (2012). Identificación de un vehículo aéreo no tripulado. *Ingeniería Electrónica, Automática y Comunicaciones, 33*(1), 45-55. Obtenido de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59282012000100005&lng=es&tlng=es

Marut, A., Wojtowicz, K., & Falkowski, K. (Junio de 2019). ArUco markers pose estimation in UAV landing aid system. *2019 IEEE 5th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*.

Meythaler, A., Rivas, D., Chicaiza, F., & Chuchico, C. (2016). *Desarrollo de un sistema de Navegación Autónoma para UAV basado en FPGA.*

Miguélez Machado, C. G., Benítez González, I. O., Rivera Rivera, A. M., & Moreno Vega, V. (2020). mplementación de sistema operativo robótico en una plataforma de robot móvil. *Ingeniería Electrónica, Automática y Comunicaciones, 41*(3), 79-92.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptor. *IEEE Trans Pattern Anal Mach Intell, 27*(10), 1615-1630.

Mikolajzyk, K., & Tuytelaars, T. (2005). A comparison of affine region detectors. *int J Comput Vis, 65*(1), 43-72.

Min, B., Cho, C., Choi, K., & Kim, D. (2009). Development of a Micro Quad-Rotor UAV for Monitoring an Indoor Environment. *Advances in Robotics, 5744*, 262-271. doi:https://doi.org/10.1007/978-3-642-03983-6_30

Mojica, P., Cuellar, S., & Medina, C. (2015). Vehículos Aéreos No Tripulados. *DRONES y sus sistemas de comunicación*.

Monajjemi, M. (2015). *bebop_autonomy - ROS Driver for Parrot Bebop Drone (quadrocopter) 1 & 2.*

Monajjemi, M., Mohaimenianpour, S., & Vaughan, R. (2016). UAV, Come To Me: End-to-End, Multi-Scale Situated HRI with an Uninstrumented Human and a Distant UAV. *International Conference on Intelligent Robots and Systems (IROS)*, 1-8.

Mondragón, I. (2011). *On-board visual control algorithms for unmanned aerial vehicles.* Universidad Politécnica de Madrid, Madrid.

Montes, Á. M. (2017). *Planificación de caminos basada en modelo combinando algoritmos de búsqueda en grafo, derivados de RRT y RRT*.* [Trabajo Fin de Máster Inédito], Universidad de Sevilla, Sevilla. Obtenido de https://idus.us.es/handle/11441/73041

Moravec, H. (1980). Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. *Tech Report CMU-RI-TR-3 Carnegie-Mellon University*.

Mordvintsev, A., & Abid, K. (2013). *Feature Matching.*

Moscoso, M. (2016). *Plan de negocios para la creación de una empresa de capacitación y entrenamiento de guardias de seguridad en la ciudad de Quito.* Quito.

Muñoz, J. (2019). *Implementación de algoritmos de planificación de trayectorias para robots móviles en entornos complejos.* Zaragoza: Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza.

Muñoz, R. (2018). *ArUco: An efficient library for detection of planar markers and camera pose .*

Narro, A. (2019). *Comparativa de algoritmos de detección de características para vision artificial .* Universidad Politécnica de Madrid, Madrid.

Nils Nilsson, J. (1969). A Mobile Automaton: An Application of Artifi cial Intelligence Techiques. *1st International joint Conference on Artificial Intelligence*.

Ogata, K. (2007). *Ingenieria de control moderna.* Prentice Hall.

OpenCV. (2019). *Python OpenCV: Procesamiento de Imagenes.*

Orozco-Soto, S., Ibarra-Zannatha, J., Malo-Tamayo, A., & Cureno-Ramirez, A. (2018). Active Disturbance Rejection Control for UAV Hover using ROS. *2018 XX Congreso Mexicano*, 1-5.

Pajares, G., & de la Cruz, J. (2007). *Vision por computador: imágenes digitales y aplicaciones.* Madrid: RA-MA.

Pajares, G., & De La Cruz, J. (2010). Aprendizaje Automático, Un enfoque práctico. *RA-MA*.

Pajares, G., de la Cruz, J., Molina, J., Cuadrado, J., & López, A. (2003). Imágenes Digitales: procesamiento práctico con JAVA. *RA-MA*.

Parrot. (2020). *Drones*. Obtenido de Support - Parrot Bebop 2: https://www.parrot.com/en

Pinto, R. (2016). *DRONES: La tecnología, ventajas y sus posibles aplicaciones.*

Pistarelli, M. (2013). *Detección y correspondencia de características en imágenes multiespectrales.* Universidad Autónoma de Barcelona.

Plaza, D. (2017). *Navegación autónoma de drones y automatización de rutas aplicadas a la limpieza de edificios.*

Puerto, K. (2016). Parrot Bebop 2, análisis: autonomía y facilidad de vuelo ideales para entrar en el mundo de los drones. *Xataka*.

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

Redondo, M. (2016). *Evaluación comparativa de técnicas de detección y descripción de puntos de interés en imágenes [Tesis Pregrado].* Universidad Autónoma de Madrid, Madrid.

Rojo, T., & Ramos, M. (2019). *Seguridad en centrales de monitoreo*. Obtenido de Seguridad en América.

Rosaldo-Serrano, M., & Aranda-Bricaire, E. (2019). Trajectory Tracking for a Commercial Quadrotor. *6th International Conference on Control, Decision and Information Technologies*, 1616-1621.

Rosten, E., & Drummond, T. (2006). Machine Learning for High-Speed Corner Detection. *European Conference on Computer Vision, 3951*, 430-443. doi:https://doi.org/10.1007/11744023_34

Rublee, E., Rabaud, V., konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *IEEE Int Conf Comput Vis* , 2564-2571.

Sahin, E., Girgin, S., Bayindir, L., & Turgut, A. (2008). Swarm Intelligence: Introduction and Applications. *Natural Compunting*, 87-100.

Salcedo, V. (2018). *Aterrizaje automático de un vehículo aéreo no tripulado basado en seguimiento de puntos de interes para superficies móviles [Tesis Pregrado].* Universidad de las Fuerzas Armadas - ESPE.

Sani, J., Tierra, A., & Robayo, A. (2015). Vehículos aéreos no tripulados - UAV para la elaboración de cartografías escalas grandes referidas al marco de referencia SIRGAS-ECUADOR. *X Congreso de Ciencia y Tecnología ESPE 2015, 10*(1).

Santamaria, L., & Kemper, N. (2014). Desarrollo de un sistema difuso aplicado al despegue de micro-UAV. *SOMI: Congreso de instrumentación XXIX Edición*.

Santana, E. (2017). *Propuesta de sistema multi-UAV para aplicaciones de cobertura de área [Tesis Doctoral].* Universidad Autónoma de Barcelona.

Schnipke, E., Reidling, S., Meiring, J., Jeffers, W., Hashemi, M., Tan, R., . . . Kumar, M. (2015). *Aeospace Research Central*. Obtenido de Autonomous Navigation of UAV through GPS-Denied Indoor Environment with Obstacles: https://arc.aiaa.org/doi/10.2514/6.2015-0715

Segovia, A., & Rombaut, M. (1993). Path Finding from a Spot Image for a Mobile Robot. *Intelligent vehicles 93 Symposium*.

Serrano, D. (2011). *Introduction to ROS - Robot Operating System.*

Shapio, L., & Stockman, G. (2001). Computer vision. *Pretice Hall*, 53-54.

Spring, K., & Russ, J. (2018). Concepts in Digital Imaging Technology. *Hamamatsu*.

Stumberg, L., Usenko, V., Engel, J., Stuckler, J., & Cremens, D. (2017). From monocular SLAM to autonomous drone exploration. *Proceedings of the 2017 European Conference on Mobile Robots (ECMR)*, 1-8.

Sucar, I., & Gomez, G. (2013). Vision computacional. *Instituto Nacional de Astrofísica, Óptica y Electrónica*.

Sucar, L. (2018). Visión de Alto Nivel. *INAOE*, 1-52.

Thu, K., & Gavrilov, A. (2017). Designing and Modeling of Quadcopter Control System Using L1 Adaptive Control. *Procedia Computer Science, 103*, 528-535. doi:https://doi.org/10.1016/j.procs.2017.01.046.

Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., . . . Burschka, D. (2012). Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue. *IEEE Robotics & Automation Magazine, 19*(3), 46-56. doi:10.1109/MRA.2012.2206473

Tormes, R., García, A., & Benítez, E. (2007). Sistema Operativo ROS. *UPTOS. "Clodosbaldo Russian"*.

University of Nevada, Reno. (2018). *2D Geometrical Transformations.*

Valavanis, K., & Vachtsevanos, G. (2015). UAV Applications: Introduction. *Handbook of Unmanned Aerial Vehicles*, 2639-2641. doi:https://doi.org/10.1007/978-90-481-9707-1_151

Valero, C. (2017). *Desarrollo de aplicaciones basadas en visión con entornos ROS para el Drone Bebop2.* Universidad de Alcalá Escuela Politécnica Superior.

Van de Sande, K., Geves, T., & Snoek, C. (2010). Evaluation color descriptors for object and scene recognition. *IEEE Trans Pattern Anal Mach Intell, 32*(9), 1582-1596.

Vinogradov, E., Sallouha, H., De Bast, S., Azari, M., & Pollin, S. (2019). Tutorial on UAV: A Blue Sky View on Wireless Communication. *Networking and Internet Architecture*.

Wilbert G., A., & Morales, S. (2016). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics, 5*(70).

Wubben, J., Fabra, F., Calafate, C., Krzeszowski, T., Marquez-Barja, J., Cano, J., & Manzoni, P. (2019). Accurate landing of unmanned aerial vehicles using ground pattern recognition. *Electronics*.

Yang, H., Lee, Y., Jeon, S. Y., & Lee, D. (2017). Multi-rotor drone tutorial: systems, mechanics, control and state estimation. *Intelligent Service Robotics, 10*(2), 79-93.

Yang, J., Huo, X., Xiao, B., Fu, Z., Wu, C., & Wei, Y. (2017). Visual servo control of unmanned aerial vehicles: An object tracking-based approach. *29th Chinese Control And Decision Conference*.

Zhang, J., Wu, Y., Liu, W., & Chen, X. (2010). Novel Approach to Position and Orientation Estimation in Vision-Based UAV Navigation. *EEE Transactions on Aerospace and Electronic Systems, 46*(2), 687-700. doi:10.1109/TAES.2010.5461649

Zhou, X., Wang, K., & Fu, J. (2016). A Method of SIFT Simplifying and Matching Algorithm Improvement. *International Conference on Industrial Informatics - Computing*

Technology, Intelligent Technology, Industrial Information Integration (ICIICII, 2, 73-77. doi:10.1109/ICIICII.2016.0029

Zidane, I., & Ibrahim, K. (2018). *Wavefront and A-Star Algorithms for Mobile Robot Path Planning.*

Zipfel, P. (2009). Modeling and simulation of aerospace vehicle dynamics. *American Institute of Aeronautics y Astronautics*.