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

TEMA: "DESARROLLO DE UN SISTEMA SIMULTÁNEO DE SEGUIMIENTO DE PERSONA Y EVASIÓN DE OBSTÁCULOS NO DEFINIDOS USANDO ESTIMACIÓN DE PROFUNDIDAD PARA IMÁGENES MONOCULARES EN UN MICRO-UAV"

AUTOR: QUISAGUANO PAREDES, FERNANDO JEFFERSON

DIRECTOR: ING. AGUILAR CASTILLO, WILBERT GEOVANNY PhD.

SANGOLQUÍ

2018

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 "**DESARROLLO DE UN SISTEMA SIMULTÁNEO DE SEGUIMIENTO DE PERSONA Y EVASIÓN DE OBSTACULOS NO DEFINIDOS USANDO ESTIMACIÓN DE PROFUNDIDAD PARA IMÁGENES MONOCULARES EN UN MICRO-UAV**" fue realizado por el señor **QUISAGUANO PAREDES, FERNANDO JEFFERSON** el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 27 de noviembre de 2018

Ing. Aguilar Castillo Wilbert Geovanny PhD.

DIRECTOR

CC: 0703844696

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Yo, **Quisaguano Paredes, Fernando Jefferson,** con cédula de identidad Nº 1725474231, declaro que el contenido, ideas y criterios del trabajo de titulación : **"DESARROLLO DE UN SISTEMA SIMULTÁNEO DE SEGUIMIENTO DE PERSONA Y EVASIÓN DE OBSTACULOS NO DEFINIDOS USANDO ESTIMACIÓN DE PROFUNDIDAD PARA IMÁGENES MONOCULARES EN UN MICRO-UAV"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, 27 de noviembre de 2018

Fernando Jefferson Quisaguano Paredes

CC: 1725474231

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo, **Quisaguano Paredes, Fernando Jefferson** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"DESARROLLO DE UN SISTEMA SIMULTÁNEO DE SEGUIMIENTO DE PERSONA Y EVASIÓN DE OBSTACULOS NO DEFINIDOS USANDO ESTIMACIÓN DE PROFUNDIDAD PARA IMÁGENES MONOCULARES EN UN MICRO-UAV"** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 27 de noviembre de 2018

Fernando Jefferson Quisaguano Paredes

CC: 1725474231

DEDICATORIA

Este proyecto quiero dedicarlo especialmente a mis padres Hernán y Zoila por siempre acompañarme, guiarme y apoyarme en todo momento, ellos han sido mi fortaleza y mi inspiración más grande para alcanzar este logro.

También dedico este trabajo a mi hermano André que con su apoyo y su amistad sincera me ha ayudado a levantar en los peores momentos, espero ser un gran ejemplo a seguir en su vida profesional.

AGRADECIMIENTOS

Agradezco a Dios por brindarme la oportunidad de culminar mis estudios académicos en esta prestigiosa Universidad.

Por todo el amor que mis padres me han brindado durante toda mi vida, sin ellos no sería nada, a mi madre por darme fuerza en los peores momentos y mostrarme toda su fortaleza como toda una mujer guerrera y valiente, a mi padre por toda la dedicación, por sus consejos y apoyo incondicional.

Al Doctor Wilbert Aguilar ya que gracias a su apoyo y guía durante todo este proyecto lo he podido culminar satisfactoriamente, un ejemplo como profesional y persona.

A mis amigos por su ayuda constante durante esta etapa de mi vida han sido un pilar fundamental para la consecución de este objetivo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En este proyecto de investigación, se presenta el desarrollo de un sistema para la detección de persona y evasión de objetos no definidos en tiempo real utilizando la cámara a bordo en un micro-UAV. Para la detección de la persona se lo realiza a través de un entrenamiento redes neuronales convolucionales. Para este entrenamiento se ha realizado con una gran cantidad de imágenes de la persona a identificar. El seguimiento de la persona es posible una vez se haya identificado la persona a seguir, se realiza el seguimiento a través de algoritmo de "medianflow". El movimiento del micro-UAV es posible por la identificación de la planta a controlar que se realiza a través de la estimación de movimiento obtenida por el cálculo de movimiento utilizando "optical flow" de "Lucas kanade". La evasión de obstáculos se lo realizar con la ayuda de la estimación de profundidad mediante un algoritmo no supervisado, el mapa de profundidad obtenido sirve como base para el cálculo de una función de aproximación y ponderación para la selección de la zona menos densa en el mapa. El sistema es sometido a pruebas experimentales las cuales son realizadas en dos escenarios diferentes uno controlado y el otro no, presentando un seguimiento y una evasión satisfactoria.

PALABRAS CLAVE:

- **REDES NEURONALES CONVOLUCIONALES**
- **VISIÓN ARTIFICIAL**
- **VEHÍCULO AÉREO NO TRIPULADO**

ABSTRACT

In this research project, the development of a system for the detection of person and evasion of undefined objects in real time is presented using the on-board camera in a micro-UAV. For the detection of the person it is done through a convolutional neural network training. For this training has been made with a large number of images of the person to identify. The tracking of the person is possible once the person to be followed has been identified, the follow-up is done through the "medianflow" algorithm. The movement of the micro-UAV is possible due to the identification of the plant to be controlled, which is carried out through the estimation of movement obtained by the calculation of movement using “optical flow” from "Lucas Kanade". The evasion of obstacles is carried out with the help of depth estimation by means of an unsupervised algorithm, the obtained depth map serves as a basis for the calculation of an approximation and weighting function for the selection of the least dense zone on the map. The system is subjected to experimental tests which are carried out in two different scenarios, one controlled and the other not, presenting a satisfactory follow-up and evasion.

KEYWORDS:

- **CONVOLUTIONAL NEURAL NETWORKS**
- **ARTIFICIAL VISION**
- **UNMANNED AERIAL VEHICLE**

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Antecedentes

Automatizar la detección visual y el seguimiento de objetos en movimiento mediante sistemas autónomos inteligentes en vehículos aéreos no tripulados (UAV, por sus siglas en inglés), ha sido un tema de investigación recurrente durante las últimas décadas en visión por computadora. Esta investigación tiene diversas aplicaciones que se extienden desde el ejército, la vigilancia, los sistemas de seguridad, la fotografía aérea, la búsqueda y el rescate, el reconocimiento de objetos, la navegación automática a las interacciones hombre-máquina (Yilmaz, Javed, & Shah, 2006).

La capacidad de detectar y evitar los obstáculos como las aves que vuelan en un bosque es fascinante y ha sido objeto de mucha investigación por sus múltiples aplicaciones (Tomoyuki Mori, 2013). Por ese motivo se han desarrollado métodos de servo-control visual basado en imágenes que utiliza solo una cámara orientada hacia adelante para rastrear y seguir objetos desde un UAV permitiéndole al usuario especificar un objeto en la imagen que el UAV debe seguir desde una distancia constante aproximada (Mondragon, Campoy, Olivares-Mendez, & Martinez, 2011)(Jesús Pestana, Sanchez-Lopez, Campoy, & Saripalli, 2013)(Jesus Pestana, Sanchez-Lopez, Saripalli, & Campoy, 2014). Así también en la utilización de múltiples UAV's para detección de personas con métodos de control y aprendizaje como "reinforcing learning" (Wang, Qin, Chen, Snoussi, & Choi, 2018).

En esta última década, el interés en los UAV y su autonomía ha aumentado constantemente, la prevención de colisiones es un requisito importante para los vuelos autónomos. Aunque existen

múltiples soluciones para la detección de obstáculos y la prevención de colisiones en un UAV, estas soluciones pueden presentar diferentes inconvenientes. Por ejemplo tradicionalmente, la evasión de obstáculos para los robots se ha realizado utilizando detección activa como LIDAR, luz estructurada, sonar o IR (Gageik, Benz, & Montenegro, 2015), la desventaja que tiene es el costo, el costo a nivel económico como también en términos de un peso extra, además cabe recalcar que influiría significativamente en los requisitos de potencia sobre todo cuando se muestra en plataformas de vuelo pequeñas, por otra parte también se ha usado detección pasiva como es estéreo visión (Oleynikova, Honegger, & Pollefeys, 2015).

Estéreo visión en imágenes de profundidad se pueden usar para la planificación de rutas (W. Aguilar & Morales, 2016)(W. G. Aguilar, Morales, Ruiz, & Abad, 2017)(W. G. Aguilar, Abad, & Ruiz, 2018)(W. G. Aguilar, Morales, & Ruiz, 2018), pero teniendo una línea de base pequeña entre cámaras en un dron, muestra problemas por su alcance limitado. Actualmente para Estéreo Visión existe en el mercado un kit para investigadores de navegación y evasión de obstáculos en drones, proporcionado por la compañía Parrot llamada "S.L.A.M. drunk", que cuenta con un procesador NVIDIA Tegra K1, dos cámaras para stereo visión, sensores ultrasónicos, como también barómetro y magnetómetro con un costo de 950 euros (Parrot Drone, s/f).

Recientemente, para imitar el comportamiento humano se ha desarrollado algoritmos basados en el aprendizaje para predecir caminos directamente desde una imagen RGB, especialmente con la popularidad de las redes neuronales convolucionales (CNN) (Giusti et al., 2016). Las redes neuronales convolucionales (CNN), son eficaces para analizar muestras de datos de longitud fija, como imágenes (Tokui, Oono, Hido, & Clayton, 2015). Con imágenes individuales se estudiaron por primera vez para la extracción de profundidad, utilizando "Supervised Learning" (Saxena,

Chung, & Ng, 2006). En esa investigación se usaron grandes cantidades de datos, imágenes y sus mapas de profundidad correspondientes, para capacitar a un clasificador y crear mapas de profundidad a partir de imágenes individuales. Pero con el éxito de “Deep Learning” el objetivo en donde más se utilizan las CNN es dando un enfoque hacia la extracción de profundidad de una sola imagen (Eigen, Puhrsch, & Fergus, 2014)(F. Liu, Shen, Lin, & Reid, 2016).

Los métodos basados en el aprendizaje han mostrado resultados muy prometedores para la tarea de estimación de profundidad en imágenes individuales. Sin embargo, la mayoría de los enfoques existentes tratan la predicción de la profundidad como un problema de regresión supervisada y, como resultado, requieren grandes cantidades de datos de profundidad. Un estudio en el 2017 sugiere un método novedoso que permite a la CNN a realizar una estimación de profundidad de una sola imagen no supervisado (Godard, Mac Aodha, & Brostow, 2016). Este estudio utiliza geometría epipolar, generando imágenes de disparidad al entrenar la red.

Paralelamente ha avanzado la investigación en estos últimos años en la detección de objetos por medio de redes neuronales convolucionales (CNN), entre los detectores de objetos modernos basados en estas redes neuronales convolucionales se tiene a Faster R-CNN (Ren, He, Girshick, & Sun, 2015), R-FCN (Dai, Li, He, & Sun, 2016), SSD (W. Liu et al., 2016) and YOLO (Redmon, Divvala, Girshick, & Farhadi, 2015) que son lo suficientemente buenos como para ser implementados en productos de consumo (e.g., Google Photos, Pin-terest, Visual Search) y algunos han demostrado ser lo suficientemente rápidos como para ejecutarse en dispositivos móviles (Cvpr & Id, 2017). Todos han sido implementados en Tensorflow de código abierto y han tenido una influencia considerable en la comunidad de visión artificial.

Una de las interfaces de comunicación y procesamiento de control más utilizados es ROS (Robot Operating System). El estudio de esta plataforma en micro-UAVs, específicamente en drones "Bebop" de la empresa Parrot se puede utilizar para acceder a las imágenes tomadas con la cámara a bordo del dron ROS. "Bebop_autonomy" es un controlador basado en el oficial "Parrot ARDro-ne_SDK3 SDK". Esto permite el uso de varias funciones del dron ya que este controlador permite utilizar las principales funciones como el vuelo de elevación, aterrizaje, navegación y más a través de la correspondientes tópicos (W. G. Aguilar & Salcedo, 2017).

1.2. Justificación e Importancia

Hoy en día, la utilización de drones ha mostrado una relevancia exponencial tanto en la investigación y como también en el ámbito comercial con un número creciente de aplicaciones, por ejemplo: búsqueda y rescate, vigilancia, inspección aérea de estructuras, monitoreo agroindustrial y forestal, videografía, entre otras. Dichas actividades se realizan tanto en ambientes exteriores como interiores densamente poblados. Por lo que se requiere que un dron vuele de manera eficiente, autónoma con la capacidad de detección y evasión de obstáculos (Martínez-Carranza, Valentín, Márquez-Aquino, González-Islas, & Loewen, 2016). Debido a las ventajas de los vehículos aéreos no tripulados (UAV, por sus siglas en inglés), tales como la extensibilidad, la maniobrabilidad y la estabilidad, teniendo cada vez más aplicaciones en la vigilancia de seguridad. Convirtiéndose en la búsqueda de objetos y la planificación de trayectorias en problemas importantes a resolver (Wang et al., 2018).

La detección o evasión de obstáculos en tiempo real es un problema desafiante especialmente para micro y pequeños vehículos aéreos no tripulados debido al número limitado de sensores que se puede tener a bordo y a la restricción de la batería por la baja carga útil que esta presenta (Al-

kaff, Meng, & Mart, 2016). Los sensores de tiempo de vuelo como LIDAR y los sensores de luz estructurados pueden ser costosos, tanto en términos de costo real como en términos de peso y requisitos de potencia, ambos de mayor valor cuando se trata de plataformas de vuelo pequeñas. Sonar e IR solo son adecuados para detección a muy corta distancia (dentro de un metro).

En la actualidad hay una gran cantidad de cuadricópteros de bajo costo en el mercado que vienen con una cámara frontal y son controlables mediante Wi-Fi usando teléfonos inteligentes habilitados para aplicaciones. Estos vehículos aéreos no tripulados, en manos de pilotos inexpertos, son propensos a choques regulares porque tienen poca capacidad autónoma de volar. El problema de chocar contra obstáculos es aún más evidente en ambientes interiores cubiertos de obstáculos (Chakravarty et al., 2017).

Por este motivo resulta importante, una investigación que se base en el seguimiento de una persona utilizando imágenes proporcionadas por la propia cámara a bordo del dron con el objetivo simultaneo de detectar y evadir obstáculos sin aumentar el consumo de batería, ya que no se agregará sensores extra y en cambio se utilizará algoritmos de estimación de profundidad y detección de objetos de una sola imagen.

1.3. Alcance del Proyecto

El presente trabajo de investigación plantea el desarrollo de un sistema autónomo utilizando algoritmos de estimación de profundidad monocular y detección de objetos para un micro-UAV basados en redes neuronales convolucionales con el objetivo simultaneo de seguimiento de persona y evasión de obstáculos, sin depender de otros sensores que no sea la cámara a bordo del micro-UAV.

Inicialmente, el trabajo de investigación consistirá en el estudio de técnicas empleadas para la determinación de la estimación de profundidad monocular tanto para entornos abiertos como en entornos cerrados. De la misma manera para técnicas de detección de objetos por medio del entrenamiento con redes neuronales convolucionales, el número de objetos a detectar será igual al número de objetos entrenados e ingresados en la red neuronal convolucional. Con esto se elegirá la mejor opción para la detección de seguimiento de persona que será el objetivo del dron a una distancia de 2 a 4 metros conjuntamente con la evasión de obstáculos no definidos.

La siguiente etapa del trabajo de investigación se centrará en la integración ambas técnicas de estimación de profundidad como la detección objetos implementadas en un micro-UAV, de esta forma diseñar un controlador para evasión que será implementado en el entorno ROS (Robot Operating System) conjuntamente con el driver de control para "Parrot Bebop drones" (cuadricópteros), basado en el "ARDroneSDK3" oficial de Parrot. El procesamiento de las imágenes y ejecución de los algoritmos se lo realizará desde una estación en tierra con un sistema operativo Ubuntu 16.04.

La última etapa del trabajo de investigación se centrará en el seguimiento de persona como objetivo en conjunto con la evasión de obstáculos analizada en la etapa anterior.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un sistema simultáneo de evasión de obstáculos y seguimiento de persona en un micro-UAV utilizando algoritmos de estimación de profundidad monocular y detección objetos basados en redes neuronales convolucionales.

1.4.2. Objetivos Específicos

- Desarrollar un estudio del estado del arte de algoritmos matemáticos para la estimación de profundidad monocular.
- Desarrollar un estudio del estado del arte de algoritmos matemáticos para la detección de objetos utilizando redes neuronales convolucionales.
- Determinar los parámetros del controlador necesarios para la evasión de obstáculos del micro-UAV y seguimiento de un objetivo.
- Integrar en un solo sistema autónomo la detección, evasión de obstáculos, seguimiento de persona en una estación en tierra que permita la comunicación y el control de un micro-UAV.
- Evaluar el desempeño de los algoritmos, a través de pruebas experimentales más severas en entornos abiertos con un mayor número obstáculos.

1.5. Descripción del Proyecto

El proyecto de investigación consiste en el desarrollo de un algoritmo que permita la evasión de obstáculos no definidos, simultáneamente el seguimiento de persona con un vehículo aéreo no tripulado, utilizando un dron “Bebop 2” de “Parrot” para la detección, evasión y seguimiento de una persona.

Para esta investigación el vehículo aéreo no tripulado realiza movimientos paralelos al plano, es decir, x e y, por simplicidad algorítmica no se controlará la altura del mismo, para la evasión de obstáculos considerando un rango máximo de detección de los mismo de 7 metros, así como un mínimo de 1,5 metros y el seguimiento de una persona con un rango de detección máximo de 5 metros y un mínimo de 1,5 metros. La persona es previamente identificada por medio de un

entrenamiento basado en redes neuronales convolucionales, para ello se obtiene una gran cantidad de imágenes de la persona a entrenar, con diferentes distancias, escenarios y alrededor de más personas.

Los obstáculos son detectados por medio de algoritmo no supervisado de estimación de profundidad el cual genera un mapa de estimación de profundidad, de este mapa se obtiene una función de aproximación y ponderación para la selección de la zona menos densa en el mapa.

Para el manejo de las funcionalidades del dron se ha realizado por medio del controlador oficial de ROS para dron “bebop 1” y “bebop 2” de Parrot “bebop_autonomy” el cual se ve más a detalle en el capítulo 3. Este controlador nos permite comunicarnos con el micro-UAV y a su ver obtener las imágenes en tiempo real del mismo.

Para el seguimiento la persona se utiliza el diseño e implementación de un controlador PI para el movimiento de “yaw”, en cambio para el movimiento de “pitch” y “roll” se utilizarán controladores on-off debido al análisis de condiciones que se ven en el capítulo 5. La identificación del seguimiento se lo realiza por medio de seguidor “medianflow”.

Una vez se tenga el seguimiento de la persona y la evasión de obstáculos se une ambos algoritmos por medio de una máquina de estados. Teniendo como resultado la finalización de proyecto.

CAPÍTULO II

2. MARCO CONCEPTUAL

2.1. Introducción

En este capítulo se detalla los conceptos necesarios para el presente proyecto de investigación tanto a nivel de software como a nivel de hardware principalmente en las herramientas utilizadas en el mismo.

2.2. Aviones pequeños de uso común

Para tener una visión clara del proyecto partiremos de la base principal que son los aviones pequeños de uso común (menos de 20 kg) que se pueden clasificar en aviones de ala fija, helicópteros y multicóptero , estos últimos también llamados multirrotores siendo estos los más populares hoy en día (Quan, 2017).

Figura 1 Avión de ala fija – Helicóptero – Multirrotor
Fuente: (Quan, 2017)

2.2.1. Multirrotores

Pueden considerarse como un tipo de helicóptero, que tiene tres o más hélices. El multirrotor más popular es el cuadricóptero, como se muestra en la Figura 2. Un cuadricóptero tiene cuatro entradas de control, que son las cuatro velocidades angulares de la hélice. Gracias a la estructura simple, un cuadricóptero es fácil de usar y presenta una alta confiabilidad y un bajo costo de

mantenimiento. Un cuadricóptero tiene cuatro hélices para generar el empuje y los momentos de tres ejes (Quan, 2017).

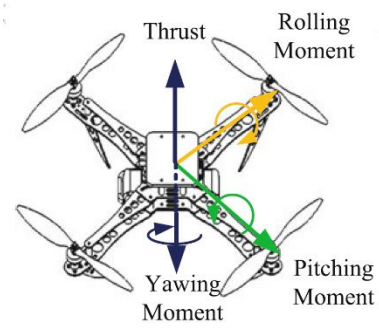

Figura 2 Cuadricóptero
Fuente: (Quan, 2017)

2.2.1.1. Dinámica del movimiento

El funcionamiento básico de un cuadricóptero, se basa en la secuencia de empuje que generan sus rotores, para poder lograr los movimientos, de manera que las orientaciones posibles del cuadricóptero son la configuración cruz equis (X) y la configuración positiva (+) (Leos Monroy, 2014).

Para el primer caso (Configuración +), este permite un mayor facilidad de maniobra, ya que para poder desplazarse en cualquier dirección, sea esta vertical u horizontal, solo es necesario el control y cambio de un solo motor (González, 2017).

Para el segundo caso (configuración X), los motores se encuentran separados a 45° uno del otro para los desplazamientos, es necesario la maniobra de los cuatro motores. Esta configuración es la más conveniente para el soporte de cámaras, ya que da un mayor rango de visualización, y los motores no intervienen en esto (González, 2017).

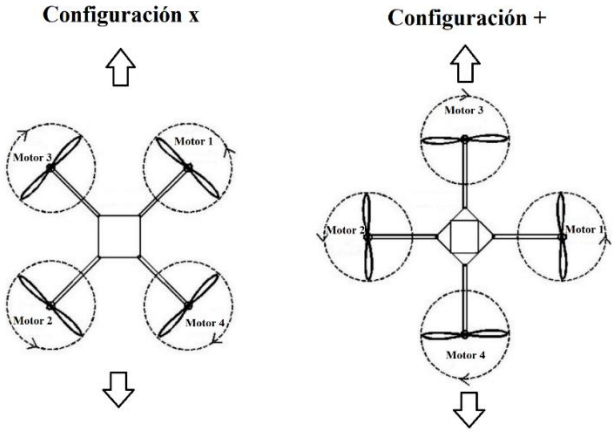

Figura 3 Tipos de configuración de cuadricóptero
Fuente: (Leos Monroy, 2014)

Los cuadricópteros, como todos los vehículos voladores, tienen tres grados de libertad llamados guiñada (yaw), inclinación (pitch) y bamboleo (roll) (Mayorga Rodríguez, 2009). Todos estos movimientos se muestran en la Figura 4.

Movimiento de guiñada (yaw)

Se refiere al movimiento cuando el cuadricóptero gira sobre su eje vertical. El cuadricóptero logra este movimiento aumentando (o disminuyendo) por igual la potencia de giro de rotores 1 y 3 y disminuyendo (o aumentando) en igual magnitud los motores 2 y 4. Este movimiento está definido por el ángulo de giro ψ (Mayorga Rodríguez, 2009).

Movimiento de inclinación (pitch)

Es el movimiento que permite dirigirse hacia adelante y atrás. El vehículo mantiene la potencia en el rotor 1 que es opuesto al sentido deseado, reduce al mínimo la del rotor 3 y deja los otros dos a potencia media (Mayorga Rodríguez, 2009). Así la sustentación del rotor 1 hace que el vehículo

se incline a favor del sentido deseado y se desplace. Este movimiento está definido por el ángulo de giro θ (Mayorga Rodríguez, 2009).

Movimiento de bamboleo (roll)

Es cuando el vehículo se mueve a la izquierda o derecha. Usa el mismo principio que el de inclinación, pero lateralmente. Este movimiento está definido por el ángulo de giro φ.

La combinación de los tres movimientos mencionados son los que hacen maniobrar al cuadricóptero libremente (Mayorga Rodríguez, 2009).

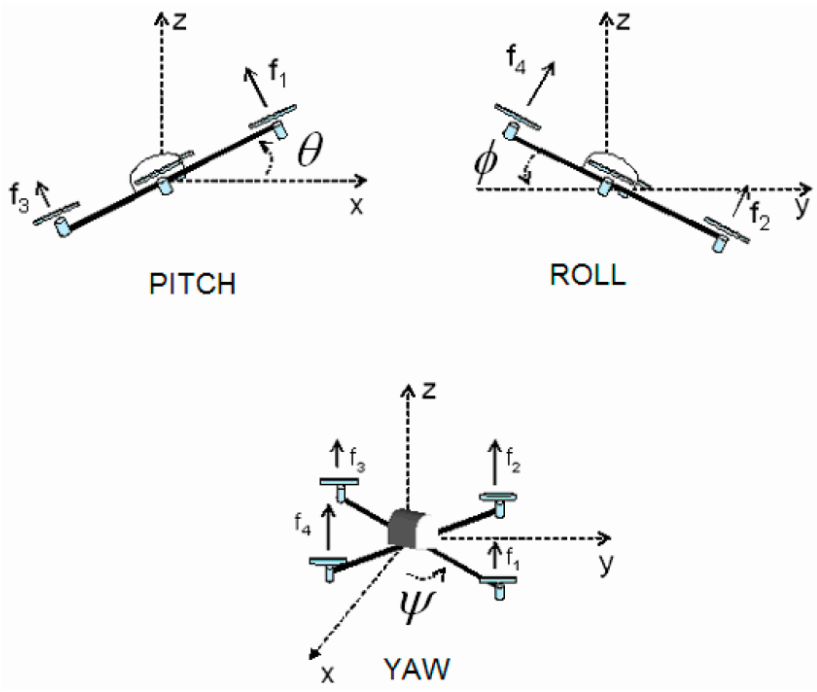

Figura 4 Descripción de los grados de libertad del cuadricóptero
Fuente: (Mayorga Rodríguez, 2009)

2.2.2. Vehículo aéreo no tripulado

Un UAV por sus siglas en inglés, Unnmanned Aerial Vehicle (vehículo aéreo no tripulado), es un sistema capaz de volar sin llevar a bordo un piloto (González, 2017)(Jara-Olmedo, Medina-

Pazmino, Tozer, Aguilar, & Pardo, 2018)(W. G. Aguilar, Angulo, & Pardo, 2017)(Olmedo et al., 2018)(W. G. Aguilar, Cobe, Rodriguez, Salcedo, & Collaguazo, 2018). Ya que, a través de una serie de procesadores incorporados, como los son: sensores, motores, sistemas de comunicación entre estos le dan la ventaja de volar de manera independiente, claro está que estos son controlados desde una estación de mando o también pueden llevar una programación ya establecida (González, 2017).

2.2.3. Clasificación de un UAV

Los UAV's se clasifican tomando en cuenta su peso, altura de operación, duración de vuelo.

Tabla 1
Clasificación de UAV's

Categoría	Peso del UAV	Altitud normal de operación	Radio de la misión	Autonomía de vuelo	Altitud
Micro	< 2 Kg	60 m	5 Km	Pocas horas	Muy poca altitud
Mini	2-20 Kg	900 m	25 Km	Hasta 2 días	Poca altitud
Pequeño	20-50 Kg	1,5 Km	200 Km	Hasta 2 días	Poca altitud
Táctico	150-600 Kg	3 Km	200 Km	Hasta 2 días	Poca altitud
MALE (Mediana altitud, gran resistencia)	> 600 Kg	13,5 Km	200 Km	Días / semanas	Altitud media
HALE (Gran altitud, gran resistencia)	> 600 Kg	20 Km	Ilimitada	Días / semanas	Gran altitud
Combate	> 600 Kg	20 km	Ilimitada	Días / semanas	Gran altitud

Fuente: (Gupta, Ghonge, & Jawandhiya, 2013)

2.3. Formación y representación de la imagen

La formación de la imagen ocurre cuando un sensor (ojo, cámara) registra la radiación (luz) que ha interactuado con ciertos objetos físicos (Sucar & Gómez, 2008).

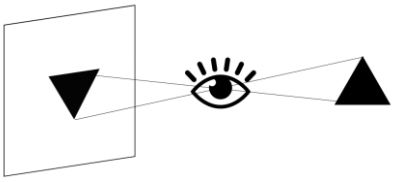

Figura 5 Formación de la imagen
Fuente: (Sucar & Gómez, 2008)

La imagen obtenida por el sensor se puede ver como una función bidimensional, donde el valor de la función corresponde a la intensidad en cada punto de la imagen. Generalmente, se asocia un sistema coordenado (x, y) a la imagen, con el origen en el extremo superior izquierdo (Sucar & Gómez, 2008).

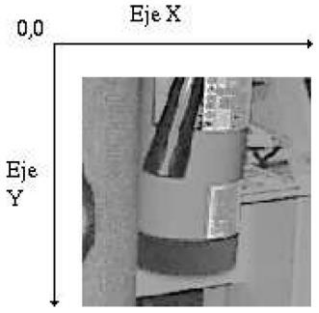

***Figura** 6* Ejemplo de los ejes (x, y) de una imagen
Fuente: (Sucar & Gómez, 2008)

Una función de la imagen es una representación matemática de la imagen. Esta es generalmente una función de dos variables espaciales (x, y).

$$I = f(x, y) \tag{1}$$

2.3.1. Proyección de la imagen

La proyección puntual es la transformación de la imagen que se presenta al pasar a muchos de los dispositivos visuales, incluyendo nuestros ojos y una cámara. La aproximación más simple este fenómeno es el modelo de la "cámara de agujero de alfiler" (pinhole camara) que consiste en proyectar todos los puntos de la imagen a través de un punto al plano de la imagen. De esta forma, un punto (X, Y, Z) en el espacio, se proyecta a un punto (x, y) en el plano de la imagen (Sucar & Gómez, 2008).

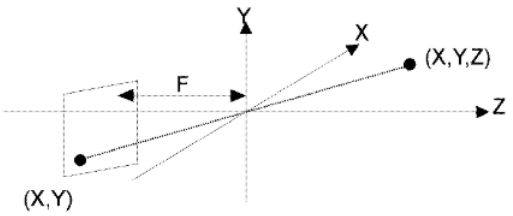

Figura 7 Modelo geométrico de la cámara
Fuente: (Sucar & Gómez, 2008)

En la Figura 7 el plano de la imagen está dado por los ejes x,y,z que esta perpendicularmente al plano x,y de la cámara y además F es la distancia del punto de proyección al plano de la imagen (distancia focal)

2.3.2. Imágenes binoculares

Al proyectarse los objetos, de un espacio tridimensional a una imagen bidimensional se pierde la información de la distancia a la cámara o profundidad (eje Z) de cada punto (W. G. Aguilar, Rodríguez, Álvarez, et al., 2017b)(W. G. Aguilar, Rodríguez, Álvarez, et al., 2017a). Una forma de tratar de recuperar esta información es mediante el uso de dos cámaras, en lo que se conoce como visión estéreo (Sucar & Gómez, 2008).

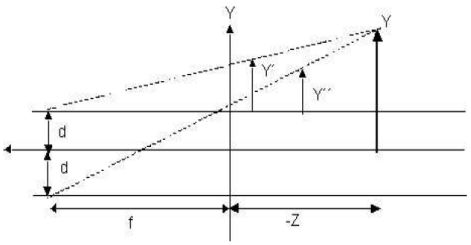

Figura 8 Imágenes binoculares
Fuente: (Sucar & Gómez, 2008)

Si consideramos que se tiene dos cámaras separadas a una distancia conocida 2d, se obtiene dos imágenes de cada punto (X, Y). Utilizando sólo la coordenada Y, el modelo geométrico se puede ver en la Figura 8.

2.3.3. Geometría epipolar

La geometría básica de un sistema de imágenes estéreo se denomina geometría epipolar (W. G. Aguilar & Angulo, 2014a, 2014b, 2016). En esencia, esta geometría combina dos modelos pinhole (uno para cada cámara) y añade dos puntos de especial interés llamados epipolos ver en la Figura 9 (Eduardo, 2014).

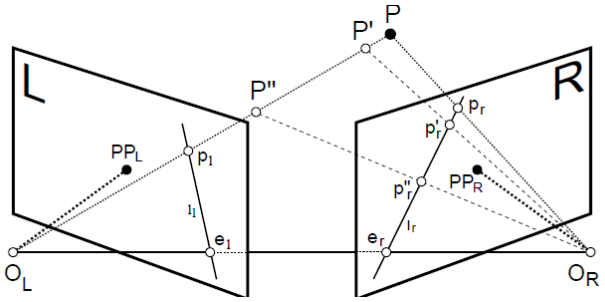

Figura 9 Relación epipolar
Fuente: (Eduardo, 2014)

Cada cámara tiene un centro de proyección asociado, OL y OR, y un plano de proyección correspondiente, L y R. El punto P en el mundo físico tiene una proyección sobre cada uno de los

planos mencionados y las etiqueta como pl y pr, respectiva-mente. Los puntos el y er se denominan epipolos de las cámaras. El epipolo el (resp. er) en el plano de imagen L (resp. R) es la proyección del centro óptico OR (OL) de la segunda (primera) cámara observada por la otra cámara (Eduardo, 2014).

2.3.4. Correspondencia estéreo

Se define como disparidad en el contexto de la correspondencia estéreo al desplazamiento horizontal entre dos píxeles homólogos, uno visto por la cámara izquierda y el otro por la derecha, que son la proyección de un mismo punto 3D (Eduardo, 2014).

Suponiendo que un punto P del mundo físico tiene una proyección visible en cada imagen, pl y pr de coordenadas (xl, yl) y (xr, yr) respectivamente, obteniendo el valor de disparidad como la diferencia.

$$d(x_l) = x_l - x_r \tag{2}$$

La disparidad sólo se puede calcular en la región visual en la que ambas vistas se solapan. Conocidas las coordenadas físicas de las cámaras o los tamaños de los objetos en la escena, podemos inferir medidas de profundidad por medio de la triangulación de la disparidad medida entre los puntos correspondientes en las dos vistas (Eduardo, 2014).

2.4. Deep Learning

El aprendizaje profundo es una forma de aprendizaje automático que permite a las computadoras aprender de la experiencia y comprender el mundo en términos de una jerarquía de conceptos. Debido a que la computadora reúne el conocimiento de la experiencia, no es necesario que un operador de computadora humano formalmente especifique todo el conocimiento que

necesita la computadora. La jerarquía de conceptos permite a la computadora aprender conceptos complicados al construirlos a partir de conceptos más simples; una gráfica de estas jerarquías tendría muchas capas de profundidad (Kim, 2016).

2.5. Redes Neuronales Convolucionales (CNN)

Una red neuronal convolucional es un tipo de Deep Neural Network, es decir, aquella red neuronal que entre la capa de entrada y la de salida existen diversas capas. Estas capas reciben el nombre de capas ocultas ("hidden layers"). La arquitectura más convencional de las CNN consiste en una o más capas convolucionales (que pueden estar conectadas a capas pooling), luego están conectadas a una o más capas totalmente conectadas (fully-connected), que son las mismas que en las redes neuronales multicapa convencionales, y finalmente, a una capa de salida (Camilo, 2018).

2.5.1. Arquitectura de dispositivos de computo (CUDA)

Es una arquitectura de cálculo paralelo de la empresa NVIDIA principalmente aprovecha la gran potencia de la GPU (unidad de procesamiento gráfico) proporcionando un incremento extraordinario del rendimiento del sistema (CUDA, 2018).

2.5.2. Tensorflow

Es una biblioteca de software de código abierto para el cálculo numérico que utiliza grafos de flujo de datos. Los nodos en el grafo representan operaciones matemáticas, mientras que las aristas del grafo representan los conjuntos de datos multidimensionales (tensores) comunicados entre ellos (Camilo, 2018). La arquitectura flexible le permite realizar cálculos en una o más CPU o GPU de un ordenador, servidor o dispositivo móvil desde la misma API. TensorFlow fue desarrollado originalmente por investigadores e ingenieros que trabajan en el equipo Brain de Google dentro de

la organización de investigación de Inteligencia Artificial para realizar investigaciones (Camilo, 2018).

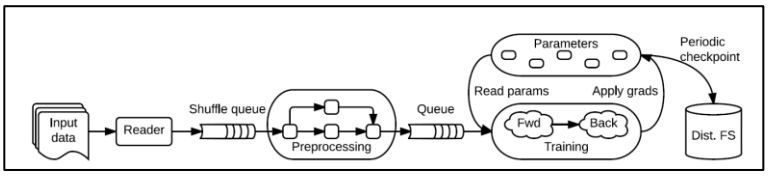

Figura 10 Un diagrama de flujo de datos de TensorFlow
Fuente:(Martín & Barham, 2016)

2.6. Control de un UAV

Basado en estudios anteriores hechos por (Salcedo, 2018) diseñando un controlador por medio de la estimación de movimiento visual del micro-UAV. Utiliza un controlador de tipo PID que como su nombre lo indica, el algoritmo de control PID consiste de tres modos básicos, el proporcional, el integral y el derivativo y cuando se usa esta técnica de control clásico, es necesario decidir cuales modos serán usados (P, I, o D) y especificar los parámetros o ajustes para cada uno de esos modos. Generalmente los algoritmos básicos usados son P, PI o PID (Lorandi, Hermida, Ladrón, & Hernández, 2011).

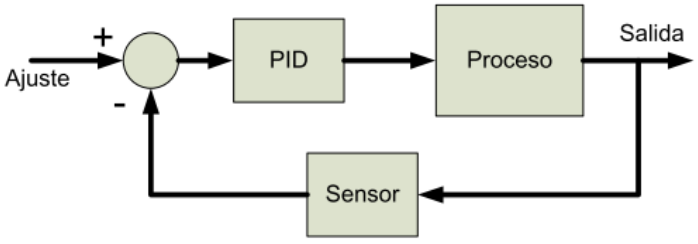

Figura 11 Representación esquemática de un modelo PID
Fuente:(Lorandi et al., 2011)

2.6.1. Característica del controlador

El controlador PID clásico tiene la forma:

$$u = K_p\left(e + \frac{1}{T_i}\int_0^t e\,dt + T_d\frac{de}{dt}\right) \quad (3)$$

En donde "e" es el error entre el valor de referencia y la salida del sistema, "u" es la salida del controlador, "Kp" la ganancia proporcional, "Ti" el tiempo integral, y "Td" el tiempo derivativo.

Tabla 2
Efecto de las constantes de control respecto respuesta de control

	Tr	**Mp**	**Ts**	**ecc**
Kp	Disminuye	Aumenta	Poco Efecto	Disminuye
Ki	Disminuye	Aumenta	Aumenta	Elimina
Kd	Poco Efecto	Disminuye	Disminuye	Poco Efecto

Fuente: (Katsuhiko, 1998)

La acción de control de Kp, Ki, Kd se ve reflejada en la Tabla 2. Los valores de Tr (tiempo de retardo), Mp (sobrepaso máximo), Ts (Tiempo de asentamiento) y e_{ee} (error en estado estable) contrasta con la Figura 12.

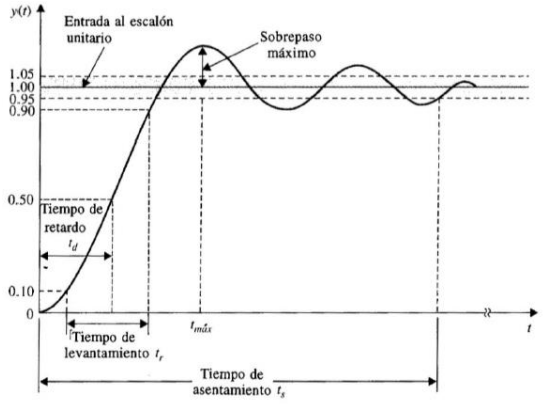

Figura 12 Respuesta de un sistema de segundo orden
Fuente: (Katsuhiko, 1998)

CAPÍTULO III

3. DESCRIPCIÓN GENERAL DEL SISTEMA

3.1. Introducción

En este capítulo especifica los componentes para el desarrollo del proyecto así como también la comunicación para la obtención de las imágenes a bordo del micro-UAV. Además se explica la utilización del controlador "bebop_autonomy*"* para el control de los movimientos básicos del micro-UAV.

3.2. Hardware del Sistema

Para el presente proyecto se utiliza un vehículo aéreo no tripulado del cual se obtiene las imágenes para su posterior procesamiento en una estación en tierra y este finalmente envié las órdenes de control de vuelta al vehículo aéreo no tripulado.

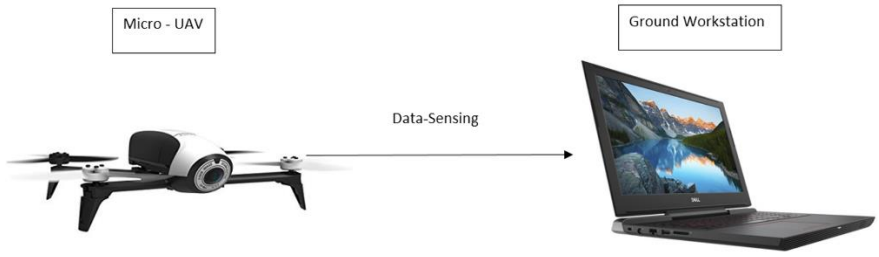

Figura 13 Componentes del hardware del proyecto

3.3. Cuadricóptero: Parrot Bebop 2

A finales de 2015 Parrot lanzó al mercado el cuadricóptero "Bebop 2", cuenta con cuatro motores con configuración de "X", este modelo presenta una mejor autonomía de vuelo que su versión anterior "Bebop 1". Además cuenta con un procesador interno de doble núcleo ARM Cortex A9 que trabaja sobre Linux. Más adelante publicó un (SDK, por sus siglas en inglés)

"Software Development Kit", que posibilitó la creación del driver llamado "bebop_autonomy" el cual nos permite interactuar con el dron a través de Linux, utilizando ROS (Valero, 2017).

Figura 14 Cuadricóptero Bebop 2 de Parrot con su caja original
Fuente: (Parrot SA., 2018)

3.3.1. Especificaciones Técnicas

Tabla 3
Especificaciones Técnicas de Parrot Bebop 2

Especificaciones Técnicas de Parrot Bebop 2	
Estructura	4 motores sin escobillas Outrunner Estructura reforzada de fibra de vidrio (20%) y Grilamid (casco)
Conectividad	Antena Wi-Fi: MIMO a frecuencias 2.4 y 5 GHz Transmisión de potencia de 21dBm Alcance de la señal de hasta 300 m Protocolo de comunicación 802.11 a/b/g/n/ac
Velocidad	Horizontal: 16 m/s - Acenso: 6 m/s
Cámara	CMOS 14 Mpx Estabilización de video digital Video con resolución de 1920 x 1080p a 30 cuadros por segundo Resolución de imagen de foto de 4096 x 3072 Memoria de almacenamiento interno de 8GB
Batería	25 minutos de vuelo con la batería 2700mAh

CONTINÚA

Procesador	Procesador dual-core con GPU quad-core
Sensores	Sensor ultrasónico con una capacidad de hasta 8 metros de altitud Sensor de flujo óptico para detección de movimientos Magnetómetro, Giroscopio, Acelerómetro de 3 ejes
Geo-ubicación	GNSS (GPS+GLONASS)
Dimensiones	38 x 33 x 9 cm
Peso	500 g

Fuente: (Parrot SA., 2018)

3.3.2. Estación en Tierra

Para la estación en tierra se ha elegido un computador portátil con las características suficientes para comunicarse vía Wi-Fi con el micro-UAV, capaz de ejecutar herramientas de software de coste computacional alto, por ende un procesamiento de imagen en tiempo real. Las características principales del computador son:

- Modelo: Dell 7577
- Procesador: Intel Core i7-7700HQ a 2.80 GHz
- Memoria RAM: 16 GB
- Tipo de sistema: Sistema operativo de 64 bits
- Tarjeta de Video: GTX 1060 Max-Q

3.4. Software del Sistema

Se eligió utilizar el sistema operativo Ubuntu 16.04 LTS por ser unos de los sistemas operativos más estables entre las distribuciones de Linux, además de permitir utilizar herramientas de bibliotecas de procesamiento de imagen como son las de Open CV.

3.4.1. Entorno de desarrollo ROS (Robot Operating System)

Mucho se ha hablado de ROS como un sistema operativo pero en realidad no lo es, ya que se instala sobre otro, en este caso particular Linux, y por ello también recibe el nombre de Meta-Sistema Operativo (Valero, 2017).

Este sistema operativo cuenta con un conjunto de librerías y herramientas que simplifican la tarea de realizar un software robusto y complejo para robótica. ROS proporciona servicios tales como abstracción del hardware, comunicación a través de mensajes entre procesos, mantenimiento de paquetes, control de dispositivos de bajo nivel (Valero, 2017).

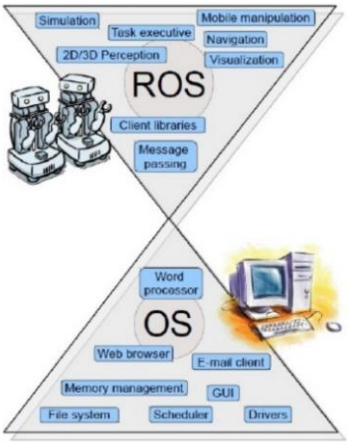

Figura 15 Esquema de ROS como Meta-Sistema Operativo
Fuente: (Valero, 2017)

3.4.1.1. Herramientas de ROS

Los conceptos básicos de ROS son nodos, maestro, servidor de parámetros, mensajes, servicios, tópicos y paquetes, todos estos los veremos a continuación (ROS, 2014):

Maestro (Master): Es el nodo principal quien proporciona el registro de nombres y la búsqueda en el resto de la Gráfica computacional.

Nodos (Node): Los nodos son procesos encargados de realizar el cálculo computacional.

Tópicos (Topic): Son canales de información entre los nodos, un nodo puede recibir o enviar mensajes por medio de los tópicos a lo que se llama subscribirse y publicar.

Mensajes (Messages): Es una estructura de datos, la cual puede admitir algún tipo de dato como son (entero, punto flotante, booleano, etc.).

El protocolo más común usado en un ROS se llama TCPROS, que usa sockets TCP / IP estándar (ROS, 2014).

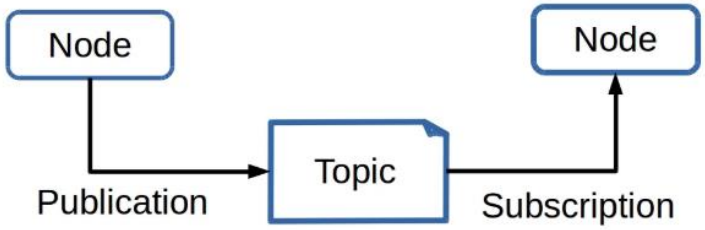

Figura 16 Esquema de flujo de información en ROS
Fuente: (Valero, 2017)

3.4.1.2. Driver bebop_autonomy

Este driver "bebop_autonomy" está desarrollado en ROS para los modelos de drones "Bebop 1" y "Bebop 2", siendo posible por la SDK ARDroneSDK3 proporcionada por Parrot. Siendo desarrollado por el "autonomy lab" en la universidad Simon Fraser por Mani Monajjemi además de otros colaboradores (Mani Monajjemi, 2015). En la misma página se puede encontrar los pasos a seguir para la correcta instalación del driver para la comunicación con "bebop 1" o "bebop 2".

Para la ejecución del driver se utiliza el archivo "*bebop_node.launch"*, este por tanto hemos de acceder a su carpeta utilizando el comando "cd" seguido de la ruta completa:

```
$ roslaunch bebop_driver bebop_node.launch
```

Una vez ejecutado el comando se generan una serie de tópicos que permiten conocer y manejar todas las variables del micro-UAV.

Tabla 4
Tópicos principales de "bebop_autonomy"

TÓPICO	MENSAJE	ACCIÓN
land	std_msgs/Empty	Aterrizar
takeoff	std_msgs/Empty	Despegar
reset	std_msgs/Empty	Detener los motores
cmd_vel	geometry_msgs/Twist	Movimientos x,y,z del dron
camera_control	geometry_msgs/Twist	Movimiento de la cámara virtual
image_raw	sensor_msgs/Image	Transmisión de imágenes

Fuente:(Mani Monajjemi, 2015)

Algunos parámetros están previamente establecidos, pero son configurables como por ejemplo la IP de comunicación "192.168.42.1". Así como también el ángulo de inclinación máxima de "pitch" y "roll" de "bebop" que es de 20 grados para ambos, accediendo y siendo capaz de cambiarlo con el parámetro "PilotingSettingsMaxTiltCurrent". Para este proyecto la mayoría de estos parámetros no han sido cambiados y se ha dejado con su valor por defecto ya que se realizará a una altura constante de 1 metro y a una velocidad a de 5 km/h que es la velocidad aproximada promedio de caminata de un hombre promedio.

3.4.1.2.1. Obtención de imágenes bebop 2

La obtención de imágenes de "bebop 2" se publica en el tópico "image_raw" como mensaje de tipo "sensor_msgs/Image". La resolución de imágenes transmitidas por "bebop" son de 856 x 480 pixeles y a una taza de transmisión aproximada de 30 Hz (Mani Monajjemi, 2015).

3.4.1.2.2. Subscriptor de Imágenes

Para realizar este proceso es necesario una conversión de mensaje de ROS a tipo de imagen "opencv", ya que ROS publica las imágenes como un tipo de mensaje como se observa gráficamente en la Figura 17.

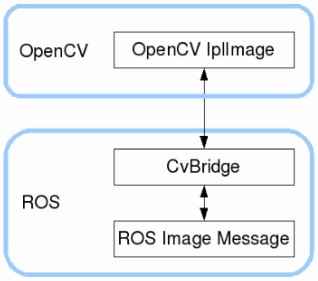

Figura 17 Conversión de mensaje de ROS a Imagen Opencv
Fuente: (ROS, 2018)

El programa se lo puede representar por medió el diagrama de flujo de la Figura 18, que muetra una ejecución contante de la petición de imagen a "bebop 2".

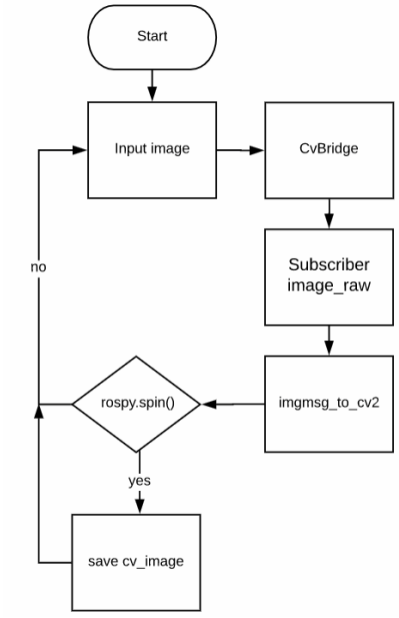

Figura 18 Diagrama de flujo de proceso de conversión de imagen ROS a open CV
Fuente: (W. G. Aguilar, Quisaguano, Alvarez, Pardo, & Proaño, 2018)

Para el proceso en términos algorítmicos de obtención de imágenes primero se debe inicializar el nodo con el comando “rospy.init_node” el cual realizará la petición de lectura, hacia el nodo que “bebop 2” está publicando de manera constante “image_raw”.

```
def main():
  bridge = CvBridge()
  rospy.init_node('Imagen', anonymous=True)
  image_sub = rospy.Subscriber('bebop/image_raw',Image,callback)
  rospy.spin()
```

Figura 19 Subscripción al nodo “image_raw”

Además como observamos en la Figura 19 con el comando “rospy.Subscriber” se realiza la subscripción al nodo y se guarda el mensaje en “Image” llamando a la función “callback” que observamos en la Figura 20, además cabe recalcar que la función “rospy.spin( )”, lo único que hace es evitar que python salga hasta que el nodo se detenga. La imagen obtenida se puede tratar normalmente como una imagen en open cv.

```
def callback(data):
  cv_image = bridge.imgmsg_to_cv2(data, "bgr8")
  proceso(cv_image)
```

Figura 20 Función de conversión de mensaje ROS a imagen Opencv

Cabe mencionar en la todas las dependencias necesarias para la utilización de las funciones anteriores.

```
import roslib
import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
```

Figura 21 Librerías necesarias para la subscripción de Imagen

Realizando mediciones se definió que el campo de visión de esta cámara virtual es de 80 grados (horizontal) y 50 grados (vertical) aproximadamente corroboradas analíticamente de la Figura 23. Primero se tomó los valores reales de una imagen obtenida del "bebop 2" como se observa en la Figura 22, tomando en cuenta una distancia del piso de 1 metro, y una distancia 1.5 metros hacia la persona de prueba, posteriormente se calculó los ángulos del campo de visón.

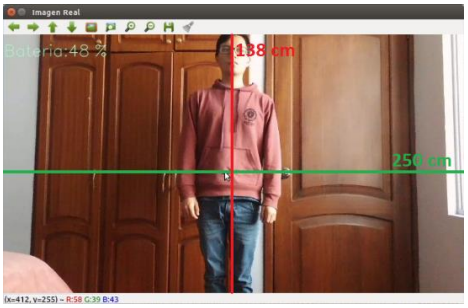

Figura 22 Distancias medidas a 1.5 metros de distancia del "bebop 2"

Para el cálculo de los ángulos 2α (campo de visión vertical) y 2β (campo de visión Horizontal) se tomó en cuenta la ecuación (4) y la ecuación (5) respectivamente.

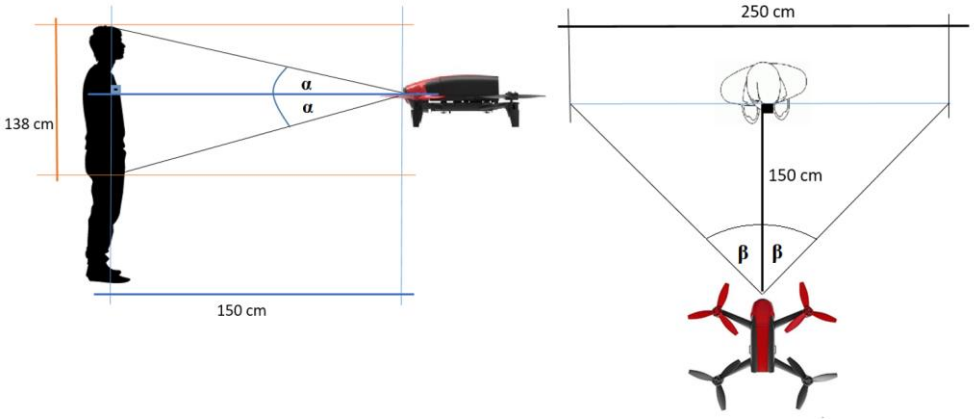

Figura 23 Análisis de las distancias obtenidas del bebop2,

(izquierda) vista lateral - (derecha) vista cenital

$$\tan(\alpha) = \left(\frac{\frac{138}{2}}{150}\right) \quad (4)$$

$$\tan(\beta) = \left(\frac{\frac{250}{2}}{150}\right) \quad (5)$$

3.4.1.2.3. Control de los movimientos pitch, roll, yaw de bebop 2

La tarea de controlar los movimientos principales del "bebop 2" se resume en la Tabla 4. Una vez se haya publicado la acción de despegue de "bebop 2". Se puede utilizar los parámetros de movimiento del mismo que se muestra en la Tabla 5.

Tabla 5
Parámetros del tipo de mensaje "geometry_msgs/Twist" del tópico "cmd_vel"

Parámetro	Valor	Acción
linear.x	+	Trasladar hacia delante
	-	Trasladar hacia atrás
linear.y	+	Trasladar a la izquierda
	-	Trasladar a la derecha
linear.z	+	Ascender
	-	Descender
angular.z	+	Girar en sentido horario
	-	Girar en sentido anti-horario

Fuente: (Mani Monajjemi, 2015)

El rango aceptable para todos los campos es [-1..1] en pasos de 0.1, siendo 0 la velocidad mínima y 1 la velocidad máxima. "Bebop 2" ejecuta el último comando recibido siempre que el controlador se esté ejecutando. Para hacer que "bebop 2" mantenga su posición actual, debe

publicar un mensaje con todos los campos establecidos en cero en cmd_vel (Mani Monajjemi, 2015).

Figura 24 Esquema de desplazamiento de bebop 2 con velocidades positivas
Fuente: (Valero, 2017)

Para entender de mejor manera la Figura 24, el valor del parámetro “linear.x” se refiere a los ángulos de inclinación de “pitch”, mientras que “linear.y” se refiere al ángulo de “roll” vistos en el ejemplo de la Figura 4, con esto es posible controlar tanto sus aceleraciones delanteras y como laterales.

Por otra parte para el parámetro “linear.z” controla la velocidad vertical del “bebop 2”. Para este proyecto este valor siempre permanece en 0 al requerir una seguimiento y evasión en el plano paralelo al eje (x, y), es decir paralelo al suelo. Finalmente el parámetro “angular.z” hace referencia al ángulo de rotación yaw como ejemplo en la Figura 4, es decir que controla la velocidad de rotación en eje z de “bebop 2”.

A continuación se muestra un ejemplo en lenguaje de programación de “Python” de la publicación de los parámetros anteriormente descritos.

Primero se debe importar todas las librerías necesarias que se muestran en la Figura 25.

```
import roslib
import rospy
import std_msgs.msg
import geometry_msgs.msg
from std_msgs.msg import Empty
from geometry_msgs.msg import Twist
```

Figura 25 Librerías de mensajes para movimiento micro-UAV

Al igual que en el ejemplo anterior se inicializa el nodo, además se define los mensajes a publicar, en este caso mensaje de velocidad “cmd_vel”, así como también “takeoff”, “land” para despegar y aterrizar respectivamente como se observa en la Figura 26.

```
def main():
     rospy.init_node('movimientos')
      pub = rospy.Publisher('/bebop/cmd_vel', Twist, queue_size = 1)
      pub1 = rospy.Publisher('/bebop/takeoff', Empty, queue_size = 1)
      pub2 = rospy.Publisher('/bebop/land', Empty, queue_size = 1)
```

Figura 26 Definición de los nodos a publicar para movimiento micro-UAV

Finalmente para hacer efectivo la publicación de estos parámetros se utiliza la función “publish()” de ROS, así como “Twist()”, para el manejo de este tipo de mensajes. Hay que mencionar que el la Figura 27 se hace referencia a “speed” y “turn” que son simples variables.

```
pub1.publish()
pub2.publish()

twist = Twist()
twist.linear.x = x*speed
twist.linear.y = y*speed
twist.linear.z = z*speed

twist.angular.x = 0
twist.angular.y = 0
twist.angular.z = th*turn
pub.publish(twist)
```

Figura 27 Publicación de mensajes para movimientos de micro-UAV

CAPÍTULO V

4. DETECCIÓN DE PERSONA Y ESTIMACIÓN DE PROFUNDIDAD MONOCULAR

4.1. Introducción

En este capítulo se muestra en primera instancia el proceso de entrenamiento de la red neuronal con la adquisición de las imágenes, su posterior etiquetado para la detección de persona. Además se realiza un análisis de la estimación de profundidad con una sola imagen a pesar de la ausencia de datos de profundidad de la verdad del terreno.

4.2. Detección de persona

Para realizar la detección de persona, primero consistirá en el entrenamiento basado en redes neuronales convolucionales de la API de detección de objetos de tensorflow, esto en un marco de código abierto que facilita la construcción, entrenamiento y despliegue de modelos de detección de objetos (Pkulzc, 2018).

4.2.1. Modelo Pre-entrenado

(Wu & Rathod, 2018) Proporciona una colección de modelos de detección pre-entrenados en el conjunto de datos “COCO” los cuales presentan resultados con mayor velocidad de respuesta.

Como se aprecia en la Tabla 6 se enumera los modelos más rápidos pre-entrenados para la detección de objetos.

Tabla 6
Modelos Pre-entrenado de COCO

Nombre del modelo	Velocidad (ms)	COCO (mAP)	Salida
ssd_mobilenet_v1_coco	30	21	Cuadros
ssd_mobilenet_v1_0.75_depth_coco	26	18	Cuadros
ssd_mobilenet_v1_quantized_coco	29	18	Cuadros
ssd_mobilenet_v1_0.75_depth_quantized_coco	29	16	Cuadros
ssd_mobilenet_v1_ppn_coco	26	20	Cuadros
ssd_mobilenet_v1_fpn_coco	56	32	Cuadros
ssd_resnet_50_fpn_coco	76	35	Cuadros
ssd_mobilenet_v2_coco	31	22	Cuadros
ssd_mobilenet_v2_quantized_coco	29	22	Cuadros
ssdlite_mobilenet_v2_coco	27	22	Cuadros

Fuente: (Wu & Rathod, 2018)

Cabe mencionar que los modelos de la Tabla 6 muestran tiempos de ejecución utilizando una tarjeta Nvidia GeForce GTX TITAN X por lo que deben tratarse como tiempos relativos.

El mejor rendimiento mostrado es de “ssd_mobilenet_v1_coco” ya que presenta una velocidad de ejecución aceptable con un alto grado de mAP (media de precisión promedio) ya que aquí, cuanto más alto es mejor. En el sitio web de (Wu & Rathod, 2018) se puede descargar los modelos.

Este modelo incluye:

- Un gráfico proto (graph.pbtxt) donde se encuentran las etiquetas de entrenamiento.
- Un punto de control (model.ckpt.data, model.ckpt.index, model.ckpt.meta).
- Un “frozen_graph” con pesos integrados en el gráfico como constantes (frozen_inference_graph.pb) que se usarán para la inferencia de las cajas de detección.

Tras descargarse el modelo pre-entrenado y la instalación de la API de “tensorflow”, se puede realizar pruebas de con la detección de objetos y observar sus ponderaciones como se observa en la Figura 28.

Figura 28 Detección de objetos con la red pre-entrenada conjunto datos COCO

4.2.2. Modelo SSD (Single Shot MultiBox Detector)

El enfoque de SSD se basa en una red convolucional de avance que produce una colección de cuadros delimitadores de tamaño fijo y puntúa la presencia de instancias de clase de objeto en esos cuadros (W. Liu et al., 2016).

Las primeras capas de la red se basan en una arquitectura estándar VGG-16 utilizada para la clasificación de imágenes de alta calidad como se obsreva en la Figura 30. Luego se agrega una estructura auxiliar a la red para producir detecciones con las siguientes características (W. Liu et al., 2016):

El modelo SSD solo necesita una imagen de entrada y cuadros de verdad (“ground truth”) para cada objeto durante el entrenamiento. De manera convolucional, se evalúa un pequeño conjunto de cuadros predeterminados con diferentes relaciones de aspecto en cada ubicación en varios mapas de características con diferentes escalas (por ejemplo, 8×8 o 4×4). Para cada cuadro predeterminado, se predice los conjuntos de formas como las diferencias para todas las categorías de objetos. En el momento de la capacitación, primero se coinciden los cuadros predeterminados con los cuadros de verdad (“ground truth”). En la Figura 29, se encaja dos cuadros predeterminados

con el gato y una con el perro, que se consideran positivas y el resto como negativas. La pérdida del modelo es una suma ponderada entre la pérdida de localización (por ejemplo, Smooth L1) y la pérdida de confianza (por ejemplo, Softmax) (W. Liu et al., 2016).

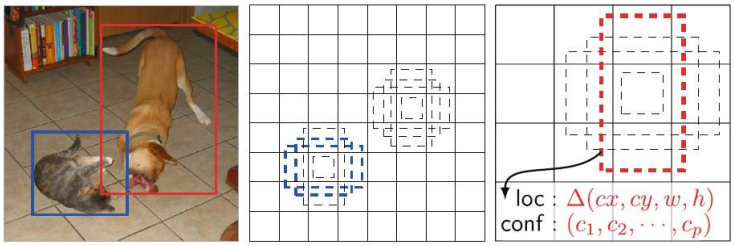

Figura 29 Imagen con cuadros (izquierda) - 8x8 mapa de características (centro) -4x4 mapa de características (derecha)
Fuente: (W. Liu et al., 2016)

Una característica importante de esta red es el entrenamiento que recibe. Para realizar un buen entrenamiento es necesario una gran cantidad de imágenes con el fin de obtener buenos resultados (Gamino & Sánchez Jonathan, 2018).

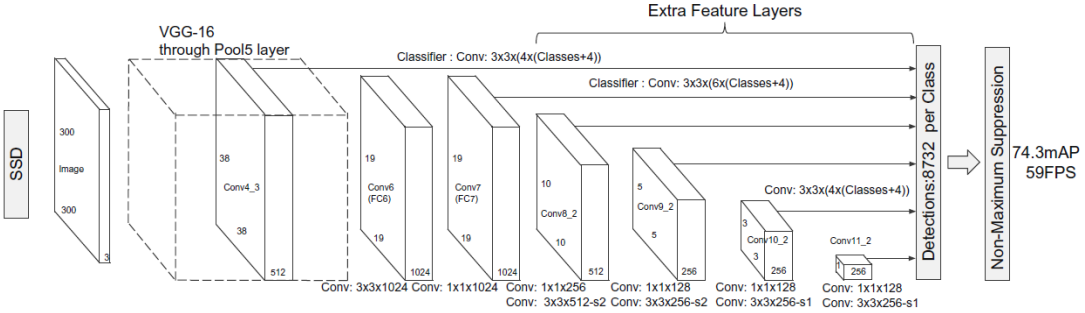

Figura 30 Modelo SSD con sus capas características
Fuente: (W. Liu et al., 2016)

4.2.3. Conjunto de datos de entrenamiento

Para la base de imágenes de entrenamiento de la persona a seguir, se las obtiene directamente de “bebop 2” con las dimensiones 856 x 480 de ancho y de alto respectivamente. La recomendación para el número de imágenes de entrenamiento es de entre 100 y 500 imágenes. En este caso particular se eligió una base de 1024 imágenes, variando la exposición de la luminosidad, la postura y diferentes distancias de la persona como se puede observar en Figura 31.

Figura 31 Etiquetado de persona en diferentes escenarios

4.2.3.1. Etiquetado de persona

Se necesita etiquetar los cuadros que se utilizarán para el entrenamiento, para esto idealmente se debe utiliza un programa que facilita este paso de etiquetado de imágenes. El programa que se eligió es “Labeling” este programa se puede instalar siguiendo la guía de su creador Tzutalin en Figura 32 (Tzutalin, 2015).

```
sudo apt-get install pyqt5-dev-tools
sudo pip3 install lxml
make qt5py3
python3 labelImg.py
```

Figura 32 Comando de instalación de Labeling

El programa es intuitivo y fácil de utilizar mostrado en Figura 33.

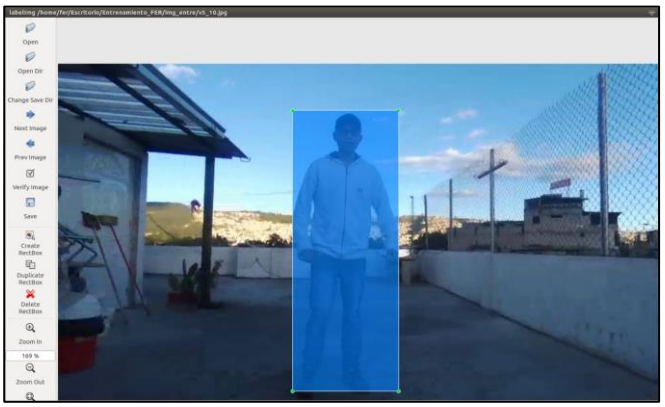

Figura 33 Entorno de programa Labeling

Este programa lo que hace es crear automáticamente un archivo XML que describe los objetos en que se enmarcaron en las imágenes, es decir las coordenadas del recuadro dentro de la imagen. Esto se debe hacer para todas las imágenes de entrenamiento una vez acabado con todas las imágenes se tendrá algo como la Figura 34.

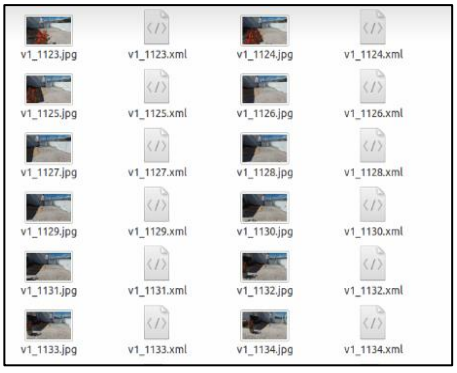

Figura 34 Conjunto de imágenes etiquetadas

4.2.3.2. Creación de TFRecord

Se tiene que convertir los archivos XML en archivo CSV singulares que luego se pueden convertir en archivos TFRecord para que sean las entradas de la red neuronal, para esto se necesita

dividir las imágenes etiquetadas con sus archivos XML en dos partes, la primera aproximadamente un 10% de todas las imágenes dentro de directorio una carpeta con el nombre de “test” (como ejemplo) y la parte restante en el directorio pero en otra carpeta con el nombre de “train” (como ejemplo).

Una vez hecho esta división de las imágenes se procede a convertir los archivos XML a CSV con ayuda del programa “xml_to_csv.py” que se puede encontrar en (Tran, 2017).

Para generar el archivo TFRecord de igual manera se utiliza el programa “generate_tfrecord.py” se puede encontrar en (Tran, 2017) y se ejecuta con la dirección de los archivos CSV de “test” y “train”. Como ejemplo en la Figura 35.

```
python generate_tfrecord.py --csv_input=data/train_labels.csv  --output_path=train.record
python generate_tfrecord.py --csv_input=data/test_labels.csv  --output_path=test.record
```

Figura 35 Como generar los archivos TFRecord

Se obtiene los archivos en la Figura 36 necesarios que servirán como entrada para la red convolucional del entrenamiento de la persona.

Figura 36 Archivos CSV a TFRecord generado

4.2.4. Configuración de entrenamiento

Los archivos de configuración se pueden encontrar en la dirección de API de tensorflow ejemplo (models/research/object_detection/samples/configs/) se puede tomar unos de esos

archivos y modificar la dirección de en la que se encuentra nuestro modelo pre-entrenado, el número de clases es decir el número de objetos nuevos que se detectará en este caso del proyecto una persona, además de tamaño de lote (“batch”) para el ciclo de evaluación, recomendable que se lo fije en 10 todo dependerá del tamaño de la memoria de su tarjeta de video.

El código de entrenamiento se utilizó el mismo de la API de tensoflow desarrollado en Python “train.py” con esta puede entrenar la red neuronal. Utilizando el comando de la

```
python train.py --logtostderr --train_dir=training/ --
pipeline_config_path=training/ssd_mobilenet_v1_persona.config
```

Figura 37 Ejecución programa de entrenamiento

Como resultado se observará el número de paso que está tomando el entrenamiento como el tiempo que ha tomado realizar esa operación.

```
2018-11-21 21:58:31.314421: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1
120] Creating TensorFlow device (/device:GPU:0) -> (device: 0, name: GeForce GTX
 1060 with Max-Q Design, pci bus id: 0000:01:00.0, compute capability: 6.1)
INFO:tensorflow:Restoring parameters from training/model.ckpt-89657
INFO:tensorflow:Starting Session.
INFO:tensorflow:Saving checkpoint to path training/model.ckpt
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 89658.
INFO:tensorflow:global step 89658: loss = 0.8426 (16.100 sec/step)
INFO:tensorflow:global step 89659: loss = 0.5348 (0.693 sec/step)
INFO:tensorflow:global step 89660: loss = 0.7796 (0.736 sec/step)
INFO:tensorflow:global step 89661: loss = 1.0223 (0.878 sec/step)
INFO:tensorflow:global step 89662: loss = 0.8167 (0.703 sec/step)
INFO:tensorflow:global step 89663: loss = 0.7788 (0.694 sec/step)
INFO:tensorflow:global step 89664: loss = 1.2244 (0.694 sec/step)
INFO:tensorflow:global step 89665: loss = 0.8249 (0.699 sec/step)
INFO:tensorflow:global step 89666: loss = 0.9838 (0.687 sec/step)
INFO:tensorflow:global step 89667: loss = 1.3800 (0.684 sec/step)
INFO:tensorflow:global step 89668: loss = 0.9655 (0.681 sec/step)
INFO:tensorflow:global step 89669: loss = 0.5451 (0.694 sec/step)
INFO:tensorflow:global step 89670: loss = 0.7271 (0.698 sec/step)
```

Figura 38 Proceso de entrenamiento de la red

En cualquier punto del entrenamiento se puede detener ya que existen puntos de control donde se guarda en entrenamiento. Utilizando la herramienta “Tensorboard” se puede observar en tiempo real o una vez terminado el entrenamiento el proceso, el tiempo y el valor que están tomando los valores de pérdida del entrenamiento. En la Figura 39 se observa como la gráfica tiende a cero,

inicialmente teniendo una pérdida total de pérdida de 4 y finalmente llegando a 0.953 entre más pasos realice más tenderá a cero. Finalmente se realizó un entrenamiento con 200 mil pasos. Con un tiempo aproximado de 3 días de entrenamiento.

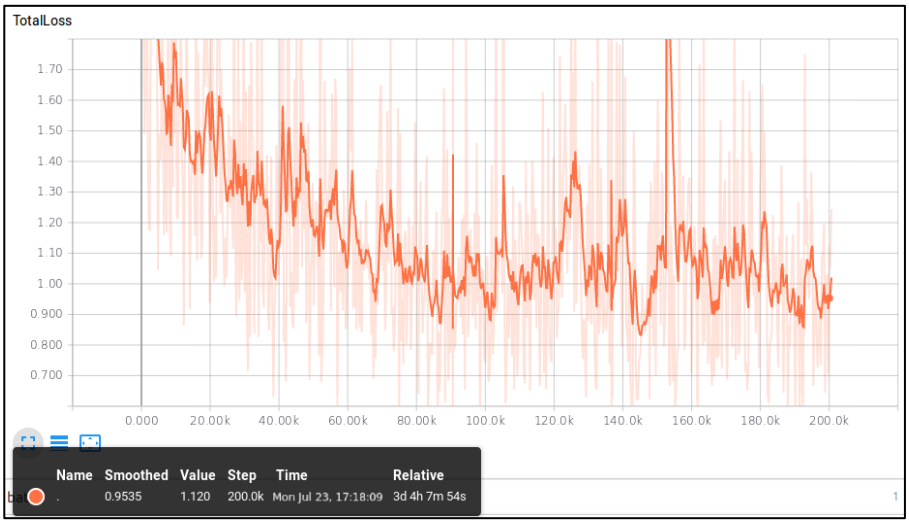

Figura 39 Gráfico en Tensorboard total de pérdidas

4.2.5. Prueba de la red neuronal

En el directorio de “tensorflow” cuenta con el programa “export_inference_graph” que ayuda a exportar el gráfico de inferencia con los pesos del nuevo entrenamiento ahora como se observa en Figura 40 se ingresa el punto de control del modelo en este caso particular “model.ckpt-20591” así como el nombre de la carpeta que se desea guardar el gráfico de inferencia. Hay de recordar que el “pipeline.config” es la configuración de entrenamiento de nuestra red neuronal.

```
python export_inference_graph.py
    --input_type image_tensor
    --pipeline_config_path training/pipeline.config
    --trained_checkpoint_prefix training/model.ckpt-20591
    --output_directory person_inference_graph
```

Figura 40 Comando de exportación de gráfico de inferencia

Para las pruebas de detección de persona se probó con diferentes imágenes, diferentes niveles de iluminación así como también con escenarios que no se encontraban dentro de las imágenes de entrenamiento. Al encontrarse con falsos positivos se procedió a tomar a más imágenes de entrenamiento en esos escenarios. Hasta que se obtenga una detección de persona más robusta como se observa en la Figura 42 un escenario bastante complicado para la detección. Además procedió a reducir a una tolerancia de 80% de certeza en la detección para que sea capaz de detectar a la persona hasta en situaciones con muy baja iluminación.

Figura 41 Prueba de la entrenamiento en escenarios diferentes

El proceso de entrenamiento se puede retomar y agregar nuevas imagen de entrenamiento ya que tensorflow genera cada cierto tiempo genera puntos de control.

Figura 42 Prueba del entrenamiento agregando nuevas imágenes para entrenar

4.3. Estimación de profundidad monocular

La implementación de Tensorflow de la predicción de profundidad de imagen única sin supervisión mediante una red neuronal convolucional (Clément, 2018).

La utilización de una sola cámara para generar un mapa de profundidad es lo que tiene en el estudio de (Godard et al., 2016). Para entender de mejor manera a continuación se explicará la base del algoritmo. La base de este estudio es la verificación de coherencia izquierda-derecha para mejorar la calidad de la imagen de profundidad.

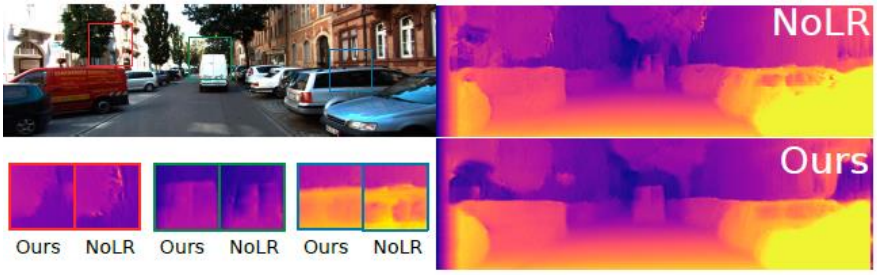

Figura 43 Comparación del método con y sin consistencia izquierda-derecha
Fuente: (Godard et al., 2016)

4.3.1. Análisis de la estimación de profundidad monocular

Dada la posición de los ojos en los humanos y la forma de moverlos, las imágenes recibidas en cada ojo son prácticamente las mismas, con una diferencia en la posición relativa de los objetos. Estas diferencias relativas en la posición en cada imagen (la disparidad), tienen una relación directa con la distancia (profundidad) (Lecumberry, 2005).

Empezando por una imagen I, El objetivo es obtener una función que demuestre predecir la profundidad de cada píxel P como en la Figura 43 y puede predecir la profundidad de cada escena por píxel $\hat{d} = f(I)$. Para el entrenamiento se necesita dos puntos de vista O_L y O_R específicamente, acceder a dos imágenes I^l(izquierda) y I^r (derecha), correspondientes a las imágenes de color izquierda y derecha de un par estéreo calibrado, capturadas en el mismo momento. La disparidad es la diferencia en las coordenadas horizontales de los puntos p_L y p_R es la $d = x_L - x_R$ que corresponde disparidad, un valor escalar por píxel. En lugar de intentar predecir directamente la

profundidad, se intenta encontrar el campo de correspondencia denso d^r que, cuando se aplica a la imagen de la izquierda, nos permitiría reconstruir la imagen de la derecha.

Al referirse a la imagen reconstruida $I^l(d^r)$ como $\widetilde{I^r}$. Del mismo modo, también se puede estimar la imagen de la izquierda dada la de la derecha, $\widetilde{I^l} = I^r(d^l)$ (Godard et al., 2016). A nivel de píxel por similitud entre triángulos. PO_LO_R , $p_Ro_RO_R$ y $p_Lo_LO_L$ se obtiene la ecuación (6):

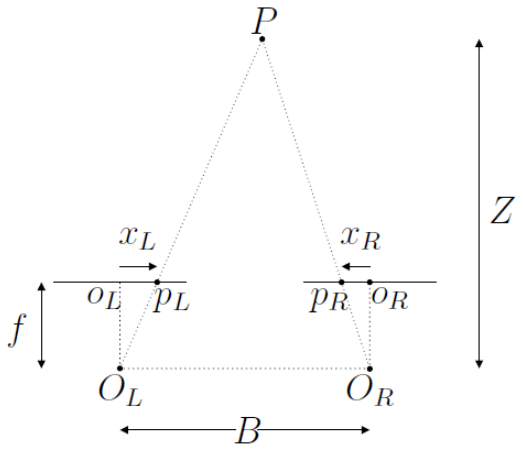

Figura 44 Relación geométrica entre los parámetros del par estéreo para obtener la profundidad Z de la disparidad d

Para obtener la función de distancia propuesta, se tiene la distancia de línea de base B entre las cámaras y la distancia focal de la cámara f, luego se puede recuperar de forma trivial la profundidad $\hat{d}$ de la disparidad predicha.

$$\hat{d} = fB/d \tag{6}$$

4.3.2. Estimación de la red neuronal

La idea clave de este método es poder inferir simultáneamente ambas disparidades (de izquierda a derecha) y de (derecha a izquierda), utilizando solo la imagen de entrada de la izquierda.(Godard et al., 2016).

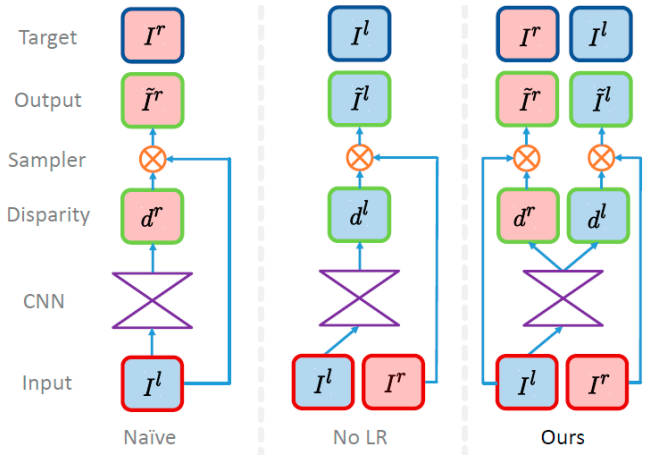

Figura 45 Estrategias de muestro para el mapeo hacia atrás
Fuente: (Godard et al., 2016)

4.3.2.1. Pérdidas en el entrenamiento

Definiendo perdidas como C_s en cada escala de salida, formando la pérdida total como la suma de $C = \sum_{s=1}^{4} C_s$, teniendo una combinación (Godard et al., 2016).

$$Cs = \alpha_{ap}(C_{ap}^{l} + C_{arp}^{r}) + \alpha_{ds}(C_{ds}^{l} + C_{ds}^{r}) + \alpha_{lr}(C_{lr}^{l} + C_{lr}^{r}) \quad (7)$$

Donde

C_{ap}: anima a la imagen reconstruida a aparecer simular a la entrada de entrenamiento.

$$C_{ap}^{l} = \frac{1}{N}\sum_{i,j} \alpha \frac{1 - SSIM\left(I_{ij}^{l}, \tilde{I}_{ij}^{l}\right)}{2} + (1-\alpha)\left\|I_{ij}^{l} - \tilde{I}_{ij}^{l}\right\| \quad (8)$$

C_{ds}: hace cumplir las disparidades suaves.

$$C_{ds}^{l} = \frac{1}{N}\sum_{i,j} \left|\partial_x d_{ij}^{l}\right| e^{-\left\|\partial_x I_{ij}^{l}\right\|} + \left|\partial_y d_{ij}^{l}\right| e^{-\left\|\partial_y I_{ij}^{l}\right\|} \quad (9)$$

C_{lr} : define que las disparidades izquierda y derecha sean consistentes.

$$C_{lr}^{l} = \frac{1}{N}\sum_{i,j}\left|d_{ij}^{l} - d_{ij+d_{ij}^{l}}^{T}\right| \quad (10)$$

4.3.3. Modelo pre-entrenados de estimación de profundidad

Los modelos pre-entrenado se los puede descargar de (Clément, 2018). Compuesta por 697 imágenes según (Eigen et al., 2014) cubriendo un total de 29 escenas. Las 32 escenas restantes contienen 23488 imágenes de las cuales se guarda 600 para el entrenamiento (Garg, Kumar, Carneiro, & Reid, 2016).

En capítulo de resultados se analiza estos modelos a diferentes distancias.

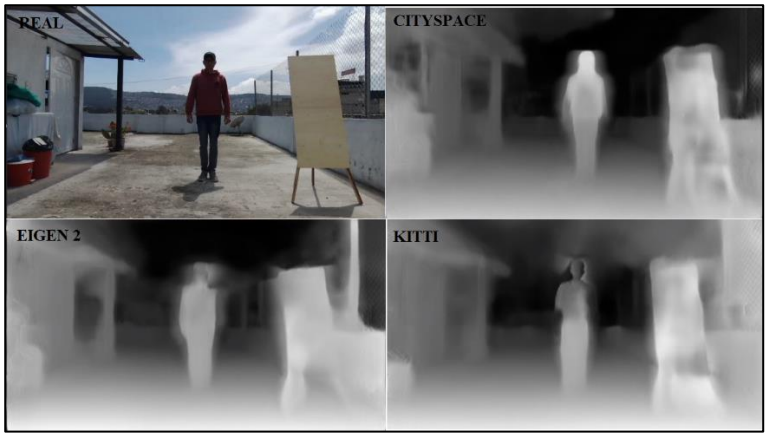

Figura 46 Comparación entre modelos

CAPÍTULO V

5. IMPLEMENTACIÓN DE ALGORITMOS EN MICRO-UAV

5.1. Introducción

En este capítulo se detalla la obtención del modelo matemático para el movimiento en "yaw" del micro-UAV, así como también su implementación para el seguimiento de persona y posteriormente para la evasión de los obstáculos.

5.2. Estimación del modelo de movimiento de micro-UAV

Dentro del estado del arte el modelo matemático de un cuadricóptero se lo ha desarrollado por medio de estimación de movimiento, con sensores como son acelerómetros (W. G. Aguilar, Casaliglla, & Polit, 2017)(W. G. Aguilar, Costa-castelló, & Angulo, 2014), en el caso particular de "Bebop 2" tiene una restricción de envió de datos de estado de 5Hz debido a esto sacar el modelo matemático de esta manera sería muy impreciso.

También se tiene por otro lado la estimación del modelo matemático del análisis de un sistema de masa sin amortiguamiento, el cual se puede controlar por medio de la entrada, teniendo como resultado la función de transferencia (Morales & Paucar, 2017).

La opción que se eligió es modelo matemático del micro-UAV por medio de estimación de movimiento obtenidas por su cámara abordo, al igual que en (Salcedo, 2018) se procede a obtener los desplazamientos a través de imágenes proporcionadas por el micro-UAV ya que esta imágenes se obtienen a una taza 30Hz realmente aceptable para un modelamiento correcto, la diferencia entre su análisis y el nuestro, es en la utilización del método de flujo óptico de Lucas Kanade el cual permite hacer un seguimiento de pixeles dentro de una secuencia de imágenes. Primero se

selecciona el objeto a seguir en este caso por su forma y tamaño se elige una caja como muestra la Figura 47.

Hay que guardar los valores de las coordenadas de la caja es decir (y_min, y_max, x_min, x_max) como se observa en la Figura 47 estos valores de coordenadas se genera una nube de pixeles.

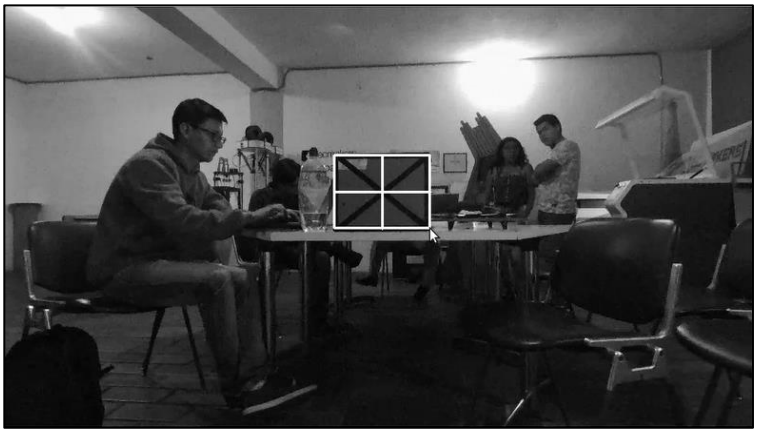

Figura 47 Selección de los límites de la caja

Se obtiene el centro de masa de la caja y se guarda ese valor conjunto con el tiempo, y los pulsos de control entre -1 y 1 al observar movimientos demasiado bruscos deformando la imagen, por esta razón se limita esta entrada al análisis solo de valores de pulsos entre -0.3 y 0.3 velocidades relativamente aceptables para la realización de este proyecto.

En este caso particular movimiento del ángulo del eje Z de micro-UAV (vea la Figura 24) para sacar el desplazamiento en ángulo “yaw”. Para el caso de planta de pitch solo se utilizará un controlador on-off al ser una planta no lineal ya que se trabaja con el área del cuadro, además la velocidad con las que trabaja micro-UAV a pulsos mayores de 0.2 aumenta el riesgo de colisionar un obstáculo cercano a menos de 2m.

Figura 48 Secuencia de seguimiento con “Lucas Kanade”

Con los valores guardados se procede a graficar las funciones obtenidas en la Figura 49, en la parte superior es el delta desplazamiento contra el tiempo para este análisis se toma en cuenta el valor de los pixeles desplazados. En la parte inferior de la gráfica se observa los pulso enviados a micro-UAV en ese mismo intervalo de tiempo.

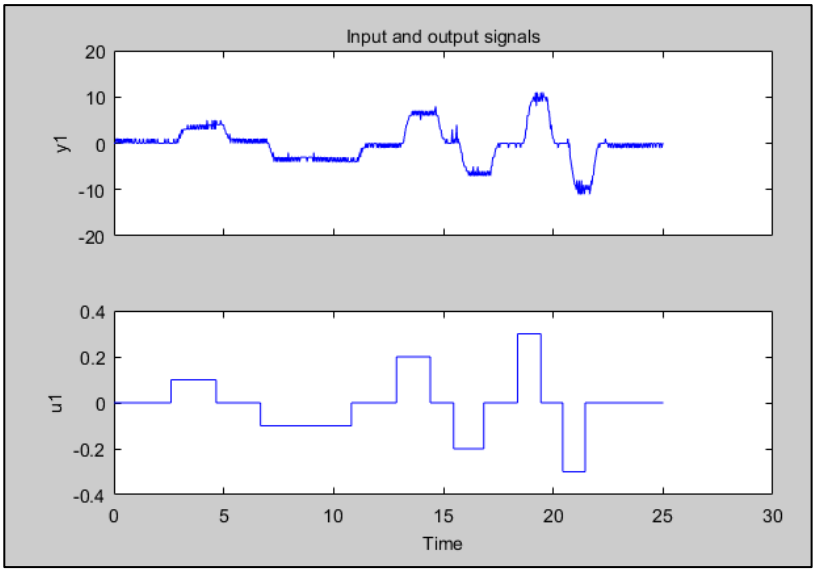

Figura 49 Inferior- Entrada de los pulsos de control
Superior - Salida del delta desplazamiento

Cabe mencionar que en cada frame obtenido se obtuvo el tiempo que transcurrió entre frame a frame por este motivo se puede calcular la frecuencia de muestreo de la adquisición de los datos que son:

$$Ts = \frac{Tiempo\ transcurrido\ total}{Número\ total\ de\ muestras} \tag{11}$$

Reemplazando los valores por el tiempo que transcurrió de 25.0466 segundos y todas las 732 muestras. Se obtiene el tiempo de muestreo igual 0.0342 segundos o a su vez de 29.22 Hz.

Con esta funciones de entra (Pulsos) y de salida (Desplazamiento) se puede utilizar la identificación de sistema de Matlab como se observa en la Figura 50.

Obteniendo la función de transferencia:

$$G(s) = \frac{71.99}{s + 1.842}$$

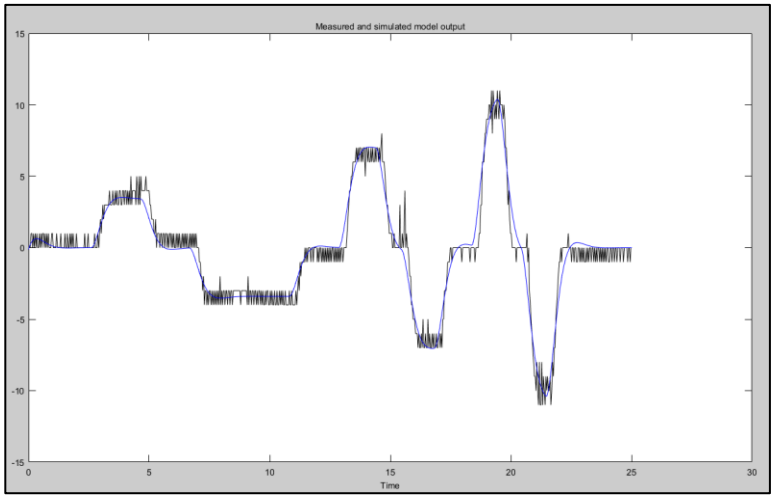

Figura 50 Identificación de la Planta

Esta función de transferencia se debe al modelo de la ecuación (12) donde $K_p = 39.08$ y $T_{p1} = 0.54288$. Para el caso particular de "yaw".

$$G(s) = \frac{K_p}{T_{p1} \times s + 1} = \frac{A}{B} \tag{12}$$

5.2.1. Diseño del controlador

Para hacer el seguimiento de una persona por medio del micro-UAV se debe implementar un controlador que asegure el correcto seguimiento del objetivo por lo tanto que en lo posible no la pierda de vista. Para este reto se eligió el movimiento en "yaw" para el seguimiento visual de la persona.

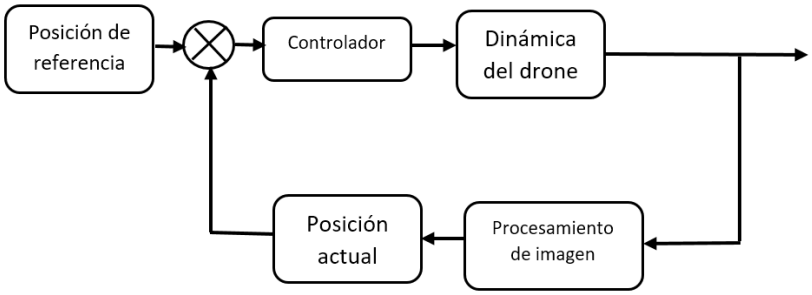

Figura 51 Diagrama de bloques del funcionamiento del sistema de control

Se requiere que el controlador sea capaz de asegurar un error en estado estable igual a cero. Por lo que se utilizará un controlador PI, en el caso de presentarse que el sistema presenta oscilaciones se procederá a utilizar un controlador PID (W. Aguilar, Casaliglla, & Pólit, 2017)(W. G. Aguilar, Casaliglla, & Polit, 2017)(W. G. Aguilar, Casaliglla, Pólit, Abad, & Ruiz, 2017).

Controlador PI

$$Gc(s) = \frac{sK_c + K_i}{s} = \frac{D}{E} \tag{13}$$

Para determinar las constantes del controlador PI se utiliza el método de asignación de polos, por lo tanto la función de transferencia en lazo cerrado sería:

$$H(s) = \frac{D\ x\ A}{E\ x\ B + D\ x\ A} \tag{14}$$

Reemplazando los valores e igualando a la ecuación característica:

$$\begin{aligned} E\ \times\ B + D\ \times\ A &= s\left(T_{p1} \times\ s + 1\right) + K_p(s \times\ K_c + K_i) \\ &= s^2 + 2\,\zeta\ \times\ w_n\ \times\ \ s + {w_n}^2 \end{aligned} \tag{15}$$

Analizando un sobre impulso de por lo menos un 10%:

$$Mp = 10\% = \ e^{-\frac{\Pi\, x\, \zeta}{\sqrt{1-\zeta^2}}} \tag{16}$$

Resolviendo tenemos:

$$\zeta = 0.5912 \tag{17}$$

Tomamos ecuación de Tolerancia 2% y definimos un $T_{s1} < 2\ seg$.

$$T_{s1} = \frac{4}{\zeta\ \times\ w_n} \tag{18}$$

$$w_n = \frac{4}{(1.5) \times (0.5912)} = 4.5406 \tag{19}$$

Resolviendo la ecuación (15) :

$$s^2 + s\left(\frac{K_c\ \times\ K_p}{T_{p1}} + 1\right) + \frac{K_p \times\ K_i}{T_{p1}} = s^2 + 2\ \times\zeta\ \times\ w_n\ \times\ \ s + {w_n}^2 \tag{20}$$

Reemplazando valores e igualando tenemos:

$$\frac{K_c\ \times\ K_p}{T_{p1}} + 1 = \ 5.36881 \tag{21}$$

$$\frac{K_p\ \times\ K_i}{T_{p1}} = 20.617 \tag{22}$$

Resolviendo la ecuación y la ecuación $Kc = 0.048992$ y $Ki = 0.286401$.

Mediante la respuesta al impulso unitario en la Figura 52 se comprueba el funcionamiento del controlador teniendo un Mp=15% y un tiempo de estabilización de 1.12 segundos. Valores aceptables para la aplicación en el seguimiento de la persona.

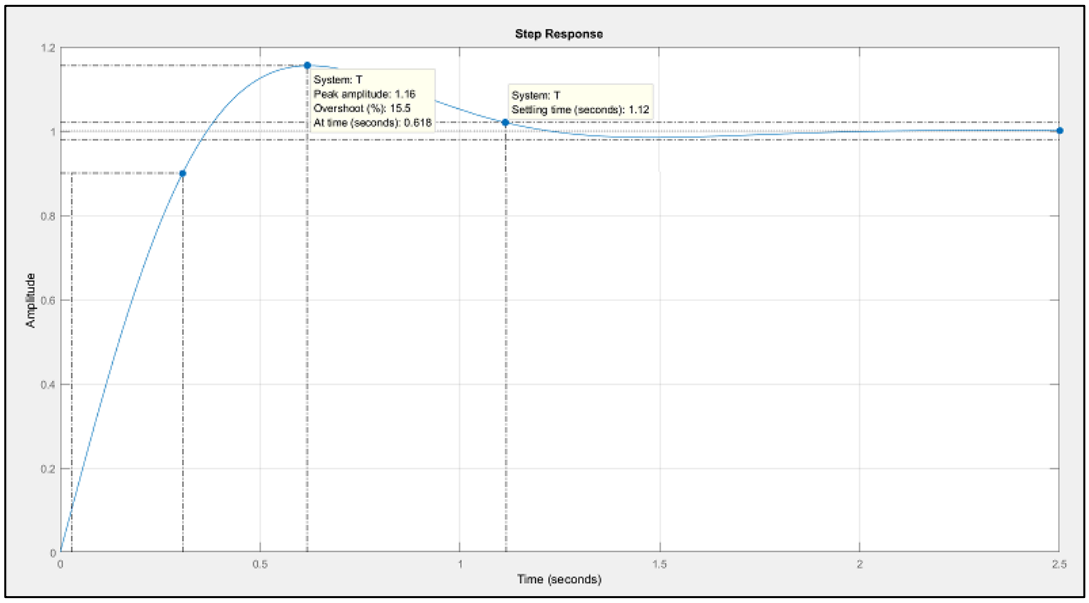

Figura 52 Respuesta impulso unitario del controlador planta yaw

5.3. Seguimiento de persona

La solución que se dio al seguimiento de persona es la utilización de un "tracker" o seguidor de la librería de "open cv", este seguir es medianflow que a continuación se explicará su funcionamiento. Se eligió este seguidor ya que funciona bastante bien al seguir un objeto acercando algo que no funciona con KCF, o MIL por ejemplo.

Para seguir a una persona primero se debe definir el cuadro a seguir, obtener el centro en "x" del cuadro y calcular el error que existe al centro de la imagen obtenida por micro-UAV como se observa en la Figura 53.

Figura 53 Error en eje de las “x” en imagen del micro-UAV

5.3.1. Implementación del controlador

Para reducir el error que se presenta entre la persona detectada y el centro de la imagen obtenida por micro-UAV, se implementa el controlador PI de la ecuación (13) reemplazando las constantes calculadas, donde la salida es co(k) en el intervalo discreto de k, en la Figura 54 se observa su implementación. De la ecuación (23) pasamos a código en Python.

$$co(k) = co(k-1) + Kc \times \big(e(k) - e(k-1)\big) + Ki \times Ts \times e(k) + \frac{Kd}{Ts} \times \big(e(k) - 2e(k-1) + e(k-2)\big) \quad (23)$$

En la Figura 54 se ingresa a la función el valor actual del eje x y el set point, además se agregan límites máximos y mínimos que pueden tomar los valores del controlador, ya que a partir de esos valores seguirían aumentando haciendo que el controlador no funcione correctamente.

```
def controlador(ejex,sp):
   e=sp-ejex
   T=0.0342
   kc=0.048992
   ki=0.286401
   kd=0

   if t==0:
      co = (kc+ki*T+kd/T)*e
      ek2=0
      ek1=e
      cok1=co
      t=t+1
   if t>0:
      co=cok1+kc*(e-ek1)+ki*T*e+(kd/T)*(e-(2*ek1)+ek2)
      if co>17.7:
         co=17.7
      if co<-17.7:
         co=-17.7
      ek2 = ek1
      ek1 = e
      cok1 = co
   return co
```

Figura 54 Aplicación del algoritmo de control

5.3.2. Seguidor Medianflow

El seguidor medianflow acepta un cuadro delimitador y un par de imágenes. Se realiza un seguimiento de una serie de puntos dentro del cuadro delimitador, se calcula su error y se filtran los valores atípicos (Kalal, Mikolajczyk, & Matas, 2010).

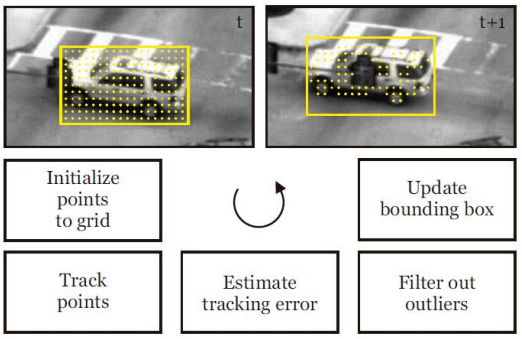

Figura 55 Proceso de seguidor medianflow
Fuente: (Kalal et al., 2010)

En la Figura 55 el rastreador acepta un par de imágenes I, I + 1 y un cuadro delimitador βt y genera el cuadro delimitador βt + 1. Un conjunto de puntos se inicializa en una cuadrícula rectangular dentro del cuadro delimitador βt. Estos puntos son seguidos por el rastreador Lucas-Kanade que genera un flujo de movimiento escaso entre It e It + 1. La calidad de las predicciones

de los puntos se estima y las peores predicciones se filtran. Las predicciones restantes se utilizan para estimar el desplazamiento de todo el cuadro delimitador (Kalal et al., 2010).

5.3.3. Algoritmo de Seguimiento de persona

En la Figura 56 se observa el diagrama de flujo del algoritmo de seguimiento de persona utilizando medianflow como seguidor de persona y detección de objetos para delimitar la persona a seguir.

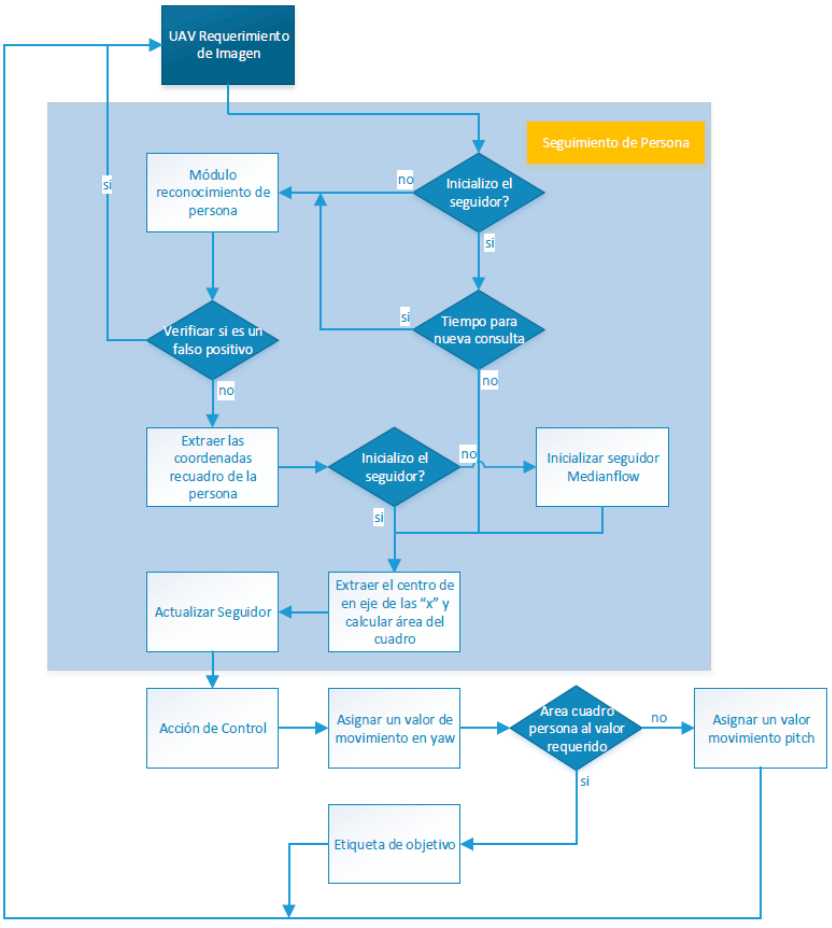

Figura 56 Diagrama de flujo de seguimiento de persona

Secuencialmente se muestra en la Figura 57 En cuadro rojo la detección de la persona, una vez se haya detectado se realiza una validación de falsos positivos, a continuación se guardan los valores de las coordenadas de las esquinas del cuadro delimitaros para enviarlas al seguidor medianflow el cual se representa con cuadro de color azul en la secuencias 2, 3 y 4 se observa como el micro-UAV intenta reducir el error a cero colocando a la persona en el centro de la imagen.

Figura 57 Secuencia de imágenes de seguimiento de persona

5.4. Evasión de obstáculos

La evasión de obstáculos no definidos es posible con la estimación de profundidad monocular, esta estimación no es exacta ya que tiene un rango de variación, pero ayuda en la detección de zonas menos densas por donde el micro-UAV puede volar.

5.4.1. Función de niveles y ponderaciones

Para esto se toma un rango de estudio en el mapa de profundidad y se hace un análisis tramo por tramo como se cómo se observa en Figura 58 donde en la parte superior se tiene la imagen

real, seguida del mapa de profundidad y finalmente los valores promedio de una pequeña zona horizontal del mapa de profundidad.

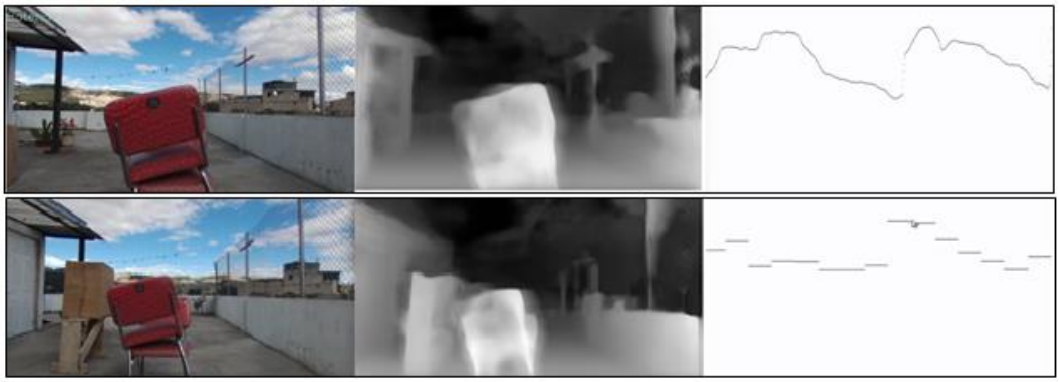

Figura 58 División por tramos de la estimación de profundidad

El promedio de esta zona es de un ancho de 40 pixeles de altura, se toma tramos en este caso 9 divisiones que equivalen a 95 pixeles de ancho, tomando desde el j=245:285. La razón por la que se dividió en 95 pixeles es el ancho de una persona a 1.5 metros de distancia del dron aproximado, esto sería el espacio mínimo por el cual micro-UAV puede pasar.

```
def plano(imagen):
     mayor=95
     menor=0
        global funcion
        global niveles
        global valido
     for a in range(0,9):
                  suma_pixel=0
           prom_pixel=0
           for i in range(menor,mayor):
                       #pixel1 = imagen[250,i]
                   for j in range(245,285):
                      pixel = imagen[j,i]
                           suma_pixel=suma_pixel+pixel
           prom_pixel = suma_pixel/3840
           menor=mayor
           mayor=mayor+95
     return niveles
```

Figura 59 Función de aproximación del plano de mapa de profundidad

Esta función generada se encuentra ponderada por los valores de estimación de profundidad como se observa en la Figura 60 tercer cuadro.

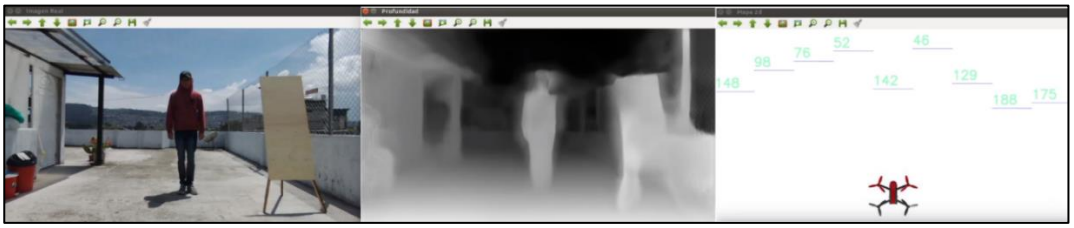

Figura 60 Ponderación de los tramos de la estimación de profundidad

5.4.2. Cálculo de obstáculos en los costados

En el caso de tener un objeto lo suficientemente cerca como para colisionar como nos indica la Figura 61 con la hélice tanto izquierda como derecha, se realiza un promedio entre los tramos primero y segundo para detectar un obstáculo extremo derecho como también un promedio entre los tramos octavo y noveno para detectar un obstáculo extremo izquierdo.

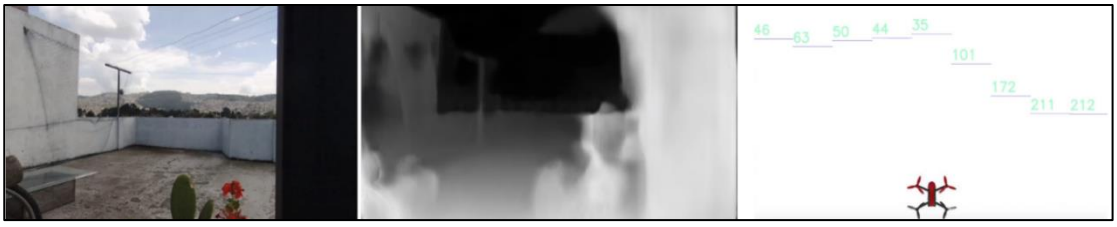

Figura 61 Acción de movimiento en roll para evadir obstáculos muy cercanos

En este caso particular se procede a realizar una pequeña corrección de trayectoria con el movimiento de roll como se observa en la Figura 62.

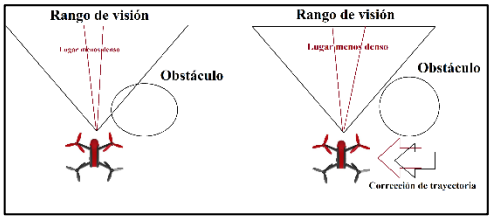

Figura 62 Corrección de trayectoria con movimiento roll

Este procedimiento se lo realiza también cuando el obstáculo se encuentre en el extremo derecho, en el caso de presentar obstáculos en ambos extremos se analiza si está libre en el centro para continuar, caso contrario retrocederá y buscará otra zona menos densa.

5.4.3. Algoritmo de Evasión de obstáculos

En el caso de evasión de obstáculos se une los puntos anteriores para seguir la zona menos densa simpre y cuando no exista objetos de obstruyan el paso del micro-UAV. En la Figura 63 se explica de mejor manera el algoritmo.

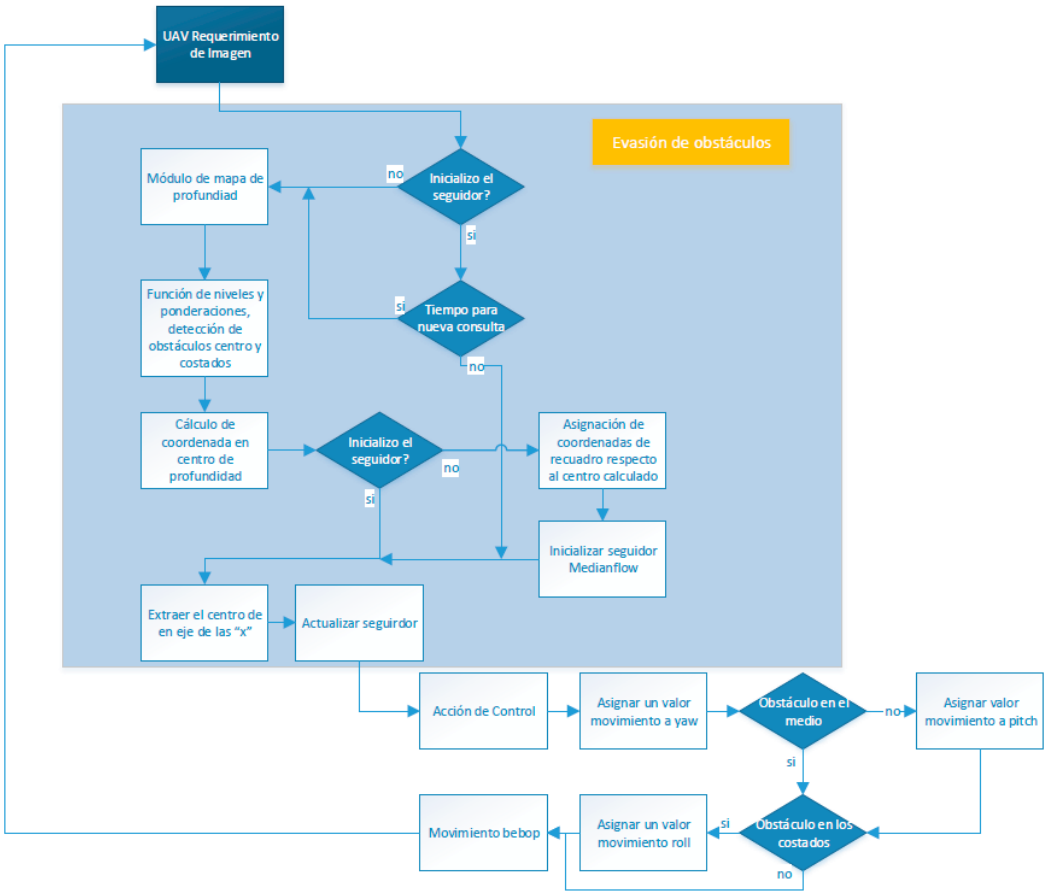

Figura 63 Diagrama de flujo de evasión de obstáculos

Para la secuencia mostrada en la Figura 64 se puede observar paso a paso a comprobación de la region mejor densa y el seguimiento de la misma. Este proceso es repetitivo y siempre se dirigue hacia delante mientras no se encuentre un objeto, en cuyo casi procede a evadirlo.

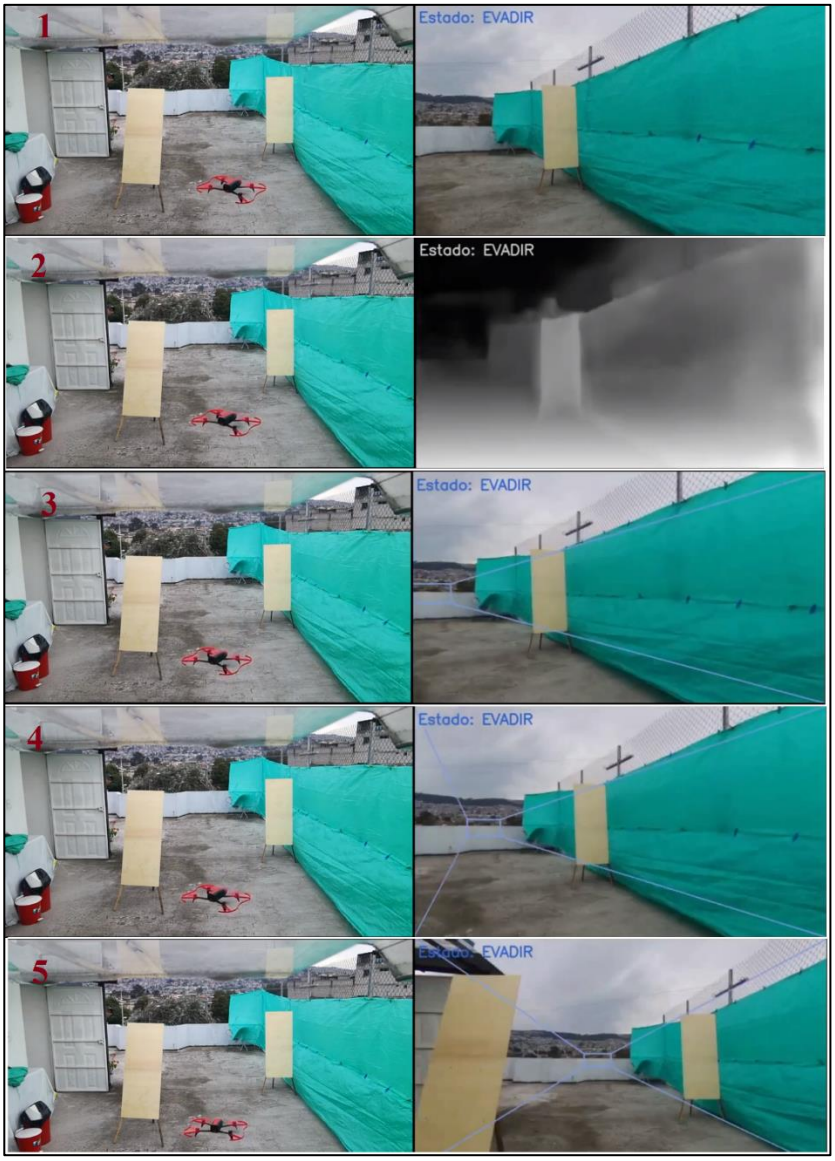

Figura 64 Secuencia de imágenes de evasión de obstáculos

5.5. Seguimiento de persona y evasión de obstáculos

Dada las condiciones tomadas en los anteriores puntos el análisis de algoritmo resúltate varía en “cuando evadir los obstáculos” y “cuando seguir a la persona”. Preferiblemente como explica la Figura 65 debe cumplir ciertas condiciones.

En el caso 1 que no se encuentre a la persona, se procede seguir por la zona menos densa y cada cierto periodo de tiempo busca en sus alrededores. Para este punto la evasión sería lenta al preguntar siempre de forma periódica donde se encuentra la persona. Si se realizara esto en forma simultánea los algoritmos saturarían el sistema.

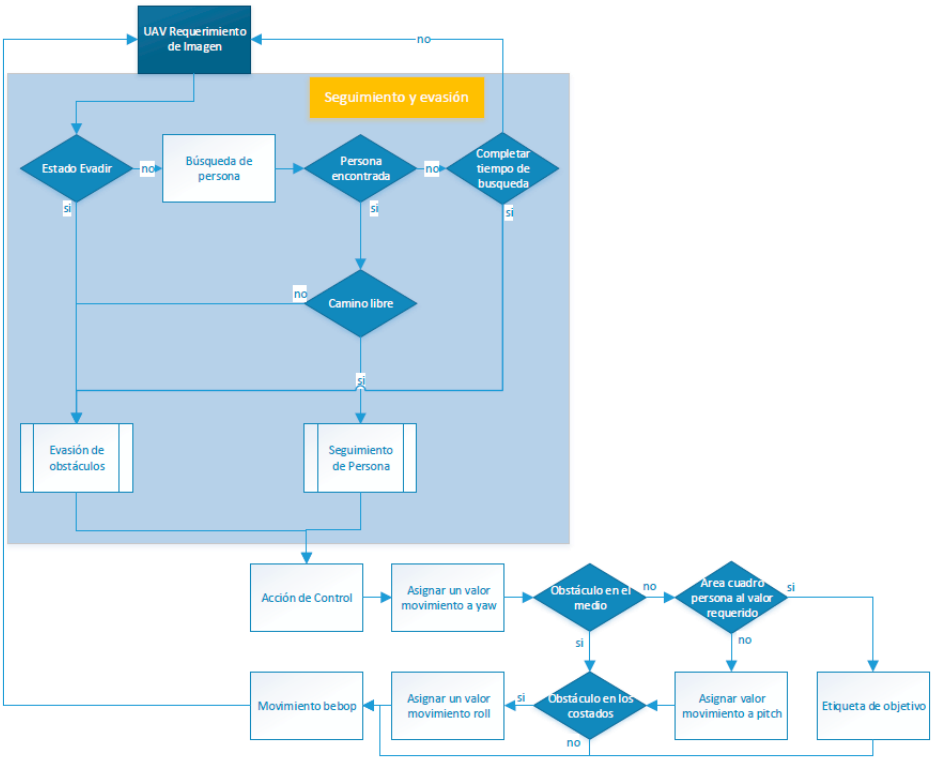

Figura 65 Diagrama de flujo de seguimiento de persona y evasión de obstáculos

En el caso 2 que sigue a la persona debe preguntar constantemente si el camino se encuentra libre, esto se lo realiza analizando si existen objetos cercanos en los extremos o en la trayectoria del micro-UAB. Caso contrario se procede a evadir.

En el caso 3 cuando se realice evasión demasiado brusca se haya perdido el rastro de la persona el micro-UAV buscará se toma una tolerancia de movimiento de la evasión si se movió más a su izquierda el micro-UAV buscará a la persona en la derecha y viceversa.

CAPÍTULO V

6. PRUEBAS Y RESULTADOS

6.1. Introducción

Este capítulo tiene la finalidad de evaluar el desempeño del sistema completo del seguimiento de persona y evasión de obstáculos, considerando diferentes escenarios. Para esto se planteado un conjunto de pruebas experimentales para diferentes condiciones, como son el viento, la luminosidad, el número de obstáculos, la distancia recorrida los cuales se detallarán más adelante.

Las pruebas realizadas se hicieron en dos entornos diferentes, el primero una terraza en el cual existe viento pero la distancia el recorrido de seguimiento es limitado. Como segundo escenario un parque teniendo como variables el aumento de número de obstáculos como son los árboles y el aumento de la distancia a recorrer.

6.2. Pruebas experimentales en una terraza

Las dimensiones del lugar con de 12 metros de profundidad y 6 metros de diámetro ideal para las primeras pruebas de funcionamiento del algoritmo. En este caso se utilizó como obstáculos dos tablas de 60 centímetros de ancho y 160 centímetros de alto.

Figura 66 Obstáculos de prueba en terraza

6.2.1. Prueba de detección y evasión con variación de luminosidad

Para este tipo de prueba se procedió a realizar el seguimiento a diferentes horas del día obteniendo como resultado diferentes intensidades de luz. Para esto se realizaron 6 pruebas con diferentes luminosidades y se obtuvieron diferentes variaciones de certeza de detección de persona. Estas pruebas se realizaron con la persona a una distancia de 3 metros.

Tabla 7
Certeza en detección a diferentes luminosidades

Prueba	Nivel de iluminación (lux)	Certeza de detección (%)
1	104541	97-99
2	31584	95-99
3	10548	90-95
4	1054	85-92
5	607	75-88
6	100	40-77

Con esto se puede decir que la iluminación influye en la detección de persona pero no en gran medida prácticamente la detección de persona responde bastante bien, esto se debe también a un factor importante en el entrenamiento de las imágenes en la red neuronal y es al aumento de imágenes con diferentes luminosidades en diferentes escenarios. Obviamente cuando se disminuya demasiado la intensidad lumínica demasiado será imposible que la red identifique alguna característica para la detección.

En la Figura 67 se observa las pruebas realizadas a diferentes horas del día y escenarios.

Figura 67 Seguimiento de imágenes con diferentes luminosidades

6.2.2. Prueba de detección y evasión con mayor número de personas

Para detectar el nivel de certeza al momento de la detección con más personas se realizará con una intensidad lumínica ideal y además con sujetos de prueba fuera del entrenamiento de la red neuronal. Inclusive con persona con característica de vestimentica casi idénticas a la persona entrenada. Para esto se realizaron 4 pruebas con 4 diferentes sujetos de prueba diferentes. De igual manera todas las pruebas se procuraron hacer a una misma distancia.

Tabla 8
Determinación detección persona con más sujetos de prueba

Prueba	Observaciones	Detección de persona entrenada (%)	Detección de persona no entrenada (%)
1	Sujeto lleva una vestimenta color rojo	95-99	35-45
2	El sujeto lleva una vestimenta de color blanca con gorra negra	93-99	70-75
3	El sujeto lleva una vestimenta de color tomate	96-98	32-40
4	El sujeto lleva una vestimenta de color amarillo	98-99	34-37

Se observa un aumento de falso positivo en el caso 2 sin embargo la detección de la persona entrenada no disminuye continua manteniendo niveles altos de aceptación lo cual se podría concluir que al aumentar la tolerancia de certeza de la detección de persona se eliminarían estos posibles falsos positivos.

Figura 68 Seguimiento de persona con más personas

6.2.3. Pruebas de detección de persona en diferentes distancias

El objetivo de esta prueba es determinar la detección mínima de una persona entrenada con valores de certeza aceptables. Para esto se ha tomado la extracción de datos a distancias desde 1 metro hasta 8 metros considerando condiciones de luminosidad favorables.

En la Figura 69 se puede observar 3 pruebas de secuencias a diferentes distancias.

Tabla 9
Valores de certeza a diferentes distancias

Distancia (m)	Valor de certeza (%)
1	95-99
2	92-98
3	91-98
4	85-95
5	72-83
6	65-71
7	58-68

Estos resultados sugieren que la distancia límite para una buena detección es de aproximadamente 4 metros, sin embargo los valores a 5 metros son aceptables desde el punto de vista de la tolerancia permitida en el seguimiento de la persona en este trabajo.

Figura 69 Detección de persona a diferentes distancias

6.2.4. Valor de la profundidad a diferentes distancias

Para esta prueba se analizará los valores obtenidos en la estimación de profundidad con obstáculos de diferente color además se realizará un comparación del algoritmo entre los diferentes dataset de entrenamiento.

Tabla 10
Comparación de distancias con diferentes colores de base de datos CitySpace

DISTANCIA (m)	verde		blanco		negro		rojo		amarillo		VARIACIÓN PROMEDIO
	min	max	min	max	min	max	min	max	min	max	
0,5	217	232	222	235	203	234	234	237	227	244	15,8
1	222	240	226	233	224	232	228	232	225	234	9,2
2	204	210	197	207	197	199	199	203	179	186	5,8
3	100	103	95	99	107	112	117	121	112	124	5,6
4	83	85	70	77	81	83	70	76	70	72	3,8
5	63	65	57	58	63	65	68	69	58	60	1,6
6	50	51	44	48	50	53	55	56	45	47	2,2
7	41	43	43	46	42	43	48	50	43	44	1,8

Tabla 11
Comparación de distancias con diferentes colores de base de datos KiTTi

DISTANCIA (m)	verde		blanco		negro		rojo		amarillo		VARIACIÓN PROMEDIO
	min	max	min	max	min	max	min	max	min	max	
0,5	132	152	135	149	154	182	205	211	193	208	16,6
1	192	196	108	117	172	189	198	209	189	194	9,2
2	202	210	146	162	187	192	190	198	200	204	8,2
3	128	130	108	113	119	123	131	133	139	141	3
4	85	87	75	80	78	84	86	88	85	91	4,2
5	67	70	69	71	64	67	67	69	66	68	2,4
6	54	56	60	62	52	55	54	57	56	57	2,2
7	45	46	50	52	45	47	47	50	49	51	2

Tabla 12
Comparación de distancias con diferentes colores de base de datos Eigen 2

DISTANCIA (m)	verde		blanco		negro		rojo		amarillo		VARIACIÓN PROMEDIO
	min	max	min	max	min	max	min	max	min	max	
0,5	197	208	137	143	224	233	233	237	192	220	11,6
1	186	188	188	200	218	224	204	213	192	198	7
2	200	205	209	213	215	218	222	227	208	216	5
3	117	121	137	138	123	128	137	140	128	132	3,4
4	82	84	90	91	86	87	98	99	92	97	2
5	62	64	67	68	66	67	73	74	78	79	1,2
6	55	56	56	57	51	53	59	59	63	64	1
7	50	53	51	52	48	49	54	55	53	54	1,4

Como se observa quien presenta menos variación es la de datos Eigen2 debido a que es la combinación entre la base de datos de Eigen y cityspaces. También se puede decir que el color blanco es quien presenta mayor diferencia entre los demás colores y así como también las distancias menos 1 metro genera mucha incertidumbre en todos los modelos. Lo que demostraría que estos modelos no funcionarían en lugares demasiado cerrados como una casa promedio.

6.3. Pruebas experimentales en un parque

Ahora la distancia no es un limitante, no es un escenario controlado y el número de obstáculos aumenta, con todas estas variables se procede realizar el análisis del seguimiento y evasión completado con estas condiciones extremas, así también con la determinación de la distancia mínima para detección de la persona.

6.3.1. Pruebas de distancias recorridas

Para esta prueba se realiza un análisis de 10 recorridos todos con una distancia constante aproximada de 20 metros.

Estas pruebas realizaron con la ayuda de dos baterías ya que micro-UAV tiene una autonomía de 15 min en vuelo.

Tabla 13
Recorridos con distancias completadas

Prueba	Trayectoria	Estado	Distancia (m)	Tiempo (min)
1	Aleatoria	Completada	22	3:21
2	Aleatoria	Completada	21	3:35
3	Aleatoria	Completada	18	2:28
4	Aleatoria	Completada	19	2:17
5	Aleatoria	Incompleta	10	1:14
6	Aleatoria	Completada	25	3:41
7	Aleatoria	Completada	28	4:05
8	Aleatoria	Incompleta	13	1:15
9	Aleatoria	Completada	21	3:25
10	Aleatoria	Completada	20	2:55

Las pruebas demostraron que el algoritmo tiene una eficiencia de 80% ya que no completo 2 de las 10 pruebas realizadas de forma continua. Cabe recalcar las trayectorias seguidas eran aleatorias es decir en tramos presenta rectas y en otras curvas. En algunas presentaba mayor número de obstáculos que en otras, las pruebas que no completo se debió a árboles demasiado juntos haciendo imposible su evasión. Esto estaría corroborado por la sección anterior en la cual se observa que a valores menores 1.5 metros de distancia los valores de estimación de profundidad varía en gran medida.

En la Figura 70 se observa una serie de secuencias las cuales comprueban el seguimiento y la evasión de obstáculos de forma continua en un sendero irregular.

Figura 70 Secuencia de imágenes de seguimiento de persona y evasión de obstáculos

6.3.2. Pruebas de detección de persona en diferentes distancias

En estas condiciones se prueba la detección de persona con una distancia mínima de 1 metro y una máxima de 8 metros.

Tabla 14
Valores de certeza a diferentes distancias

Distancia (m)	Valor de certeza (%)
1	85-90
2	82-89
3	75-86
4	65-71
5	58-65
6	50-52
7	34-38
8	20-30

Figura 71 Distancia aceptable de seguimiento entorno desconocido

En este caso tanto la poca luminosidad como el número de árboles presentes se obtuvieron una detección menor con respecto a las realizadas anteriormente, para solucionar esto se debería tomar nuevas imágenes en este tipo de escenario. Presentando además en las últimas pruebas un notable aumento de falsos positivos por debajo de la tolerancia de 50%.

CAPÍTULO VI

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

Se implementó un sistema simultáneo de detección de persona y evasión de obstáculos con una eficiencia al completar su misión de un 80 por ciento. Cabe recalcar que las trayectorias seguidas eran aleatorias dificultando la evasión, presentando tramos rectos y curvados. Así como también en otras presentaba un mayor número de obstáculos que en otras, las pruebas que no completaron el recorrido completo se debe a árboles demasiado juntos alrededor del tamaño de la mitad diámetro del marco de una puerta haciendo por el momento imposible su evasión. Esto está presentado por la sección de pruebas en donde valores menores 1.5 metros de distancia, los valores de estimación de profundidad varía de forma significativa teniendo un valores de incertidumbre muy grandes.

El seguimiento de persona se realiza en base una distancia de 1.5 metros aproximadamente, siempre tratando de mantener a la persona en el centro. Siendo posible solo con un controlador PI para el en movimiento de ángulo de giro "yaw", las variables de velocidad enviadas al "bebop 2" fueron de -0.3 a 0.3 teniendo en cuenta que a mayores velocidades no se podía apreciar una buena fluidez y calidad de imagen al obtenerlas.

El mejor análisis matemático para el control servo visual del "bebop 2" es a través de su estimación de movimiento generada por la obtención de imágenes a bordo, no se realizó un controlador para el movimiento de "pitch" y "roll" ya que el proyecto no requería un seguimiento a altas velocidades, además sería poco viable realizar una estimación de movimiento con la

diferencia de área generada por el movimiento de “pitch”, ya que se presentaría un sistema no lineal y para estos casos se deben aplicar otros criterios de control.

Utilizando un controlador PI fue suficiente para el desarrollo de control de ángulo de giro. No se necesitó una constante derivativa ya que el sistema no presentaba grandes oscilaciones, esto se puede explicar ya que “bebop 2” se encuentra en base a un controlador de estabilización intrínseco desarrollado por la empresa fabricadora del mismo.

Se desarrollaron pruebas de funcionamiento de detección mínima en la detección de persona entrenada con diferentes distancias concluyendo que en el primer escenario se obtuvo una tolerancia aceptable de máximo de 4 metros a distancias mayores la tolerancia disminuye considerablemente pudiendo generar falsos positivos.

Se realizó pruebas de medición de distancias con intercalados de 1 metro hasta 7 metros con los diferentes modelos pre-entrenados de estimación de profundidad tomando en cuenta la menor variación promedio se demostró que el mejor modelo es el de eigen2 utilizada en este proyecto. Además también se concluye que a distancias menores 1.5 metros la estimación falla considerablemente tendiendo un variación muy por encima de permitida en una evasión de obstáculos.

La utilización del sistema operativo ROS fue de gran ayuda en la realización de este proyecto tanto para la obtención de los datos de modelamiento matemático, ya que permitió realizar operaciones en paralelo, por medio de la fluidez al envío y recepción de datos, así también en la ejecución final del programa, destacando el uso de Python fue fundamental ya que gracias a su

fácil codificación se obtuvo resultados mucho mejores que al que se podría tener al utilizar otros lenguaje de programación más complejo.

7.2. Recomendaciones

Se recomienda tener en cuenta la autonomía de batería “bebop 2” que es aproximadamente de 15 minutos en vuelo, una pequeña ayuda para aumentar esta autonomía es dejando de grabar constante en vuelo, pero aun así es más factible adquirir una batería extra para la ejecución de nuevos de trabajos posteriores, así mismo en el ámbito de pruebas en terrenos externos.

Se recomienda en el apartado de la adquisición de imágenes de entrenamiento, obtenerlas en diferentes escenarios, diferentes luminosidades lo más variado que sea posible. Lo recomendable conjuntamente es que se vaya probando la detección de los objetos entrenados y se siga adjuntando una mayor base de imágenes diferentes y conjuntamente se siga entrenado con dichas nuevas imágenes.

Se recomienda para el desarrollo de trabajos futuros, realizarlos primero en lugares cerrados con buena luminosidad y amplios, ya que en este proyecto primero se lo hizo en una terraza generando un agente externo que es el viento que dificulto en las primeras pruebas de funcionamiento del mismo.

Para una futura investigación sobre el seguimiento de persona se recomienda utilizar un mejor seguidor que al visto en este trabajo “medianflow” ya que falla cuando los movimientos son demasiados bruscos. Una posibilidad es utilizar un seguidor que este desarrollado con redes neuronales convolucionales como es el caso “GOTURN”.

Se recomienda constantemente revisar la documentación del controlador “bebop_ autonomy” ya que en su última versión presenta cambio en la recepción de datos de control al “bebop 2”, así mismo se recomienda tener actualizado “tensorflow” para que al inicio de este proyecto se utilizó la versión 1.4 y se notó una mejoría al utilizar la última versión 1.12 en el entrenamiento de nuevas imágenes.

Para futuros trabajos se recomienda aumentar el número de sensores de adquirió de imágenes, es decir las cámaras abordo en un micro-UAV ya que de esta manera no existiría puntos ciegos en la evasión de obstáculos.

Se recomienda para solucionar los errores grandes de la estimación de profundidad a distancias menores a 1.5 metros se realice un entrenamiento con imágenes de interiores de esta manera se corregiría este error para la prueba en espacios sumamente cerrados.

BIBLIOGRAFÍA

Aguilar, W., Casaliglla, V., & Pólit, J. (2017). Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles. *Electronics*, *6*(1), 10.

Aguilar, W. G., Abad, V., & Ruiz, H. (2018). RRT-Based Path Planning for Virtual Bronchoscopy Simulator. *AVR 2017*, *10850*, 155–165.

Aguilar, W. G., & Angulo, C. (2014a). Estabilización de vídeo en micro vehículos aéreos y su aplicación en la detección de caras. *IX Congreso de Ciencia y Tecnologia ESPE 2014*, 155–160.

Aguilar, W. G., & Angulo, C. (2014b). Robust video stabilization based on motion intention for low-cost micro aerial vehicles. *2014 IEEE 11th International Multi-Conference on Systems, Signals and Devices, SSD 2014*, 1–6.

Aguilar, W. G., & Angulo, C. (2016). Real-Time Model-Based Video Stabilization for Microaerial Vehicles. *Neural Processing Letters*, *43*(2), 459–477.

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, CHILECON 2017 - Proceedings*, *2017–Janua*, 1–4.

Aguilar, W. G., Casaliglla, V. P., & Polit, J. L. (2017). Obstacle Avoidance for Low-Cost UAVs. *Proceedings - IEEE 11th International Conference on Semantic Computing, ICSC 2017*, 503–508.

Aguilar, W. G., Casaliglla, V. P., Pólit, J. L., Abad, V., & Ruiz, H. (2017). Obstacle Avoidance for Flight Safety on Unmanned Aerial Vehicles. *IWANN 2017*, *17*(2), 195–197.

Aguilar, W. G., Cobe, B., Rodriguez, G., Salcedo, V. S., & Collaguazo, B. (2018). SVM and RGB-D Sensor Based Gesture Recognition for UAV Control. *AVR 2018*, *10850*, 713–719.

Aguilar, W. G., Costa-castelló, R., & Angulo, C. (2014). Control Autónomo de Cuadricopteros para Seguimiento de Trayectorias. *IX Congreso de Ciencia y Tecnologia ESPE 2014*, 144–149.

Aguilar, W. G., Morales, S., & Ruiz, H. (2018). RRT* GL Based Path Planning for Virtual Aerial Navigation. *AVR 2017*, *10850*(November 2010), 176–184.

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). RRT* GL Based Optimal Path Planning for Real-Time Navigation of UAVs Wilbert. *IWANN 2017*, *17*(2), 195–197.

Aguilar, W. G., Quisaguano, F. J., Alvarez, L. G., Pardo, J. A., & Proaño, Z. (2018). Monocular Depth Perception on a Micro-UAV Using Convolutional Neuronal Networks. *Ubiquitous Networking*, *10542*, 392–397.

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017a). Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing (pp. 410–419). Springer, Cham.

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017b). Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing (pp. 596–606). Springer, Cham.

Aguilar, W. G., & Salcedo, V. S. (2017). Computational Neuroscience, *720*, 94–105.

Aguilar, W., & Morales, S. (2016). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics*, *5*(4), 70.

Al-kaff, A., Meng, Q., & Mart, D. (2016). Monocular vision-based obstacle detection / avoidance for unmanned aerial vehicles. *Proceedings of the Intelligent Vehicles Symposium*, (4), 3–8.

Camilo, C. (2018). *Reconocimiento de marcas de agua embotellada (Tesis Pregrado)*. Unviersidad Politécnica de cataluña.

Chakravarty, P., Kelchtermans, K., Roussel, T., Wellens, S., Tuytelaars, T., & Van Eycken, L. (2017). CNN-based single image obstacle avoidance on a quadrotor. *Proceedings - IEEE International Conference on Robotics and Automation*, 6369–6374.

Clément, G. (2018). Monodepth. Recuperado a partir de https://github.com/mrharicot/monodepth

CUDA. (2018). Procesamiento paralelo CUDA | Qué es CUDA | NVIDIA. Recuperado el 15 de noviembre de 2018, a partir de https://www.nvidia.es/object/cuda-parallel-computing-es.html

Cvpr, A., & Id, P. (2017). Speed / accuracy trade-offs for modern convolutional object detectors - 3562. *Cvpr*, 7310–7319.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks, (Nips).

Eduardo, S. (2014). *Desarrollo de un sistema de estimación de mapas de profundiad densos a prtir de sncuencias reales de video 3D (Tesis Pregrado)*. Universidad Politécnica de Madrid.

Eigen, D., Puhrsch, C., & Fergus, R. (2014). Depth Map Prediction from a Single Image using a

Multi-Scale Deep Network. *Nips*, 1–9.

Gageik, N., Benz, P., & Montenegro, S. (2015). Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors. *IEEE Access*, *3*, 599–609.

Gamino, I., & Sánchez Jonathan. (2018). *Detector de elementos para el detector de elementos para el videojuego Dark Souls (Tesis de Pregrado)*. Universidad Politécnica de Madrid.

Garg, R., Kumar, V., Carneiro, G., & Reid, I. (2016). Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue, *3021*, 740–756.

Giusti, A., Guzzi, J., Ciresan, D. C., He, F.-L., Rodriguez, J. P., Fontana, F., … Gambardella, L. M. (2016). A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots. *IEEE Robotics and Automation Letters*, *1*(2), 661–667.

Godard, C., Mac Aodha, O., & Brostow, G. J. (2016). Unsupervised Monocular Depth Estimation with Left-Right Consistency.

González, W. S. (2017). *Diseño y contrucción de un vehículo aéreo no tripulado (UAV) del tipo drone cuadricóptero de carrera*. Institución Universitaria Politécnico Grancolobiano.

Gupta, S. G., Ghonge, M. M., & Jawandhiya, P. M. (2013). Review of Unmanned Aircraft System (UAS). *International Journal of Advanced Research in Computer Engineering & Technology*, *2*(4), 1646–1658.

Jara-Olmedo, A., Medina-Pazmino, W., Tozer, T., Aguilar, W. G., & Pardo, J. A. (2018). E-services from Emergency Communication Network: Aerial Platform Evaluation. *2018 5th International Conference on eDemocracy and eGovernment, ICEDEG 2018*, 251–256.

Kalal, Z., Mikolajczyk, K., & Matas, J. (2010). Forward-backward error: Automatic detection of tracking failures. *Proceedings - International Conference on Pattern Recognition*, 2756–2759.

Katsuhiko, O. (1998). *Ingeniería de control moderna*. *Journal of Experimental Psychology: General* (Tercera ed, Vol. 136). Pearson Educación.

Kim, K. G. (2016). Book Review: Deep Learning. *Healthcare Informatics Research*, *22*(4), 351.

Lecumberry, F. (2005). Cálculo de disparidad en imágenes estéreo, una comparación. *XI Congreso Argentino de Ciencias de la Computación*.

Leos Monroy, G. (2014). *Implementación de una aeronave no tripulada de despegue vertical de cuatro rotores (Tesis de Pregrado)*. Universidad Nacional Autónoma de México.

Liu, F., Shen, C., Lin, G., & Reid, I. (2016). Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(10), 2024–2039.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector (pp. 21–37).

Lorandi, A., Hermida, G., Ladrón, E., & Hernández, J. (2011). Controladores PID y contoladores difusos. *Revista de la Ingeniería Industrial*, *5*(1), 13.

Mani Monajjemi. (2015). bebop_autonomy - ROS Driver for Parrot Bebop Drone (quadrocopter) 1.0 ; 2.0. Recuperado el 14 de noviembre de 2018, a partir de https://bebop-autonomy.readthedocs.io/en/latest/

Martín, A., & Barham, P. (2016). TensorFlow: A system for large-scale machine learning. *Methods in Enzymology*, *101*(C), 582–598.

Martínez-Carranza, J., Valentín, L., Márquez-Aquino, F., González-Islas, J. C., & Loewen, N. (2016). Detección de obstáculos durante vuelo autónomo de drones utilizando SLAM monocular Obstacle Detection during Autonomous Flight of Drones Using Monocular SLAM. *Research in Computing Science*, *114*, 111–124.

Mayorga Rodríguez, A. R. (2009). *Sistema de Navegación para Vehículos Aéreos Cuadricópteros (Tesis de Pregrado)*. Universitat Politécnica de Catalunya.

Mondragon, I. F., Campoy, P., Olivares-Mendez, M. a, & Martinez, C. (2011). 3D object following based on visual information for Unmanned Aerial Vehicles. *Robotics Symposium, 2011 IEEE IX Latin American and IEEE Colombian Conference on Automatic Control and Industry Applications (LARC)*, 1–7.

Morales, L., & Paucar, C. (2017). *Investigación e implementación de un sistema de seguridad fijo y móvil mediante un dron, usando visión artificial para detección y seguimiento de personas en un ambiente externo específico de la Universida de las Fuerzas Armadas ESPE-L (Tesis de Pregrado)*. Universidad de las Fuerzas Armadas ESPE-L.

Oleynikova, H., Honegger, D., & Pollefeys, M. (2015). Reactive avoidance using embedded stereo vision for MAV flight. *Proceedings - IEEE International Conference on Robotics and Automation*, *2015–June*(June), 50–56.

Olmedo, A., Medina, W., Rafael, M., Araujo, B., Aguilar, W. G., & Pardo, J. (2018). Interface of Optimal Electro-Optical/Infrared for Unmanned Aerial Vehicles Aníbal. *MICRADS 2018*, *94*,

372–380.

Parrot Drone. (s/f). Parrot S.L.A.M.dunk | Sitio Web Official de Parrot. Recuperado el 13 de febrero de 2018, a partir de https://www.parrot.com/es/profesional/parrot-slamdunk#parrot-slamdunk

Parrot SA. (2018). Dron Parrot Bebop 2. Recuperado el 9 de noviembre de 2018, a partir de https://www.parrot.com/es/drones/parrot-bebop-2-fpv#parrot-bebop-2-fpv-details

Pestana, J., Sanchez-Lopez, J. L., Campoy, P., & Saripalli, S. (2013). Vision based GPS-denied Object Tracking and following for unmanned aerial vehicles. *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2013*.

Pestana, J., Sanchez-Lopez, J. L., Saripalli, S., & Campoy, P. (2014). Computer Vision Based General Object Following For GPS-Denied Multimotor Unmanned Vehicles. *IEEE American Control Conference*, 1886–1891.

Pkulzc. (2018). Tensorflow Object Detection API. Recuperado a partir de https://github.com/tensorflow/models/tree/master/research/object_detection

Quan, Q. (2017). *Introduction to Multicopter Design and Control*. Singapore: Springer Nature.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Nips*, 91–99.

ROS. (2014). ROS Wiki - conceptos. Recuperado el 16 de noviembre de 2018, a partir de

http://wiki.ros.org/ROS/Concepts

ROS. (2018). ROS Wiki - cv_bridge. Recuperado el 14 de noviembre de 2018, a partir de http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImages Python

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

Saxena, A., Chung, S. H., & Ng, A. Y. (2006). Learning Depth from Single Monocular Images. *Advances in Neural Information Processing Systems*, *18*, 1161–1168.

Sucar, L. E., & Gómez, G. (2008). *Visión Computacional*.

Tokui, S., Oono, K., Hido, S., & Clayton, J. (2015). Chainer: a Next-Generation Open Source Framework for Deep Learning. *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 1–6.

Tomoyuki Mori, Se. S. (2013). First Resuts in Detecting and Avoiding Frontal Obstacle from Monocular Camera for Micro Unmanned Aerial Vehicles. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, *53*(9), 1689–1699.

Tran, D. (2017). Raccoon Detector Dataset. Recuperado a partir de https://github.com/datitran/raccoon_dataset

Tzutalin. (2015). LabelImg. Recuperado a partir de https://github.com/tzutalin/labelImg

Valero, C. (2017). *Desarrollo de aplicaciones basadas en visión con entornos ROS para el DRone Bebop 2. (Tesis de Pregrado)*. Universidad de Alcalá.

Wang, T., Qin, R., Chen, Y., Snoussi, H., & Choi, C. (2018). A reinforcement learning approach for UAV target searching and tracking.

Wu, N., & Rathod, V. (2018). Tensorflow detection model zoo. Recuperado a partir de https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking. *ACM Computing Surveys*, *38*(4), 13–es.