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 VINCULADO A UN MICRO-UAV PARA LA DETECCIÓN Y EVASIÓN DE OBJETOS DINÁMICOS A PARTIR DE IMÁGENES MONOCULARES"

AUTOR: ALVAREZ SAMANIEGO, LEANDRO GABRIEL

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 VINCULADO A UN MICRO-UAV PARA LA DETECCIÓN Y EVASIÓN DE OBJETOS DINÁMICOS A PARTIR DE IMÁGENES MONOCULARES**" fue realizado por el señor **ALVAREZ SAMANIEGO, LEANDRO GABRIEL** 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, **Alvarez Samaniego, Leandro Gabriel,** con cédula de identidad Nº 1722908314, declaro que el contenido, ideas y criterios del trabajo de titulación: "**DESARROLLO DE UN SISTEMA VINCULADO A UN MICRO-UAV PARA LA DETECCIÓN Y EVASIÓN DE OBJETOS DINÁMICOS A PARTIR DE IMÁGENES MONOCULARES**" 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

Leandro Gabriel Alvarez Samaniego

CC: 1722908314

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Yo, **Alvarez Samaniego, Leandro Gabriel,** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**DESARROLLO DE UN SISTEMA VINCULADO A UN MICRO-UAV PARA LA DETECCIÓN Y EVASIÓN DE OBJETOS DINÁMICOS A PARTIR DE IMÁGENES MONOCULARES**" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 27 de noviembre de 2018

Leandro Gabriel Alvarez Samaniego

CC: 1722908314

DEDICATORIA

La mayoría de las veces parece que estamos en una guerra sin fin, de repente todo entra en reposo y se juntan ideales y fuerzas por conseguir objetivos, como el presente Proyecto de Investigación.

El presento trabajo lo dedico a mi Madre que ya no está en este mundo, porque en su honor día a día soy un mejor caballero, porque a lo largo de mi vida ha sido fuerza de inspiración para luchar contra las batallas que he tenido que vivir, y las que estoy preparado a vivir.

A mi Padre, por ser mi mayor ejemplo de Trabajo, de Crecimiento, de Profesionalismo. Por confiar y apoyarme en todos mis Proyectos Personales, Profesionales y de Emprendimiento.

A mi hermana, quien ha sido como mi madre desde mis doce años, su ejemplo de Resiliencia, Educación y Altruismo han hecho de mi un hombre que puede aportar al mundo.

Lo dedico a mi familia, a mis sobrinos porque con su admiración me inspiran a ser un mejor ejemplo.

Finalmente lo dedico a todos los investigadores, por su aporte al conocimiento y al desarrollo tecnológico que gracias a ellos avanza constantemente.

Leandro Gabriel Alvarez Samaniego

AGRADECIMIENTO

Mi profundo agradecimiento es a toda mi familia, especialmente a mi padre Gilberto y a mi hermana Mary, porque a lo largo de mi vida han estado en mis mejores y duros momentos, porque su apoyo y paciencia y ejemplo de dominio me han ayudado a seguir adelante a pesar de mis tropiezos.

Agradezco a mi tutor, el Dr. Wilbert Aguilar por permitirme conocer el mundo de la investigación por darme la oportunidad de trabajar en sus proyectos y en el desarrollo del presente proyecto, con su ejemplo de amistad, humildad y profesionalismo motiva al estudiante al Desarrollo personal y colaborativo.

A los profesores de quienes me llevo lo mejor en lo personal y profesional, me indicaron el camino del conocimiento y del trabajo, y por mostrarme las realidades de la vida.

A todos mis amigos y amigas, por permitirme ser parte de su mundo y del mío, y que, en medio de risas, angustias, pruebas y aciertos hemos logrado formar grandes amistades, grandes proyectos, aprendizajes importantes y sobre todo grandes sociedades. Me llevo todo ese aprendizaje que ha sembrado en mi la inspiración y el deseo de ser un Profesional Emprendedor, enfocado en vías de aportar a la ciudadanía y sociedad.

"Los mejores negocios en los que puedes invertir son la Honradez y la Generosidad"

Leandro Gabriel Alvarez Samaniego

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente proyecto de investigación describe el desarrollo de un sistema de evasión de obstáculos para un vehículo aéreo no tripulado (UAV) controlado de manera remota desde una estación terrena, a partir de imágenes monoculares. La estación terrena es un computador portátil que forma parte de la red wifi del UAV, de esta proporciona información visual de su entorno hacia el computador para su respectivo procesamiento. Para cumplir con el objetivo de este proyecto se implementa una serie de métodos y técnicas para la detección, seguimiento, estimación de proximidad y evasión objetos en movimiento que se acerquen al UAV y comprometan su estabilidad. En la detección de movimiento se aplica una técnica de diferenciación entre fotogramas consecutivos para capturar información del objeto en movimiento, una vez que el objeto está en la mira del sistema con el método de Shi-Tomasi implementado se extrae puntos característicos de la imagen del objeto, a través de estimación de flujo óptico y herramientas estadísticas se logra un seguimiento del objeto a tiempo real. Es posible estimar cuando un objeto se acerca o aleja del UAV aplicando un método de regresión lineal, el cual se basa en el crecimiento de área del objeto representada en la imagen proporcionada por la cámara del UAV. La acción de evasión del UAV se realiza a través de un controlador difuso que tiene como variables de entradas la distancia y velocidad de acercamiento del objeto, la salida es la velocidad enviada al UAV.

PALABRAS CLAVE:

- **EVASIÓN OBSTÁCULOS**
- **ESTIMACIÓN DE PROXIMIDAD**
- **SEGUIMIENTO DE OBJETOS**
- **NAVEGACIÓN AUTÓNOMA**
- **CONTROL DE UAV**

ABSTRACT

This research project describes the development of an obstacle evasion system for an unmanned aerial vehicle (UAV) controlled remotely from remote station. The remote station is a portable computer that belongs at the UAV's Wi-Fi network, from which it provides visual information of its environment to the computer for its respective processing. To do the objective of this project, a several methods and techniques are implemented for the detection, tracking, proximity estimation and evasion of objects in motion that approaches to the UAV and compromise its stability. In motion detection, a differentiation technique between consecutive frames is applied to capture information about the object in motion, once the object is in the target of the system with the Shi-Tomasi method characteristic points are extracted from the image of the object , through optical flow estimation and statistical tools, object tracking is achieved in real time. Is possible to estimate when an object approaches or moves away from the UAV by applying the linear regression method, which is based on the growth of the area of the object represented in the image provided by the UAV camera. The evasion action of the UAV is done through a fuzzy controller that has as inputs variables the distance and approach speed of the object, the output is the speed sent to the UAV.

KEYWORDS:

- EVASION OBSTACLES
- PROXIMITY ESTIMATION
- MONITORING OF OBJECTS
- AUTONOMOUS NAVIGATION
- UAV CONTROL

CAPITULO 1

INTRODUCCIÓN

1.1 Antecedentes

El interés de los vehículos aéreos no tripulados (UAVs) tanto a nivel de centros de investigación como de sectores de usuarios potenciales, ha aumentado de manera considerable en los últimos años. Se entiende por UAV (Unmanned Aerial Vehicle) una aeronave que es capaz de realizar una misión sin necesidad de tener una tripulación embarcada sin excluir la existencia de piloto, controlador de la misión u otros operadores, que pueden realizar su trabajo desde tierra (Crespi de los Reyes, 1000). Kendoul (Kendoul, 2012) clasificó los UAV en cinco categorías según el tamaño y la carga: Categoría I: escala completa; las características principales son la solidez de la estructura física y la carga útil que pueden transportar; Categoría II: mediana escala; tienen una carga útil superior a 10 kg y un peso total superior a 30 kg; Categoría III: pequeña escala; estos UAV tienen una carga útil de 2 a 10 kg, con un peso total inferior a 30 kg; Categoría IV: mini; tienen una carga útil de 2 kg, son de funcionamiento eléctrico, de bajo costo, tienen un mantenimiento sencillo y una operación segura; y Categoría V: los micro vehículos aéreos (MAV) tienen una carga útil inferior a 100 gramos y se utilizan en navegación y detección.

Los UAV cubren una amplia gama de aplicaciones civiles y militares, pueden realizar misiones tanto al aire libre como bajo techo en entornos muy desafiantes, pueden ser equipados con varios sensores y cámaras para realizar misiones de inteligencia, vigilancia y reconocimiento. Las aplicaciones se pueden basar en el tipo de misiones militar o civil, tipo de zonas de vuelo exterior o interior. Por ejemplo, estos aviones no tripulados se pueden utilizar para misiones de búsqueda y

rescate, protección del medio ambiente, envío por correo y entrega, realizar misiones en océanos u otros planetas y otras aplicaciones diversas (Liu, Zhang, Yu, & Yuan, 2016), pueden proporcionar una visión general rápida alrededor del área objetivo sin ningún peligro, los drones equipados con cámaras infrarrojas pueden proporcionar imágenes incluso en la oscuridad, debido a sus reducidas dimensiones, se pueden utilizar micro-UAV para reconocimiento dentro de edificios (Radim Stuchlík & , Kamil Láska, s/f).

El diseño de UAVs autónomos requiere de un sistema de navegación apropiado para la detección de obstáculos, estimación de estado y percepción del entorno. En (Hassanalian & Abdelkefi, 2017) describe la IMU (Unidad de Medición Inercial) como un sensor de navegación convencional, el LIDAR (Gageik, Benz, & Montenegro, 2015) como un dispositivo adecuado para la cartografía y la detección de obstáculos, ya que mide directamente el rango mediante el escaneo de un haz de láser en el entorno. Las cámaras se destacan como un método popular para el ambiente de detección, puesto que son ligeros, pasivos, y compactos; proporcionando amplia información acerca del movimiento del vehículo y su entorno. La visión por computador es una solución popular para aplicaciones que requieren la detección y seguimiento de objetivos, reconocimiento de objetivos y otras tareas.

Yilmaz, Javed, Mubarak (Yilmaz, Javed, & Shah, 2006) describen varios métodos de detección y seguimiento de objetos en el entorno, los objetos pueden ser representados como formas geométricas, puntos, el contorno del objeto, en el caso de personas como modelos articulados, o densidades de probabilidad, tomando en cuenta la características de seguimiento como el color, la textura, el flujo óptico o los bordes. Flujo óptico es un método usado a menudo para evitar colisiones no basado en visión estereoscópica. Es un fenómeno visual con experiencia diaria al

observar un objeto que se mueve a una velocidad diferente a la del observador. El movimiento del obstáculo observada depende de la distancia entre el observador y obstáculo y su velocidad relativa (Zingg, Scaramuzza, Weiss, & Siegwart, 2010). Los algoritmos de detección basados en puntos de interés, como el detector SIFT (Vedaldi, 2008) siendo el más robusto y resistente a deformaciones de la imagen, los algoritmos de segmentación de imágenes es para dividir la imagen en regiones perceptualmente similares como el método exclusivo para seguimiento en tiempo real Mean-shift (Comaniciu, Ramesh, & Meer, s/f); el aprendizaje supervisado es un método que requiere una gran colección de muestras de cada clase de objeto, entre los enfoques de aprendizaje se tiene a las redes neuronales, máquinas de vector de soporte (SVM) arboles de decisión.

El flujo óptico puede surgir del movimiento relativo de los objetos y el espectador, en consecuencia, puede proporcionar información importante sobre la disposición espacial de los objetos vistos y la velocidad de cambio de esta disposición. Se calcula utilizando la restricción de brillo, que asume la constancia brillo de los píxeles correspondientes en tramas consecutivas comúnmente como una característica en aplicaciones de segmentación y seguimiento basados en movimiento. Barron, Fleet, & Beauchemin realizan comparaciones empíricas en la precisión, confiabilidad y densidad de las mediciones de velocidad de las técnicas existentes para la estimación del flujo óptico que se puede clasificar en 4 categorías: las basadas en gradientes espacio-temporales, las basadas en comparación de regiones, las basadas en fase y las basadas en energía, en todas las estrategias de estimación de flujo óptico se parte de la hipótesis de que los niveles de gris permanecen constantes ante movimientos espaciales en un tiempo dado (Barron, Fleet, & Beauchemin, 1994).

El método de estimación del flujo óptico propuesto por Lucas y Kanade (Tomasi & Kanade, 1991) calcula el flujo óptico sólo sobre un conjunto de puntos de interés, este método asume que cada punto de interés se encuentra en una región R, que presenta un flujo óptico constante, y no sobre todos los píxeles pertenecientes a la imagen. Tomando en cuenta que no aporta resultados confiables, se plantea como solución la técnica de Shi y Tomasi (Jianbo Shi & Tomasi, 1994), la cual se basa en el criterio de "buenos puntos para seguir", el cual encuentra regiones que contienen bordes y esquinas dentro de la imagen, las cuales resultan fáciles de seguir.

La tendencia de varios grupos de investigación es la navegación sobre entornos conocidos, es decir, el lugar es previamente explorado, puesto que el principal problema es la evasión de obstáculos, como en el trabajo de investigación basado en sensor LIDAR en UGVs (Wilbert G. Aguilar, Sandoval, et al., 2018). El autor Richter demostró maniobras agresivas para los cuadricopteros que volaban en ambientes interiores obstruidos (Richter, Bry, & Roy, 2016),el algoritmo de planificación utilizado, RRT* ("Robotics: Science and Systems VI - Yoky Matsuoka, Hugh Durrant-Whyte, José Neira - Google Libros", s/f), no se implementó en tiempo real. La fase de planificación se realizó fuera de línea, con un mapa a priori de obstáculos, produciendo puntos de referencia que son la distancia mínima, no necesariamente el tiempo mínimo, hasta el objetivo.

Existen múltiples soluciones para la evasión de obstáculos, tradicionalmente usando detección activa como el sonar, Infrarrojos, LIDAR mencionados en el trabajo de Paul Benz (Gageik et al., 2015), estos sensores pueden ser costosos, tanto en términos de costo real como en términos de peso y requisitos de potencia. La detección pasiva común es la estéreo visión que se encarga de obtener información 3D desde 2 puntos de vista (2 cámaras) (Marr & Poggio, 1979), teniendo como limite la distancia de tomas de datos. Actualmente para existe en el mercado un kit de navegación

y evasión de obstáculos llamado SLAM drunk que permite el desarrollo de drones o aviones no tripulados.

Los sistemas para la navegación comúnmente están compuestos, por el micro-UAV que tiene incorporado el dispositivo de visión, controlados desde una estación terrestre (PC Portátil) desde donde se envían las órdenes, aquí se procesa la información provista durante el vuelo. La plataforma que permite el enlace mediante red Wifi es el Sistema Operativo para Robots (ROS), el cual contiene una variedad de Herramientas y Drivers necesarios para este propósito.

1.2 Justificación e Importancia

En la Actualidad el uso de drones ha tenido un crecimiento notable de aplicaciones tanto en la investigación como en lo comercial, por ejemplo, en seguridad y aplicaciones militares contar con un vehículo aéreo no tripulado puede ser útil en misiones de reconocimiento ("Aplicaciones y usos, intelligenia DYNAMICS", s/f). Se puede aplicar en la agricultura para el control y monitorización de los cultivos. Exploración y rescate de lugares de difícil acceso como cuevas, precipicios, vigilancia, navegación de reconocimiento, entre otras (Martínez-Carranza, Valentín, Márquez-Aquino, González-Islas, & Loewen, 2016).

La autonomía de los UAV simplifica el trabajo manual que realizan los operarios, el vuelo de estos es a bajas alturas y comprometido a colisiones por esta razón se requiere de un sistema capaz de detectar obstáculos y tener una respuesta ante ello. La planificación de movimiento como rutas en 3D (W. Aguilar, Morales, Aguilar, & Morales, 2016), RRT* en UAVs (Wilbert G. Aguilar, Morales, Ruiz, & Abad, 2017b, 2017a), simulador de broncoscopia virtual (Wilbert G. Aguilar, Abad, Ruiz, Aguilar, & Aguilar-Castillo, 2017), identificación de personas (Aguaiza Guerrero, 2018), estimación de tiempo de viaje y rutas en Transporte público (Salgado, Tierra, Sandoval, &

Aguilar, s/f), seguimiento de ruta, evasión de obstáculos, detección de objetivos son áreas que incluyen los sistemas autónomos.

Los grupos de investigación han propuesto diferentes técnicas y sensores que permiten la detección de obstáculos, de alto costo como el sonar, el LIDAR (Gageik et al., 2015) o cámaras R-GBD (Thesis, University, & 2012, s/f), que en cuestión de autonomía no favorecen debido a su alto costo en el uso de recursos, como la energía que consume y el peso. Una alternativa a este problema sería el uso de cámaras externas, que no presentan consumo de energía al UAV, pero que estaría limitado a ambientes cerrados. La técnica más común es el uso de la visión estereoscópica, que se basa en el principio bajo el cual funciona la visión humana y permite diferenciar la profundidad a la que se encuentran los objetos, los costos de los equipos de estéreo visión son altos.

Las aplicaciones de imágenes monoculares son flexibles, puesto que el costo computacional permite la aplicación de varias técnicas y algoritmos, correspondientes a visión artificial (Yilmaz et al., 2006), la detección de objetos, el seguimiento, la estimación de aproximación, flujo óptico son algunas de las técnicas de visión artificial que posibilitan la navegación autónoma.

El Flujo óptico es un método comúnmente usado para la detección y evasión de obstáculos, muchas diferentes estrategias para evadir obstáculos dependen de este fenómeno, que a menudo se presenta como inspiración biológica. El movimiento del obstáculo observada depende de la distancia entre el observador y obstáculo y su velocidad relativa. por lo tanto, de flujo óptico puede ser utilizado para la estimación de distancias relativas. Se mide en unidades de velocidad angular como radianes por segundo o grados por segundo.

El trabajo implica la investigación de varias técnicas y algoritmos para la detección y estimación de proximidad de objetos basados en Flujo Óptico. Colocando la visión artificial como una

propuesta considerable y escalable para futuros trabajos. A diferencia de (Martínez-Carranza et al., 2016) que realiza SLAM monocular.

1.3 Alcance de Proyecto

El presente trabajo propone el desarrollo de un sistema autónomo, que esté en condiciones de detectar, estimar la colisión, y evadir un objeto que se aproxima a una velocidad no mayor a un metro por segundo, y pueda comprometer el estado del micro-UAV, partiendo de imágenes monoculares.

El trabajo de investigación iniciaría con la exploración de métodos para la detección de objetos a través de visión por computador, utilizando la información de imágenes monoculares. La estimación de Flujo óptico es una técnica que ayudar a encontrar y discriminar objetos que se desplazan en una secuencia de imágenes, se realizaría por medio de la versión piramidal del algoritmo de Lucas y Kanade (JEAN-YVES & B., 1999), la cual estima el flujo óptico por medio de pirámides Gaussianas de una manera iterativa. Los métodos comúnmente usados en (Yilmaz et al., 2006) que se sugiere son a partir de: Puntos de interés, por ejemplo detector Moravec, o Harris. Segmentación de imágenes con el algoritmo de Mean-Shift. La detección de objetos también se puede realizar por aprendizaje supervisado, como es el caso de entrenamiento con redes neuronales convolucionales, máquinas de vector de soporte (SVM), el algoritmo de adaboost, entre otros.

Una vez definida la técnica a emplear para la detección de objetos, de igual forma se abordará la estimación monocular de proximidad del objeto detectado. Por ejemplo, el indicador de Tiempo hasta Colisión (TCC), la distancia relativa del objeto hacia el plano focal de la cámara, llamado

punto de aproximación más cercano (CPA), basados en Flujo Óptico o en otros métodos de estimación (Bauer, Hiba, Vanek, Zarandy, & Bokor, 2016).

El siguiente paso es integrar la detección y estimación para diseñar un controlador, o un planificador de movimiento según sea el caso, puesto que al detectar el obstáculo, el vehículo debe recuperar la posición, además de estimar si el obstáculo está en la trayectoria de vuelo, el error de posición y la distancia del área objetivo entre el obstáculo y el micro-UAV.

El procesamiento del sistema se llevará a cabo en un computador personal, que recibirá toda la información de las imágenes monoculares transmitidas desde el micro-UAV con cámara integrada. Cabe mencionar que se abre la posibilidad de aplicar otras técnicas para la detección de objetos, dado que en el estado del arte pueden existir más, o se pueda aplicar técnicas propias como aporte.

1.4 Objetivos

1.4.1 Objetivo General

- Desarrollar un sistema autónomo para la detección y evasión de objetos dinámicos de un micro-UAV usando algoritmos de detección y estimación de proximidad.

1.4.2 Objetivos Específicos

- Desarrollar un estudio del estado del arte de técnicas y algoritmos para la detección y estimación de proximidad de objetos.
- Determinar los parámetros del controlador necesarios para la evasión de objetos dinámicos.
- Integrar la detección, estimación de proximidad y evasión de objetos dinámicos en un solo sistema.
- Implementar el sistema autónomo en un computador personal que permita comunicación y control del micro-UAV.
- Evaluar el sistema, a través de pruebas experimentales en entornos abiertos y cerrados.

CAPITULO II

MARCO CONCEPTUAL

2.1 Introducción

En este capítulo se realizará una descripción de todos los conceptos utilizados para el desarrollo de este proyecto de investigación, de igual forma el estado del arte de las investigaciones actuales sobre proyectos enfocados a la robótica móvil.

2.2 Vehículo Aéreo No Tripulado (UAV)

2.2.1 Descripción General

Actualmente en el campo de la robótica móvil, la investigación ha centrado sus desarrollos tecnológicos en vehículos aéreos no tripulados, sus siglas en inglés son UAV, el vuelo de los vehículos aéreos no tripulados (UAV) se puede controlar de manera autónoma mediante computadoras a bordo o mediante el control remoto desde un piloto en tierra o en otro vehículo. Los vehículos aéreos no tripulados también se llaman drones (Quan, 2017c). Ejemplo, redes ad hoc para tele operación en casos de desastre (Anibal Jara-Olmedo, Medina-Pazmino, Tozer, Aguilar, & Pardo, 2018), estabilización de video en UAVs (Wilbert G. Aguilar, Angulo, & Pardo, 2017), Carga útil infrarroja para la detección de tráfico (Aníbal Jara-Olmedo et al., 2018), Máquina de vectores de soporte para el manejo UAVs (Wilbert G. Aguilar, Cobeña, Rodriguez, Salcedo, & Collaguazo, 2018), mapeo 3D a partir de sensor RGB-D (Wilbert G. Aguilar et al., 2017).

Es importante comprender las diferencias entre UAVs y Aeronaves Modelo, Un UAV es más complejo que una aeronave modelo en términos de su composición. Un avión no tripulado consiste

en un fuselaje, un sistema de propulsión, un piloto automático, un sistema de tareas, un sistema de enlace de comunicación y una Estación de Control en Tierra (GCS), etc. Un avión modelo normalmente consta de un fuselaje, un sistema de propulsión, un simple estabilizado. Los drones se controlan de manera autónoma mediante computadoras a bordo o por pilotos remotos en tierra o en otro vehículo, mientras que los aviones modelo solo son controlados por pilotos remotos. Los drones de función se utilizan a menudo con el propósito de aplicaciones militares o civiles especiales. Se espera que realicen misiones particulares, mientras que los aviones modelo son usados para entretenimiento (Quan, 2017c).

2.2.2 Composición

Los multicópteros o multirrotores básicamente son helicópteros que en su estructura les compone más de dos hélices, tienen la capacidad de despegue y aterrizaje vertical (VTOL). El más operado en el campo de la robótica es el Cuadricóptero, en la *Figura 1* se puede ver la ilustración que posee cuatro hélices, las cuales, son las 4 entradas de Control.

Figura 1. **Cuadricóptero**

Debido a la estructura de múltiples rotores, sus momentos anti-torque pueden cancelarse entre sí. Gracias a la estructura simple, un multicóptero es fácil de usar y presenta alta confiabilidad y bajo costo de mantenimiento, pero su capacidad de carga útil y el tiempo de resistencia están comprometidos (Quan, 2017a)

2.2.3 Dinámica del Movimiento

De acuerdo con la teoría mecánica clásica, una fuerza se puede traducir en el cuerpo rígido y convertirse en una fuerza y un momento asociado, mientras que el momento se puede traducir directamente al eje central. Como resultado, se regulan el empuje y los momentos sobre su centro. Con el impulso y los momentos cambiados, la altitud y la posición de un cuadricóptero están controladas (Quan, 2017b). Como se muestra en la Figura 2, en posición estable, teniendo en cuenta que la forma de las hélices es contraria, todas las hélices giran a la misma velocidad angular, las hélices 1 y 3 giran en sentido antihorario y las hélices 2 y 4 en el sentido horario.

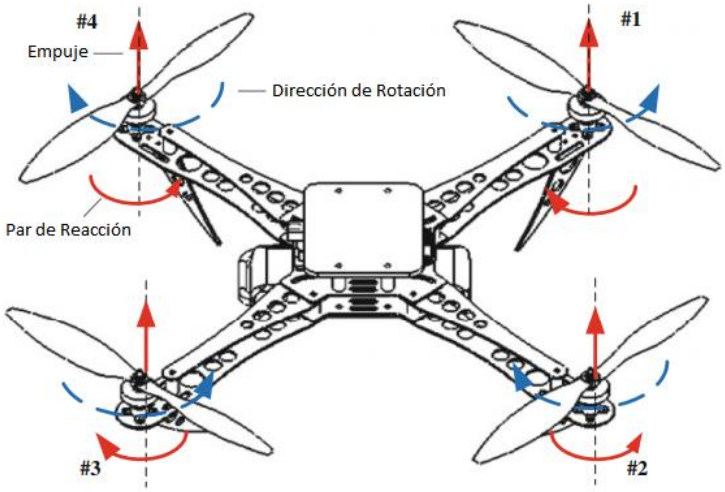

Figura 2. **Empuje y dirección de giro de las hélices de un cuadricóptero**

Haciendo referencia a la Figura 3, el movimiento hacia arriba o abajo del cuadricóptero se obtiene al aumentar o disminuir respectivamente, la velocidad angular de las helices en la misma cantidad para todas, los momentos de todas las hélices se cancelan.

Cuando las velocidades angulares de las hélices 1 y 4 disminuyen por igual, mientras la velocidad angular de las hélices 2 y 3 aumentan en la misma proporción, se obtiene un momento que provoca el avance del cuadricóptero. Y viceversa si el momento sería hacia atrás. Para la inclinación de izquierda y derecha se realiza la misma variación de velocidades, es decir las hélices 1 y 2 reducen su velocidad, mientras que las hélices 3 y 4 incrementan su velocidad para su movimiento izquierda. Para el Alabeo (Yaw) el momento de giro hacia la derecha se incrementa porque las velocidades angulares de las hélices 1 y 3 se incrementan en la dirección contraria a la de las agujas del reloj. Por otro lado, el momento de giro hacia la izquierda del cuadricóptero disminuye porque las velocidades angulares de las hélices 2 y 4 disminuyen en el sentido de las agujas del reloj. Esto provoca el giro hacia la derecha.

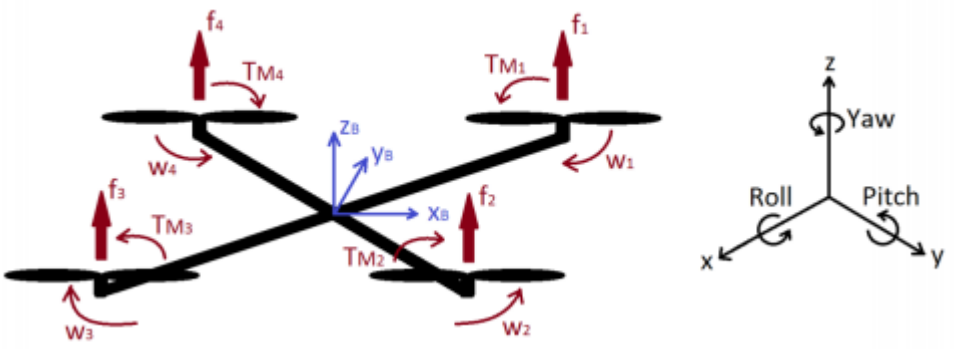

Figura 3. **Movimiento del Sistema con respecto a los ejes de rotación**
Fuente:(Salcedo Peña, 2018)

2.3 Percepción Visual

La percepción visual en los humanos se puede definir como la capacidad para interpretar la información que la luz del espectro visible hace llegar hasta nuestros ojos (Cognifit, 2017). En la robótica se requiere que los robots perciban su entorno, la visión artificial ofrece una percepción artificial a estos robots, para que puedan percibir el entorno real de una imagen o una secuencia de fotogramas, y así ser procesadas por un ordenador.

Los sensores basados en la visión se definen como sensores pasivos y tienen una ventaja intrínseca sobre los sensores láser y de radar: la posibilidad de adquirir datos de forma no invasiva, sin alterar el entorno. Además, pueden usarse para algunas aplicaciones específicas para las cuales la información visual desempeña un papel básico (como la localización de las marcas de los carriles, el reconocimiento de las señales de tráfico y la identificación de obstáculos) sin necesidad de realizar modificaciones en las infraestructuras viales (Bertozzi et al., 2002).

2.3.1 Visión Artificial

El hombre ha imitado muchas veces, en la construcción de sus artefactos, a la naturaleza. En este caso también se cumple. Las cámaras de vídeo con sus ópticas hacen las veces del globo ocular, mientras el computador realizará las tareas de procesamiento, emulando el comportamiento del cerebro. La visión artificial es una analogía a la visión del ser humano, y su propósito es “entender” las características de una imagen y en consecuencia generar una acción. La visión artificial está profundamente apoyada en el Procesamiento Digital de Imágenes, pero de ninguna forma son lo mismo (Rojas Hernández, Ramón, Ortigoza, María, & Molina, 2007). El proceso de obtener información a través de la visión artificial para la toma de una decisión puede esquematizarse como en la Figura 4.

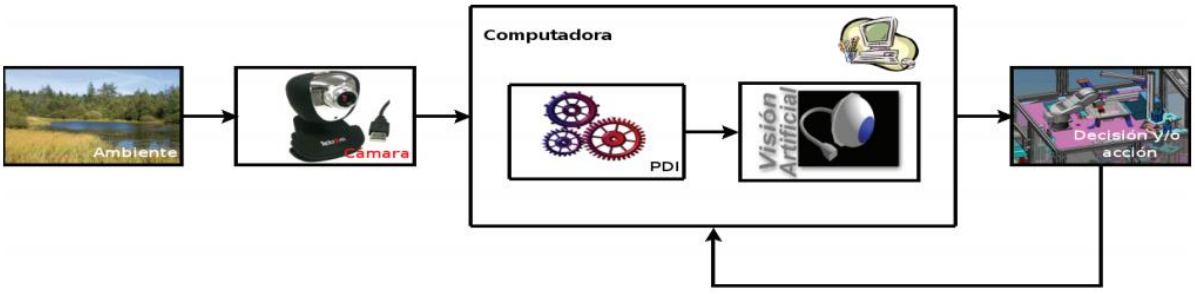

Figura 4. **Funcionamiento de la Visión Artificial en la Robótica**
Fuente: (Rojas Hernández et al., 2007)

La cámara es el elemento que reemplaza al ojo humano, tiene como propósito convertir la luz presente en el lugar de trabajo, en una señal eléctrica que una computadora pueda procesar. En la actualidad gracias a la visión artificial se ha logrado realizar tareas como: Reconocimiento de Gestos (Wilbert G. Aguilar, Cobeña, et al., 2018), Estabilización de Video en UAVs (Wilbert G Aguilar & Angulo, 2014a), Detección de Peatones a bordo de UAVs (Wilbert G. Aguilar, Luna, et al., 2017), Percepción de Profundidad y Detección de Objetos en UAVs (Wilbert G. Aguilar, Quisaguano, et al., 2018).

2.3.2 Visión Monocular

El sistema de visión humano nos permite percibir y comprender el mundo que nos rodea y nos suministra más de un 70% de la información total (imágenes, sonidos, sensaciones, etc.) que recibimos. Las escenas que percibimos suelen ser tridimensionales (3D) con sensor RGB-D (Wilbert G. Aguilar et al., 2017, 2017), y cuando las capturamos mediante dispositivos (cámaras fotográficas o de vídeo, pantallas de rayos X, etc.) obtenemos imágenes bidimensionales (2D). Esta proyección a una dimensión inferior supone una gran pérdida de información. Por ello, dotar a los ordenadores de la capacidad de ver no es una tarea fácil (José Muñoz Pérez, 2010a).

Uno de los métodos menos costosos en cuanto a implementación y computo, son los sensores monoculares que dan como respuesta imágenes bidimensionales, las cuales muestran características claves como variación de textura, desenfoque, degradación, como estimar profundidad con redes neuronales convolucionales (Wilbert G. Aguilar, Quisaguano, et al., 2018). Relacionar puntos tridimensionales del mundo a puntos bidimensionales de la imagen se le conoce como Calibración de la Cámara. En visión monocular el proceso de Calibración es la base para los procesos de recuperación de información del entorno del robot. Para lo cual una escena es proyectada de forma ortográfica resultando en puntos que dependen directamente con la distancia focal y distancia a la imagen.

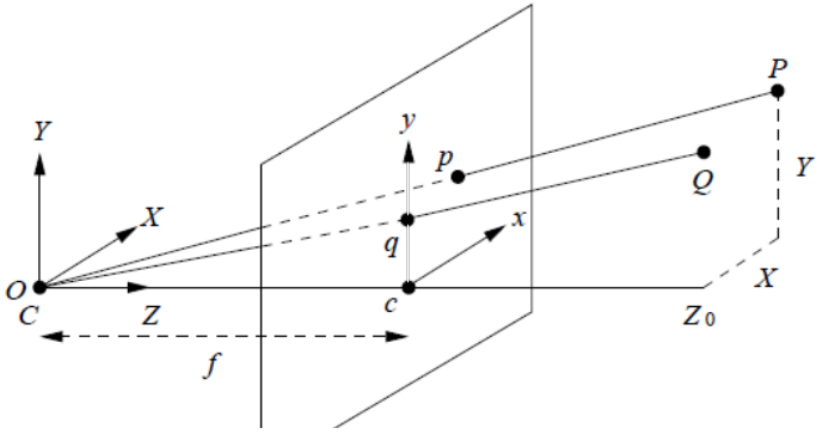

***Figura 5*. Proyección ortográfica a escala**
Fuente: (Kheng, s/f)

En la Figura 5 se aprecia el modelo de una cámara por medio de proyección escalada, siendo los puntos P y Q pertenecientes al mundo real, proyectados en el plano paralelo con coordenadas (x,y) respecto a la distancia focal de la cámara. Los pintos correspondientes se definen por:

$$\boldsymbol{x = sX \ ; y = sY \ ; s = \frac{f}{Z_o}} \tag{1}$$

2.3.3 Imágenes Monoculares obtenidas del UAV

La carga útil de cualquier UAV estará determinada por los objetivos de las tareas de monitoreo a realizar, pero comúnmente se hallan equipados con geoposicionadores (GPS), cámaras de alta resolución, multiespectrales y/o termográficas, videocámara frontal (FFC), sensor de luminosidad incidente (ILS), equipo de radioenlace, conectividad con red de telefonía celular (Procisur & Di Leo, 2015). Las cámaras a bordo de UAVs más comunes son monoculares, con transmisión directa a una estación terrena, como puede ser el caso de un Smartphone, o un computador.

2.3.3.1 Representación y Formación de una Imagen

Podríamos definir una imagen como una función bidimensional *f(xy)* donde x e y son las coordenadas espaciales, y el valor de *f* en cualquier par de coordenadas *(x,y)* es la intensidad de la imagen en dicho punto (José Muñoz Pérez, 2010b). Aquí el término imagen se refiere a la función bidimensional *f(x,y)* que representa la intensidad luminosa en el punto *(x,y)*, y viene dada por:

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{i}(\boldsymbol{x},\boldsymbol{y}) \times \boldsymbol{r}(\boldsymbol{x},\boldsymbol{y}) \qquad (2)$$

Para procesar una imagen por ordenador se debe discretizar (digitalizar) tanto espacialmente como en intensidad luminosa. La digitalización de la coordenada espacial *(x,y)* se llama muestreo de la imagen y la digitalización de la intensidad luminosa se llama cuantificación (Salcedo Peña, 2018). Si la imagen es rectangular y la rejilla cuadriculada se obtiene un conjunto de $M \times N$ valores discretos que corresponden a los tonos de gris de cada uno de los píxeles obtenidos por el proceso de muestreo, expresados por la siguiente matriz:

$$\begin{pmatrix} \boldsymbol{f(1,N)} & \cdots & \boldsymbol{f(M,N)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{f(1,1)} & \cdots & \boldsymbol{f(M,1)} \end{pmatrix} \quad (3)$$

Las imágenes se forman sin el uso de un objetivo/lente, se utiliza un pequeño agujero entre los planos paralelos, permitiendo le paso de un solo fotón de luz de manera que el objetivo se refleja de forma invertida como se observa en la Figura 6.

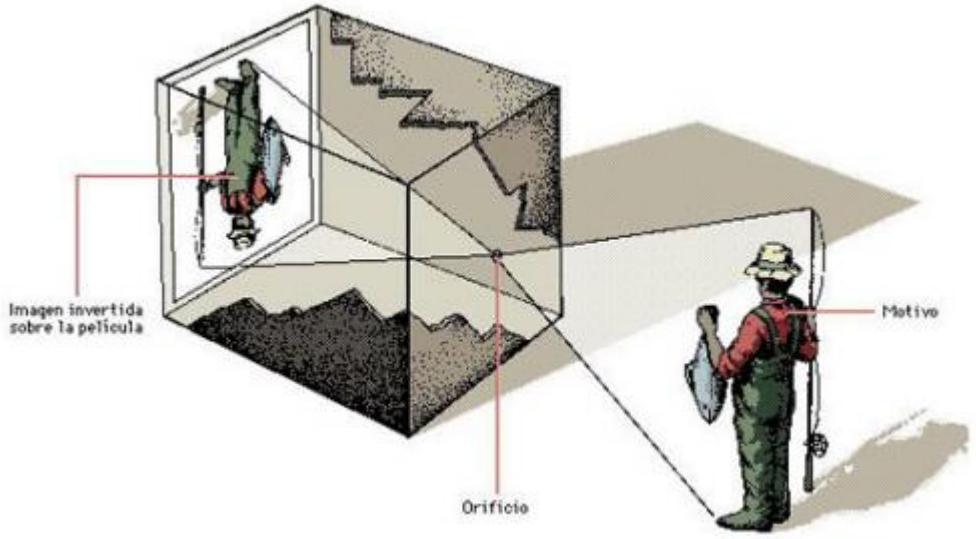

Figura 6. **Formación de una Imagen con el modelo Pin-Hole**

2.3.3.2 Modelo de Cámara

El modelo de la cámara Pinhole describe la relación matemática que existe entre las coordenadas de un punto 3D y su proyección en el plano de la imagen, este modelo de cámara puede ser utilizado como una aproximación de primer orden en el trazo de un mapa de una escena 3D a una imagen 2D (Emmanuel Jiménez Camacho, 2009). Este modelo es frecuentemente utilizado por sistemas basados en visión por computador.

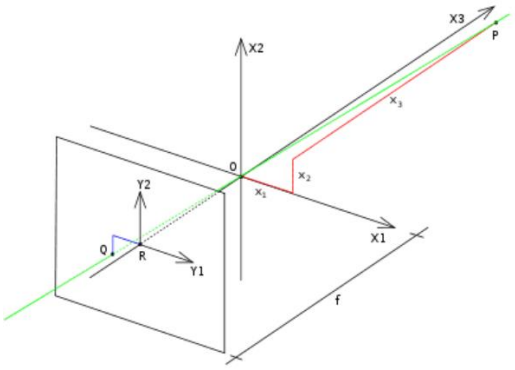

Figura 7. **Modelo de la cámara Pin-hole**
Fuente: (Shapiro, 2000)

Cómo se aprecia en la Figura 7 un sistema de coordenadas 3D con origen en *O*, Los ejes de este sistema son *X1, X2, X3*, siendo este último el que apunta la dirección de vista de la cámara que pasa por el eje óptico. El plano proyectado es paralelo a los ejes *X1* y *X2* a una distancia focal *f*, El punto *R* es la intersección del eje óptico y el plano de la imagen. El punto *Q* es la proyección del punto *P* en el plano. Las coordenadas *Y1*,*Y2* del punto *Q* son relativas al sistema de coordenadas *X1*, *X2*, *X3*, con origen en R.

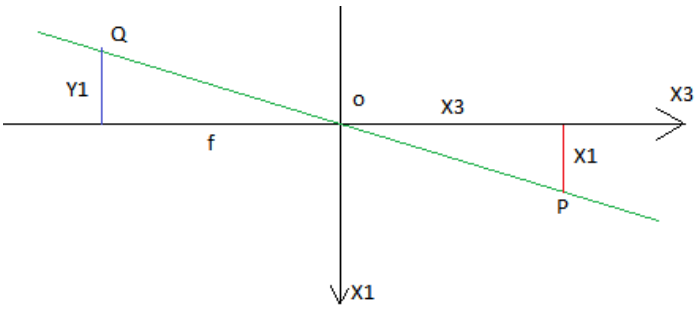

Figura 8. **modelo cámara pinhole en perspectiva X2**

En la Figura 8 es posible distinguir dos triángulos similares ambos compartiendo la línea de proyección como hipotenusa Los catetos del triángulo del lado izquierdo son —*y*, y *f* y los catetos del triángulo del lado derecho son *x*, y *x3*. Como estos triángulos son similares entonces se puede decir que:

$$-\frac{y_1}{f} = \frac{x_1}{X_3} \tag{4}$$

En perspectiva de la dirección negativa del eje *X1*, sería

$$-\frac{y_2}{f} = \frac{x_2}{X_3} \tag{5}$$

En resumen, quedaría así:

$$\frac{y_1}{y_2} = -\frac{f}{x_3} \cdot \frac{x_1}{X_2} \tag{6}$$

Esta es una expresión que describe la relación entre las coordenadas del punto *P* y el punto *Q*. si se desea trabajar con la imagen derecha solo resta suprimir el signo negativo:

$$\frac{y_1}{y_2} = \frac{f}{x_3} \cdot \frac{x_1}{X_2} \tag{7}$$

2.3.3.3 Métodos de Caracterización de Imágenes

El poder segmentar las características correctas en una imagen, es primer paso fundamental para el seguimiento, comenzando con la estabilización de video (W. Aguilar & Angulo Bahón, 2014; Wilbert G. Aguilar & Angulo Bahón, 2013; Wilbert G Aguilar & Angulo, 2014b, 2014c) y compensación de efectos generados en la imagen por el movimiento (W G Aguilar & Angulo, 2012; Wilbert G Aguilar & Angulo, 2012) . Por ejemplo, el color se utiliza como una característica para las representaciones del aspecto basado en histograma, mientras que para la representación basada en contorno, objeto bordes se utilizan por lo general como características (Yilmaz et al., 2006).

En general, existen dos métodos para representar imágenes, a saber, características globales y características locales. En la representación global de características, la imagen está representada por un multidimensional (Hassaballah, Abdelmgeid, & Alshazly, 2016). Las características de la imagen local (también conocidas como puntos de interés, puntos clave y características salientes) se pueden definir como un patrón específico que se distingue de sus píxeles inmediatamente cercanos, que generalmente se asocia con una o más de las propiedades de la imagen (Salahat & Qasaimeh, 2017). Tomando en cuenta el costo computacional, no es necesario localizar todas las características locales de una imagen, puesto que representaría un gran impacto en el rendimiento general del sistema. Salahat (Salahat & Qasaimeh, 2017)en su artículo establece que las características ideales deben tener las siguientes cualidades mostradas en la Tabla 1:

Tabla 1

Características ideales recomendadas en una imagen

Distinción	Los patrones de intensidad deben tener variaciones suficientes que permitan distinguir las características y combinarlas.
Localidad	Deben ser locales para reducir la posibilidad de oclusión permitiendo una estimación simple entre deformaciones geométricas entre dos frames
Cantidad	El número de características detectadas deben ser suficientes, no demasiadas
Precisión	Las características deben ubicarse con precisión respecto a la escala, forma y posición
Repetibilidad	Un alto porcentaje de características detectadas deben mantenerse entre dos cuadros consecutivos
Invarianza	En un escenario con deformaciones como escala, rotación. Las deformaciones deben estar modeladas matemáticamente.

Los detectores de características se pueden clasificar en tres categorías: detectores de escala única, detectores de escala múltiple y detectores invariantes afines. Los detectores de una sola escala son invariables a las transformaciones de la imagen, como la rotación, la traducción, los cambios en las iluminaciones y la adición de ruido. Sin embargo, son incapaces de lidiar con el

problema de escalado. Dadas dos imágenes de la misma escena relacionadas por un cambio de escala, queremos determinar si se pueden detectar o no los mismos puntos de interés (Hassaballah et al., 2016). Algunos de los métodos que serán tomados en cuenta para el presente proyecto de investigación son el Detector de Harris (Harris & Stephens, s/f), Detector FAST (Rosten & Drummond, 2005), Detector Hessiano (BEAUDET & R., 1978).

El Detector de Harris es una combinación de detección de bordes y esquinas, al tener variación de intensidad resulta un mejor detector en términos de repetibilidad y detección. Se basa en una matriz de correlación simétrica de 2x2, describe sus estructuras locales representados dela siguiente forma:

$$\boldsymbol{M(x,y)} = \sum \boldsymbol{u,v}\, \boldsymbol{w(u,v)} * \begin{bmatrix} \boldsymbol{I_x^2\,(x,y)} & \boldsymbol{I_x I_y\,(x,y)} \\ \boldsymbol{I_x I_y (x,y)} & \boldsymbol{I_y^2\,(x,y)} \end{bmatrix} \tag{8}$$

donde I_x e I_y son derivadas de imágenes locales en las direcciones $\boldsymbol{x}$ e y respectivamente, y $\boldsymbol{w(u,v)}$ denota una ventana de ponderación sobre el área $\boldsymbol{w(u,v)}$. Un diagrama del funcionamiento del detector se observa en la .

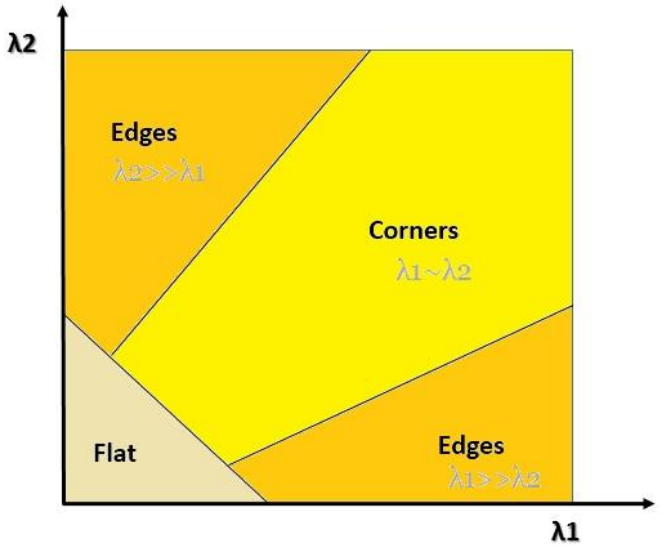

Figura 9. **Clasificación de puntos de imagen en función de los valores propios de la matriz de autocorrelación M**
Fuente: (Fabio Nelli, 2017)

Detector FAST (Características de prueba de segmento acelerado) se detectan los puntos a partir de segmentos de pruebas considerando un circulo de 16 pixeles alrededor de cada pixel candidato a ser esquina como base de cálculo. Si un conjunto de n píxeles contiguos en el círculo con un radio *r* es más brillante que la intensidad del píxel candidato (indicado por *Ip*) más un valor de umbral *t*, *Ip* + *t*, o todos más oscuros que la intensidad del píxel candidato menos el valor umbral *Ip* - *t*, entonces *p* se clasifica como una esquina. Se puede utilizar una prueba de alta velocidad para excluir un gran número de puntos que no son de esquina; la prueba examina solo los cuatro píxeles 1, 5, 9 y 13. Una esquina solo puede existir si tres de estos píxeles de prueba son más brillantes que *Ip* + *t* o más oscuros que *Ip* - *t* y el resto de píxeles se examinan para la conclusión final (Hassaballah et al., 2016). La Figura 10. Detección de Características de una imagen usando FAST ilustra el proceso, donde los cuadrados resaltados son los píxeles utilizados en la detección de la esquina.

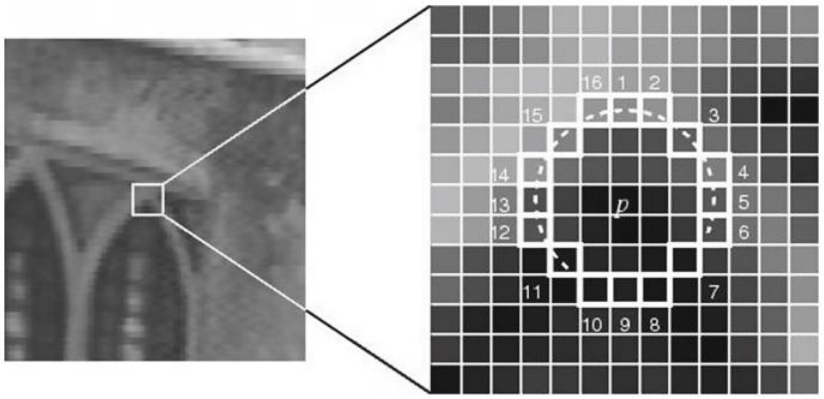

Figura 10. **Detección de Características de una imagen usando FAST**

El detector de Hess en una matriz de 2x2 de derivadas de segundo orden de intensidad de imagen *I(x,y),* con esta matriz se analizan estructuras de imágenes locales expresadas en la forma en que Ixx, Ixy, e Iyy son derivadas de imágenes de segundo orden calculadas utilizando la función gaussiana de la desviación estándar σ.

$$\boldsymbol{H(x,y,\sigma)} = \begin{bmatrix} \boldsymbol{I_{xx}(x,y,\sigma)} & \boldsymbol{I_{xy}(x,y,\sigma)} \\ \boldsymbol{i_{xy}(x,y,\sigma)} & \boldsymbol{I_{yy}(x,y,\sigma)} \end{bmatrix} \tag{9}$$

Para detectar características de interés, busca un subconjunto de puntos donde las respuestas derivadas son altas en dos direcciones ortogonales. Es decir, el detector busca puntos donde el determinante de la matriz de Hess tiene un máximo local (Hassaballah et al., 2016).

$$\boldsymbol{det(H) = I_{xx}I_{yy} - I_{xy}^2} \tag{10}$$

2.4 Detección y Seguimiento de Objetos

Entre las prioridades de la visión por computador es el seguimiento de objetivos (target) La aparición de computadores de alto rendimiento, la disponibilidad de cámaras de vídeo de bajo costo y alta calidad, y la creciente necesidad de un análisis automatizado de vídeo ha generado un gran

interés para el seguimiento. En el análisis de video existen 3 pasos necesarios: detección de objetos en movimiento interesantes, el seguimiento de este tipo de objetos de cuadro a cuadro, y el análisis del objeto realiza un seguimiento de reconocer su comportamiento (Yilmaz et al., 2006).

2.4.1 Descripción general y especificaciones de un objeto

Un objeto puedes ser definido por sus características de interés que puedan ser analizadas o procesadas. Por ejemplo, un vehículo en la carretera, peatones (Wilbert G. Aguilar et al., 2017, 2017; Wilbert G. Aguilar, Luna, et al., 2017; Andrea, Byron, Jorge, Inti, & Aguilar, 2018) en la acera, aviones en el cielo, entre otros. A continuación, se describirá las representaciones de objetos más empleadas para el seguimiento.

El objeto se representa por un punto, por lo general, es el centroide como se observa en la Figura 11(a), o por un conjunto de puntos, así como la Figura 11(b).

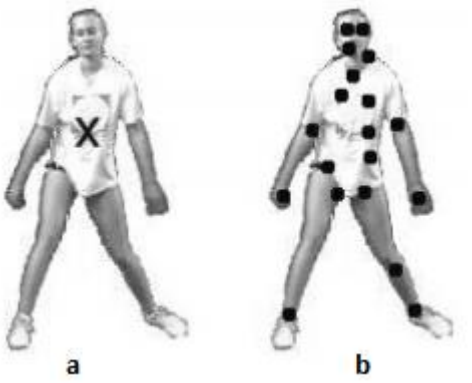

Figura 11. **Representación del objeto: centroide (a), o conjunto de puntos (b)**

La forma del objeto puede ser encapsulado en formas geométricas (Ver Figura 12) como, un rectángulo, una elipse, generalmente se modela para proyecciones (Homografía). Aunque las formas geométricas primitivas son más adecuadas para representar objetos rígidos simples, también

se usan para rastrear objetos no rígidos, por motivos de rapidez o rendimiento. Un modelo articulado del objeto es una opción para representar partes del cuerpo.

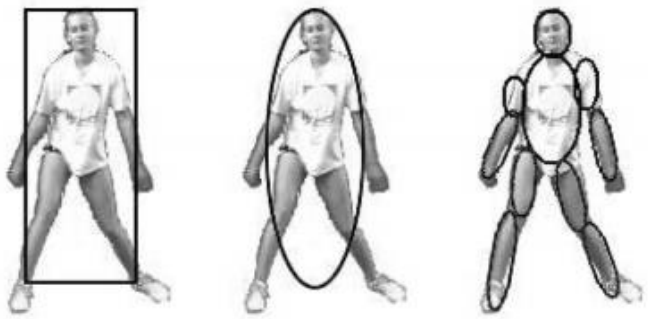

Figura 12. **Representación geométrica de un objeto**

La descripción de un objeto a basado en el contorno define la silueta del objeto de interés, las representaciones de la silueta y el contorno son adecuadas para el seguimiento de complejos no rígidos Formas, ver en Figura 13.

Figura 13. **Representación de un objeto por contornos**

En general, hay una fuerte relación entre las representaciones de objeto y los algoritmos de seguimiento. Para el seguimiento de objetos, que aparecen muy pequeños en una imagen, la representación de punto suele ser apropiado. Para los objetos cuyas formas se puede aproximar por rectángulos o elipses, primitivas representaciones de forma geométrica son más apropiados. Para el seguimiento de objetos con formas complejas, por ejemplo, seres humanos, un contorno o una representación basada Silueta es apropiado.

2.4.2 Detección de Características ideales para el Seguimiento de Objetos

El propósito del seguimiento de objetos es determinar la posición del mismo en las imágenes de forma continua y confiable, por ejemplo en (Wilbert G. Aguilar, Quisaguano, et al., 2018) se logra detectar objetos e partir de redes neuronales convolucionales. Para lograr el seguimiento se presenta 2 algoritmos relevantes que logran acorde al proyecto, con el objetivo.

2.4.2.1 Transformación de características invariantes de escala (SIFT)

Estos puntos de características aparecen invariables a cualquier escala, rotación o traducción de las imágenes. Por lo tanto, las características de SIFT se pueden integrar con un sistema de seguimiento establecido para mejorar el rendimiento (Zhou, Yuan, & Shi, 2009). Consta de 4 etapas que minimizan el costo computacional realizando grandes operaciones en ubicaciones específicas de un filtro en cascada(Sayem, 2016).

La primera etapa identifica las ubicaciones clave en el espacio de escala buscando ubicaciones que sean máximas o mínimas de una función de diferencia de Gauss. Cada punto se usa para generar un vector de características que describe la región de imagen local muestreada en relación con su marco de coordenadas de espacio de escala. Las características mantienen una invarianza parcial a las variaciones locales, como afines o proyecciones 3D, al difuminar las ubicaciones de gradiente de imagen. Los vectores de características resultantes se denominan teclas SIFT. La extracción de características se puede calcular de manera muy eficiente mediante la construcción de una pirámide de imágenes con un nuevo muestreo entre cada nivel (Zhou et al., 2009).

El espacio de escala de una imagen sería $L(x, y, \sigma_L)$, resultante de la convolución de un Gaussiano de escala variable, $G(x, y, \sigma_L)$ para una imagen $I(x, y)$. Donde:

$$\boldsymbol{G(x, y, \sigma_L) = \frac{1}{2\pi\sigma_L^2} \exp^{-(x^2+y^2)/\sigma_L^2}} \tag{11}$$

Las ubicaciones de puntos clave estables en el espacio de la escala se pueden calcular a partir de la diferencia de gaussianos separados por un escalar multiplicativo constante $\boldsymbol{S}$:

$$\boldsymbol{G(x, y, \sigma_L) = L(x, y, S\sigma_L) - L(x, y, \sigma_L)} \tag{12}$$

La función de la diferencia de Gauss se puede tratar como una aproximación al Laplaciano de escala normalizada del Gaussiano $\sigma_L^2 \nabla^2 G$ (Lindeberg & Lindeberg, 1994). D puede vincularse a $\sigma_L^2 \nabla^2 G$ mediante la siguiente ecuación:

$$\boldsymbol{\sigma_L^2 \nabla^2 G = \frac{\partial G}{\partial \sigma_L}} \tag{13}$$

Entonces:

$$\boldsymbol{G(x, y, S\sigma_L) - G(x, y, \sigma_L) \approx (s - 1)\sigma_L^2 \nabla^2 G} \tag{14}$$

Esto indica que la función de diferencia de Gauss tiene escalas que difieren en un factor constante, mientras que incorpora la normalización de la escala $\boldsymbol{\sigma_L^2}$ requerida para el Laplaciano invariante de escala.

2.4.2.2 Algoritmo de SHI y TOMASI

Shi y Tomassi definen un conjunto de puntos con características singulares que se basan principalmente en propiedades específicas de los bordes y texturas como lo es la intensidad espacial

(Mora Mariño & Páez Melo, 2012). Para determinar un conjunto de "buenos puntos para seguir", se parte del desplazamiento de dichos puntos dentro en una ventana de tamaño fijo.

A medida que la cámara se mueve, los patrones de intensidad de la imagen cambian de forma compleja. Sin embargo, lejos de los límites de oclusión y las marcas cercanas a la superficie, estos cambios a menudo se pueden describir como movimiento de imagen:

$$\boldsymbol{I(x, y, t + \tau) = I(x - \varepsilon(x, y, t, \tau), y - \eta(x, y, t, \tau))} \quad (15)$$

Esto quiere decir que una imagen que pertenece al tiempo $t + \tau$, se puede obtener al desplazar en una cantidad adecuada, cada punto de la imagen actual, perteneciente al tiempo t. La cantidad de movimiento $\delta = (\varepsilon, \eta)$ es llamada *desplazamiento* del punto en $X = (x, y)$. El vector de desplazamiento δ es una función de la posición de imagen x, y las variaciones en δ a menudo son notables incluso dentro de ventanas pequeñas utilizadas para el seguimiento (Jianbo Shi & Tomasi, 1994). Un campo de movimiento afín en una mejor representación sería:

$$\boldsymbol{\delta = Dx + d} \quad (16)$$

Donde

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{d_{xx}} & \boldsymbol{d_{xy}} \\ \boldsymbol{d_{yx}} & \boldsymbol{d_{yy}} \end{bmatrix} \quad (17)$$

Es la matriz de deformación, y d es el desplazamiento del centro de la ventana de interés.

Con todas estas informaciones, la función encuentra esquinas en la imagen. Se rechazan todas las esquinas por debajo del nivel de calidad. A continuación, clasifica las esquinas restantes según la calidad en orden descendente. Entonces la función toma la primera esquina más fuerte, tira todas

las esquinas cercanas en el rango de distancia mínima y devuelve N esquinas más fuertes. El resultado del algoritmo se puede observar en la Figura 14.

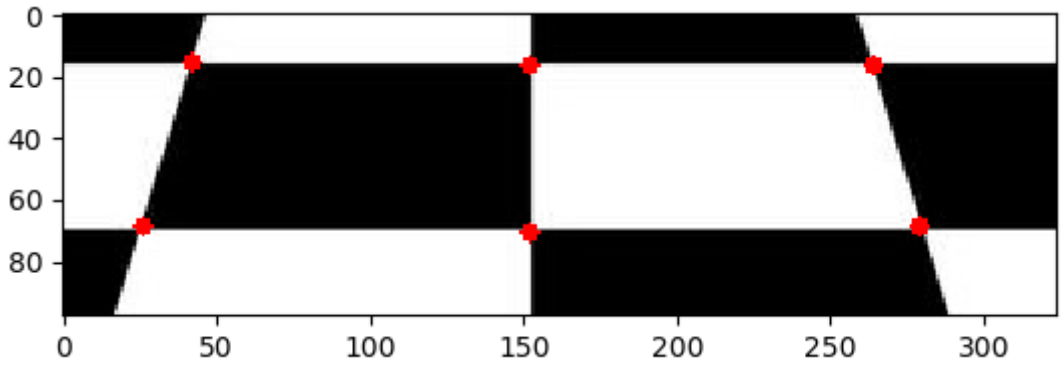

Figura 14. **Detección de Esquinas con algoritmo Shi y Tomasi**

2.4.2.3 Flujo óptico

Reconocer el movimiento de objetos vistos en una secuencia de imágenes es un problema, que se ha tratado de resolver usando varios modelos que estiman las características de dicho movimiento. Dadas dos imágenes secuenciales tomadas en tiempos diferentes t_0 y t_1, es posible describir el desplazamiento de cada partícula o característica local de la imagen en el intervalo de tiempo $[t_0, t_1]$, sí es que se tiene una forma de reconocer cada una de las características locales de las dos imágenes (Ocegueda, 2003). Se asume que las características locales se mantienen a lo largo del movimiento, por lo tanto, se puede estimar el campo vectorial que describe el movimiento de cada característica en la imagen, esto es, Flujo Óptico

Una definición de flujo óptico puede ser el desplazamiento aparente de las características claves o patrones de intensidad de una imagen, al decir que es un desplazamiento aparente se asume que aquellos patrones cambian muy poco en el tiempo. En situaciones en las cuales el movimiento de los objetos implica un movimiento de sus patrones de intensidad en el plano de la imagen, el flujo

óptico puede relacionarse directamente con el movimiento de los objetos en la escena (Mora Mariño & Páez Melo, 2012).

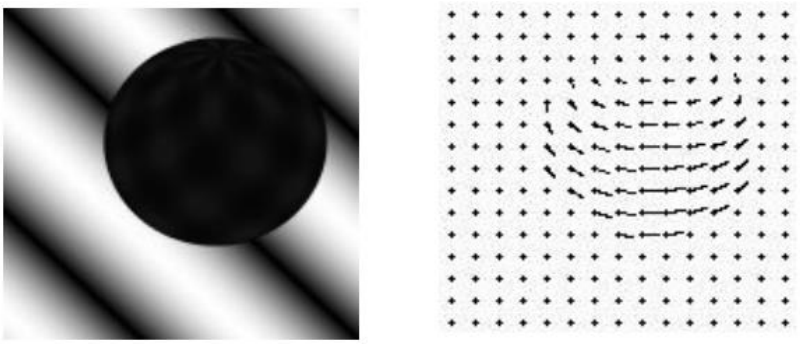

Figura 15. **Representación de movimiento en una escena**

En referencia a la Figura 15, una esfera gira en el sentido de las agujas del reloj, entre cada una de las secuencias de imágenes se realiza una estimación de Flujo Óptico. El campo vectorial que describe el movimiento se observa en imagen derecha, cada vector indica el sentido y velocidad de cada punto característico extraído de la imagen.

La mayoría de las técnicas existentes para la estimación del flujo óptico se puede clasificar en 4 categorías: las basadas en gradientes espacio temporales, las basadas en comparación de regiones, las basadas en fase y las basadas en energía.

- Métodos diferenciales: Estos métodos se limitan a que las derivadas se queden en el dominio de la imagen, dependiendo de la información de las derivadas (Barron et al., 1994). Eestableció:
    - Métodos Locales: utilizan la información en una vecindad alrededor de un píxel para estimar su movimiento. El método más representativo de esta familia es el de Lucas-Kanade (Lucas & Kanadc, s/f). Este método calcula el desplazamiento

a partir de la militarización de la ecuación del flujo óptico alrededor de una ventana centrado en un píxel.

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I(x,y,t)}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I(x,y,t)}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I(x,y,t)}{\partial t} = 0 \quad (18)$$

- **Métodos variacionales**: Una de las ventajas que ofrecen es que todas estas restricciones están presentes en la energía; no existe ningún tipo de suposición adicional que no se refleje en el modelo y si en la implementación (Riascos Segura, Cardona Gallego, & Electrónico, 2015). A diferencia de otro tipo de técnicas las estimaciones son densas por lo que no es necesario realizar ningún proceso de interpretación. La solución se obtiene directamente mediante la militarización de esta energía.

$$u^{k+1} = \overline{u}^{k} - \frac{I_x\left(I_x\overline{u}^{k} + I_y\overline{v}^{k} + I_t\right)}{\alpha^2 + I_x^2 + I_y^2} \quad (19)$$

2.4.2.4 Lukas-Kanade

El algoritmo de flujo Óptico Lukas-Kanade es un método que estima el movimiento de las características relevantes o puntos de interés, que pertenecen a imágenes sucesivas de una escena. Este algoritmo fundamenta su funcionamiento en dos suposiciones implícitas:

- El desplazamiento de tiempo entre cada imagen es muy pequeño, por lo tanto, los objetos no se han desplazado de forma significativa.
- Las imágenes representan una escena natural que contiene objetos texturados, que muestran tonos de gris, los cuales cambian suavemente.

El algoritmo no busca en la imagen actual una coincidencia del punto de interés. Su acción consiste en tratar de adivinar en qué dirección se ha movido un objeto de manera que los cambios locales en la intensidad pueden ser explicados (Rojas, 2010).

2.4.2.4.1 Método

Supongamos que vemos una escena a través de un agujero cuadrado. La intensidad *a* visible a través del agujero es variable.

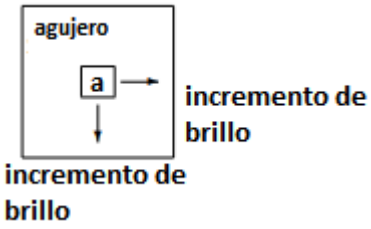

Figura 16. **Representación de desplazamiento de punto característico**

En el siguiente cuadro, la intensidad del píxel ha aumentado a *b*. Sería sensato suponer que se ha producido un desplazamiento del objeto subyacente hacia la izquierda y hacia arriba, de modo que la nueva intensidad *b* sea ahora visible debajo del agujero cuadrado (Rojas, 2010).

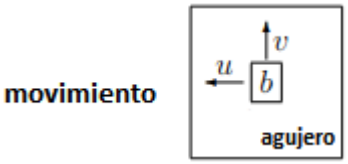

Figura 17. **Punto Caractetrístico desplazado**

Si sabemos que el aumento del brillo por píxel, en el píxel (x, y), $I_x(x, y)$ es la dirección en x, y el aumento del brillo por píxel en la dirección en y, es $I_y(x, y)$, se tiene un incremento total en el brillo, después de un movimiento de u píxeles en la dirección x y v píxeles en la dirección y de:

$$\boldsymbol{I_x(x,y) \cdot u \ + \ I_y(x,y) \ \cdot v} \tag{20}$$

Esto coincide con la diferencia local en intensidad $(b \ - \ a)$ llamada $I_t(x, y)$, de modo que:

$$\boldsymbol{I_x(x,y) \cdot u \ + \ I_y(x,y) \cdot v \ = \ -I_t(x,y)} \tag{21}$$

El signo negativo es necesario para los positivos I_x, I_y, y I_t tenemos un movimiento hacia la izquierda y hacia abajo.

2.4.2.4.2 Técnica de Pixeles Cercanos

El método de Lukas-Kanade toma una región de 3x3 alrededor del pixel a analizar, por lo tanto, los 9 pixeles tienen movimiento (Rojas, 2010). En ese caso se establecen 9 ecuaciones lineales:

$$\boldsymbol{I_x(x+\Delta_x, y+\Delta_y).u + I_y(x+\Delta_x, y+\Delta_y).v = -I_t(x+\Delta_x, y+\Delta_y)} \quad (22)$$

Para: $\Delta_x = -1,0,1$ y $\Delta_y = -1,0,1$

Las ecuaciones lineales se pueden resumir como la matriz:

$$\boldsymbol{S\begin{pmatrix} u \\ v \end{pmatrix} = \vec{t}} \quad (23)$$

Donde $\boldsymbol{S}$ es una matriz de 9x2 que contiene las filas $\left(\boldsymbol{I_x(x+\Delta_x, y+\Delta_y), I_y(x+\Delta_x, y+\Delta_y)}\right)$ y $\vec{\boldsymbol{t}}$ es un vector que contiene los 9 términos: $\boldsymbol{-I_t(x+\Delta_x, y+\Delta_y)}$.

Puesto que la resolución de las 9 ecuaciones con dos variables desconocidas es compleja. Una mejor resolución se obtiene con el método de mínimos cuadrados:

$$\boldsymbol{S^T S\begin{pmatrix} u \\ v \end{pmatrix} = S^T \vec{t}} \quad (24)$$

Invirtiendo $\boldsymbol{S^T S}$ se tiene:

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{pmatrix} = (\boldsymbol{S}^{\boldsymbol{T}}\boldsymbol{S})^{-1}\boldsymbol{S}^{\boldsymbol{T}}\vec{\boldsymbol{t}} \quad (25)$$

Entonces, desde el punto de vista del usuario, la idea es simple, le damos algunos puntos para seguir, recibimos los vectores de flujo óptico de esos puntos. Pero de nuevo hay algunos problemas. Hasta ahora, estábamos tratando con movimientos pequeños, por lo que falla cuando hay un movimiento grande. Para hacer frente a esto utilizamos pirámides. Cuando subimos en la pirámide, los pequeños movimientos se eliminan y los grandes movimientos se convierten en pequeños movimientos. Entonces, al aplicar Lucas-Kanade allí, obtenemos flujo óptico junto con la escala.

2.5 Control de UAVs

Tomando en cuenta que un UAV es un sistema, se puede representar por una función de transferencia (Wilbert G. Aguilar & Angulo, 2016; Wilbert G. Aguilar, Manosalvas, Guillén, & Collaguazo, 2018; Orbea, Moposita, Aguilar, Paredes, León, et al., 2017; Orbea, Moposita, Aguilar, Paredes, Reyes, et al., 2017). Esta es una función que representa y modela la respuesta del sistema en función de la entrada. En la Figura 18 se ve el esquema típico de una función de transferencia $G(s) = \frac{Y(s)}{U(s)}$.

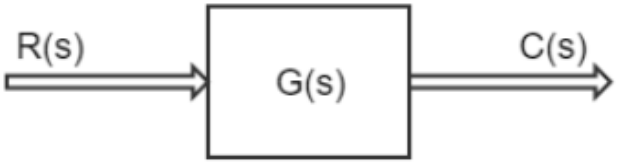

Figura 18. **Función de Transferencia**

El objetivo al controlar un sistema es primero estabilizarlo si es inestable y luego cambiar la posición de los polos hacia una en que la respuesta del sistema satisfaga unos criterios.

La realimentación, que también se puede llamar sistema en lazo cerrado, como en el esquema de la Figura 19, consiste en llevar la salida, o parte de ella, hacia la entrada. El bloque H(s) es la

función de transferencia del sensor que mide la salida Y(s). Se considerará H(s) = 1. La ganancia K se encarga del control de la realimentación.

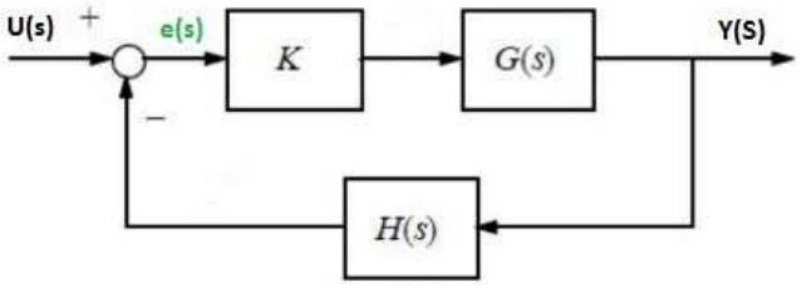

Figura 19. **Esquema general de Retroalimentación**

El sistema de control gestiona la regulación del dispositivo a través de la aplicación de señales de mando en la velocidad de los motores de cuadricóptero (FernándezFern, Directores, Macho, & Luis Porras Gaí an, 2014). La dificultad que se presenta al operar una aeronave incluso a pequeña escala, y las perturbaciones existentes en el medio de vuelo han generado la necesidad de implementar controladores que estabilicen a las aeronaves.

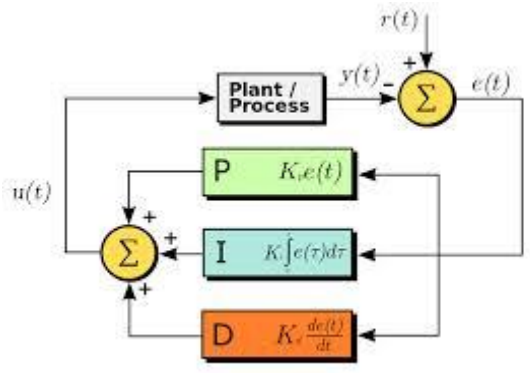

Figura 20. **Esquema de controlador PID**

Es así como los controladores PID rápidamente se volvieron muy usados en esta área debido a su relativa sencilla implementación y ajuste (Figura 20). Existen solamente dos inconvenientes en el uso de éste control, en primer lugar, los controladores PID se pueden aplicar solamente en sistemas lineales, y el cuadricóptero, al ser no lineal, el control se diseña en base a un punto de

operación en equilibrio, de modo que el controlador sea efectivo en una cierta región limitada alrededor de dicho punto (Control Luis Eduardo Romero Paredes, 2014).

La lógica difusa, en las últimas investigaciones ha sido muy utilizada para el seguimiento de trayectorias u objeto. Por ejemplo, en una investigación (Olivares-Mendez, Mejias, Campoy, & Mellado-Bataller, 2012), las entradas del controlador difuso son: el ángulo entre los cuadricópteros y el objeto a evitar, y la segunda entrada es la medida de la evolución de este ángulo entre los dos últimos cuadros. La salida del controlador es el ángulo de orientación deseado que el cuadricóptero necesita girar. En la Figura 21 se aprecia el esquema del controlador difuso.

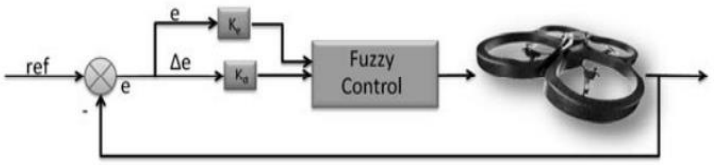

Figura 21. **Esquema de control Difuso para un UAV**

CAPITULO III

DESCRIPCIÓN GENERAL DEL SISTEMA

En este capítulo se describe cada uno de los componentes que son parte del sistema para la evasión de objetos dinámicos. Así como las configuraciones y software necesario para la integración del sistema.

3.1 Especificaciones técnicas del Hardware

El hardware integrado en el sistema consta de: El Vehículo Aéreo No Tripulado, La cámara integrada en el UAV, la estación terrena de procesamiento sería el Computador Portátil, la cual tendría conectividad con el UAV.

3.1.1 Vehículo Aéreo No tripulado (UAV)

El UAV selecciona es un cuadricóptero comercial modelo Bebop 2 (Figura 22), pertenece a la empresa Parrot. Se caracteriza por su peso ligero, tiene una autonomía de 25 minutos, posee una computadora interna que lo convierte en un cuadricóptero auto controlado. Además, su software es parcialmente libre, es decir, posee un SDK para desarrolladores, que facilita el acceso a el control de sus movimientos, y visualizar las imágenes a tiempo real de la cámara integrada a bordo del UAV.

A finales de 2015 Parrot lanzó al mercado el Bebop 2, un modelo mejorado y con más autonomía de vuelo que su predecesor, sin embargo, se rige por el mismo sistema operativo haciendo también posible su control a través del driver *bebop_autonomy* (Valero Lavid, 2017).Este dron se puede pilotar de tres maneras distintas, todas ellas a través de Wifi:

- Utilizando la aplicación *Freeflight* con un alcance máximo de 300m. Esta aplicación está disponible para dispositivos Android e IOS.
- Utilizando el mando *Skycontroler* que aumenta su rango de acción a 2km.
- A través del SDK publicado por Parrot donde el alcance dependerá del tipo de antena que utilicemos en nuestra estación remota.

Figura 22. **Cuadricóptero Modelo Bebop 2, Marca Parrot**
Fuente: (Parrot, 2015)

3.1.1.1 Estructura Física

Parrot ha querido que el Bebop 2 apenas tenga partes móviles, algo que solo queda para las hélices. No hay nada colgando, de hecho, la cámara está incluida en el cuerpo, en la posición del pico de la 'aeronave'. A pesar de su gran angular, no le molestan las hélices en las grabaciones. En sus 500 gr de peso y sus dimensiones 328 X 382 X 89 [mm], ver Figura 23 El cuerpo, construido con la técnica de sinterización por láser selectivo y el innovador material de fabricación aditiva Windform GT, combina excelentes resultados estéticos con una excelente resistencia al impacto y cambios de temperatura.

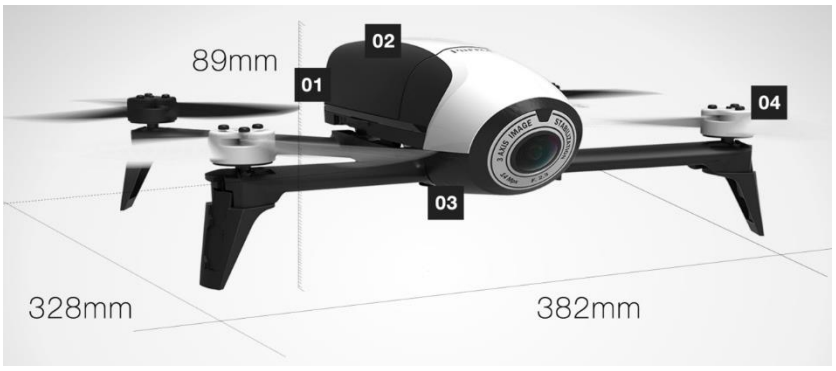

Figura 23. **Estructura Física Parrot Bebop 2**
Fuente: (Parrot, 2015)

Los datos recopilados por siete sensores se analizan y combinan gracias a la impresionante capacidad de cálculo de su computadora a bordo (Kote, 2016). Los sistemas son:

- La estabilización vertical la consigue gracias a que la cámara hace una toma cada 16 milisegundos, que compara con las anteriores para determinar posición y velocidad del Bebop 2.
- Un sensor de ultrasonidos en la zona inferior del drone, mide la altura del suelo en un rango de 5 metros.
- El sensor anterior se ve acompañado por un sensor de presión que le determina la altura cuando subimos de esa franja.
- Un giroscopio de 3 ejes para determinar el ángulo en el que se encuentra el dron.
- También hay un acelerómetro que ayuda con la posición y velocidad lineal de vuelo.
- Una brújula digital posiciona al dron en el espacio.
- Cuenta con sistema GPS y GLONASS para geolocalizar el equipo.

3.1.1.2 Sensor (Cámara) a bordo del UAV

El Bebop 2 de Parrot viene con una cámara Full HD que toma fotografías a 14mp y vídeos con una resolución de 1920x1080p. Además, el gran ojo de pez con el que viene equipado nos proporciona imágenes con mayor luz y nitidez incluso en condiciones de baja luminosidad.

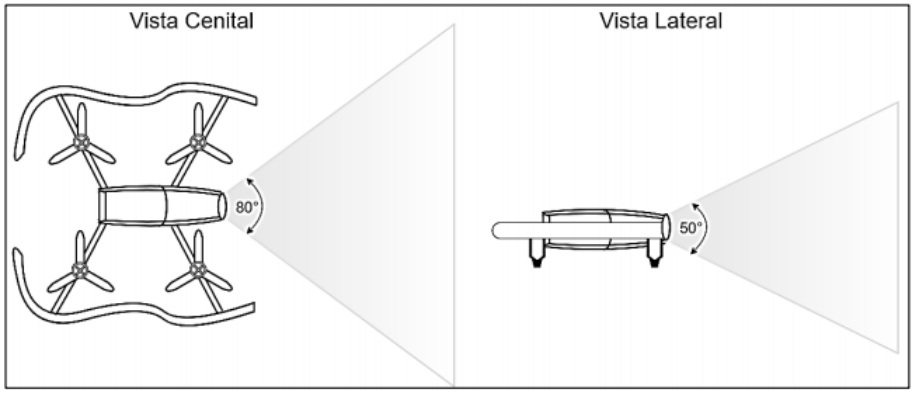

Figura 24. **Ángulos de apertura de video para la cámara del Bebop 1 y Bebop 2 desde la perspectiva cenital y lateral.**
Fuente: (Salcedo Peña, 2018)

El ángulo de apertura que se presenta en video para la cámara del drone (Ver Figura 24) es de aproximadamente 50° en el eje vertical (ordenadas) y de 80° en el eje horizontal (abscisas), así el rango de visión varía en función de la distancia de la cámara respecto al objetivo.

Además el drone retorna un video completamente rectificado y sin distorsiones debido a los algoritmos que lleva incorporados que se encargan de suprimir la distorsión de perspectiva y mantienen un modelo proyectivo simple que conserva las propiedades de la escena captada traduciéndolas en imágenes 2D estables (Salcedo Peña, 2018).

3.1.2 Terminal Terrestre de Procesamiento

Para que sea posible la conexión con el dron vía Wifi es necesario una estación de control remota, formada en este caso por un ordenador como en la Figura 25. El ordenador podrá enviar ordenes de control al cuadricóptero gracias a las herramientas disponibles que serán configuradas en el mismo.

Figura 25. **Esquema de Conexión Estación Remota con UAV**
Fuente: (Valero Lavid, 2017)

El computador constará con un Sistema Operativo Linux, Ubuntu 16.04 de 64 bits.

Tabla 2
Especificaciones técnicas de la Estación Terrena

Marca	SONY VAIO
Modelo	SVS151A11
Procesador	Intel® Core™ i5 2.50GHz x 4 Núcleos
Memoria RAM	6 GB
Arquitectura	64 bits

3.2 Especificaciones Técnicas del Software

Gracias a las prestaciones que ofrece el Software libre junto con el Kit de Desarrollo de Software (SDK) proporcionado por Parrot. Se desarrollará en Sistema Operativo Ubuntu 16.04 de 64 bits junto con librería de procesamiento de Imágenes OpenCV, la última versión 3.1.0 instalada, ofrece una gama extensa de métodos y algoritmos para procesamiento de imágenes.

3.2.1 Sistema Operativo para Robots (ROS)

ROS es un sistema de código abierto, construido desde cero para fomentar el desarrollo de software de robótica colaborativa. La idea era que cada grupo de expertos que trabajase con ROS pudiese aportar sus partes de software y que pudiesen ser utilizadas por cualquier otra persona en cualquier parte del mundo (Valero Lavid, 2017).

ROS trabaja como si se tratase de un sistema operativo, ya que proporciona servicios estándares tales como abstracción del hardware, comunicación a través de mensajes entre procesos, mantenimiento de paquetes, control de dispositivos de bajo nivel e implementación de funcionalidad que son propios de éstos (Valero Lavid, 2017). Se habla de que trabaja como un sistema operativo, pero en realidad no lo es, ya que se instala sobre otro, generalmente Linux, y por ello también recibe el nombre de Meta-Sistema Operativo. Detalles para la instalación de ROS en (Mani Monajjemi, 2015b).

Para este proyecto se desarrollará el sistema en ROS, debido a que posee un driver para el control del dron Parrot Bebop 2, basado en el ARDroneSDK3 oficial de Parrot. El SDK ayudará a conectarse, pilotear, recibir transmisiones, guardar y descargar medios (fotos y videos), enviar y jugar planes de vuelo de piloto automático y actualizar su avión no tripulado(Parrot Developers, s/f).

3.2.1.1 Conceptos Básicos

Los conceptos básicos que son parte de la red de procesos de ROS son:

- Nodos (*Node*): Los nodos son los encargados de realizar la computación propiamente dicha. ROS está estructurado modularmente de forma que un robot

puede comprender muchos nodos que controlen distintas partes del robot, por ejemplo, un nodo específico para el movimiento del robot, otro para las distintas articulaciones, y así sucesivamente. Un nodo de ROS está escrito usando una biblioteca específica de cliente, tales como *roscpp (C++) o Rospy (Python)* (Valero Lavid, 2017) .

- Maestro (*Master*): Es el nodo principal. El Master proporciona el registro de nombres y de consulta para la computación gráfica. Sin éste, los nodos son incapaces de trabajar ya que no encontrarían los mensajes necesarios o los servicios que una aplicación requiera. El Master, en general, se ejecuta mediante el terminal con el comando roscore, aunque hay excepciones en las que se ejecuta sólo mediante la carga de archivos *launch* (Valero Lavid, 2017).
- Servidor de Parámetros (*Parameter Server*): Almacena datos en una localización centralizada.
- Mensajes (*Messages*): son los datos que se transmiten entre nodos. Los mensajes son una estructura de datos, que se caracteriza por su tipo de formato.
- Temas (*Topics*): Los temas viene a ser el canal por el cual los nodos comparten información mandando y recibiendo mensajes, estos mensajes son publicados con un nombre especial llamado *Topics*. Los nodos tienen el permiso para publicar o suscribirse a un *Topic*. Un nodo puede suscribirse a distintos *Topics*.
- Servicios (*Services*): Los servicios proporcionan una comunicación mediante un método de pregunta/respuesta. Es decir, está formado por dos partes: un mensaje que es el que pregunta y otro mensaje que es el que responde. Cuando un nodo

quiere llamar a un servicio, tiene que llamarle y esperar la respuesta. A diferencia de la comunicación de los *Topics*, un nodo sólo puede llamar a un servicio bajo un nombre particular (Valero Lavid, 2017).

3.2.1.1.1 Comunicación con UAV

El Sistema Operativo para Robots contienes varias herramientas para la conexión y control del Cuadricóptero. Para este proyecto es necesario entender algunos conceptos que se describen en Figura 26:

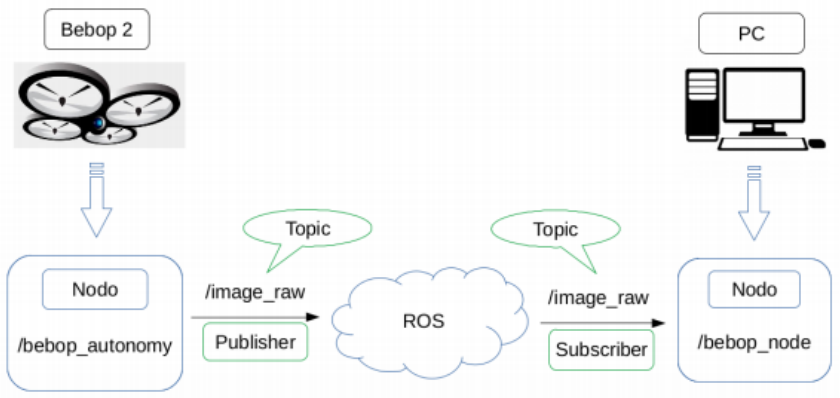

***Figura 26*. Esquema de conexión PC - Bebop 2 a través de ROS**

ROS se basa en una arquitectura *Master-Slave*, en la que es necesario ejecutar ROS en un dispositivo (El computador portátil será el *Master*), y luego en todos los demás dispositivos en los que se ejecute un nodo se conectarán al Master (*Slaves*), el cuadricóptero será el Slave (Valero Lavid, 2017).

Los nodos se comunican entre sí transmitiendo parámetros en streaming mediante *Topics*, servicios de llamada a procedimiento remoto (RPC) y un servidor de parámetros, estos nodos están diseñados para operar a bajo nivel, por tanto, los sistemas de control de un robot normalmente

compondrán muchos nodos. Por otro lado, a través de cada *Topic* se envía un tipo de información con un tipo de mensaje concreto (Valero Lavid, 2017).

Así pues, se tendrá por ejemplo un *Topic* llamado *cmd_vel* en el que se publicarán las velocidades que se quieran dar al dron. Los mensajes son del tipo *geometry_msgs/Twist.msg* y los mensajes *Twist.msg* a su vez están formados por dos vectores de tres componentes cada uno que indican la velocidad lineal y angular en los ejes X, Y y Z.

3.2.1.2 Kit de Desarrollo de Software (SDK) para UAV

En la Wiki de ROS se encuentra el driver basado en la SDK oficial de Parrot ARDroneSDK3 (Mani Monajjemi, 2015a). Este controlador ha sido desarrollado en el Laboratorio de Autonomía de la Universidad Simon Fraser por Mani Monajjemi y otros colaboradores. Este software es mantenido por Sepehr MohaimenianPour (AutonomyLab, Simon Fraser University), Thomas Bamford (Dynamic Systems Lab, University of Toronto) y Tobias Naegeli (Advanced Interactive Technologies Lab, ETH Zürich).

Este driver permite controlar los movimientos básicos del cuadricóptero como despegue, elevación, aterrizaje, girar, navegar. Luego de instalar ROS compatible con la distribución de Linux correspondiente 16.04, La instalación del driver *bebop_autonomy* se detalla en el GitHub de Parrot ARDrone3 SDK (Mani Monajjemi, 2015b).

Una vez instalado, el archivo ejecutable *launch* y el de configuración *config* se encuentra dentro de las carpetas del driver. Al ejecutar el archivo *.launch,* el driver inicia con la configuración predeterminada adjunta en el archivo de configuración *.yaml* que se encuentra dentro de la carpeta config. En la Figura 27 se aprecia algunos de los parámetros del archivo .yaml

```
PilotingSettingsMaxAltitudeCurrent: 1.5
PilotingSettingsMaxTiltCurrent: 5.0
PilotingSettingsMaxDistanceValue: 10.0
PilotingSettingsNoFlyOverMaxDistanceShouldnotflyover: 1
PilotingSettingsMinAltitudeCurrent: 0.0
SpeedSettingsMaxVerticalSpeedCurrent: 0.2
SpeedSettingsMaxRotationSpeedCurrent: 10
SpeedSettingsOutdoorOutdoor: 0
```

Figura 27. **Parámetros de configuración archivo .yaml**

Los parámetros que se agregan al archivo yaml (Ver Figura 27) para la configuración inicial del dron son los siguientes:

- **Altitud Máxima:** La altura máxima que dron podrá elevarse.

  PilotingSettingsMaxAltitudeCurrent:15

- **Máxima Inclinación:** En grados, que el dron tomará para hacer desplazamientos en los ejes X e Y.

  PilotingSettingsMaxTiltCurrent: 5.0

- **Maxima Distancia:** que el dron se puede alejar de la estación terrena.

  PilotingSettingsMaxDistanceValue: 10.0

- **Altitud Mínima:** que se permite volar el dron

  PilotingSettingsMinAltitudeCurrent: 0.0

- **Maxima velocidad de Rotación:** que el dron puede hacer al girar, en grados por segundo:

  SpeedSettingsMaxRotationSpeedCurrent: 10

3.3 Obtención de Imágenes Monoculares

El proyecto se basa en el procesamiento de las imágenes recibidas por el cuadricóptero, para posteriormente enviar comandos de movimiento hacia el mismo dependiendo de la tarea a realizar. En este caso la evasión de objetos que se acercan.

3.3.1 Parámetros y suscriptor de Imagen

La secuencia de video de la cámara frontal del Bebop se publica en el *Topic image_raw*. Este es un mensaje de tipo *sensor_msgs/Image*. Al suscribirse a este *Topic* ya se puede recibir la secuencia de video. La calidad del flujo de video está limitada a 856x480 a 30 Hz, por tanto, se recibirán 30 fotogramas por segundo. Para iniciar el controlador el necesario abrir el terminal de Linux y ejecutar el siguiente comando:

```
$ source ~/bebop_ws/devel/setup.bash
```

Para dar inicio al controlador y recibir en tiempo real las imágenes de la cámara se ejecuta:

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

Como se observa en la Figura 28, al ejecutar el controlador se visualiza en una ventana las imágenes que transmite la cámara del Bebop 2.

Figura 28. **Imágenes transmitidas por el Bebop 2**

3.3.2 ROS y OpenCV

Los desarrolladores que desean implementar procesamiento de imágenes con OpenCV, se toparán con el problema de que ROS, transmite imágenes en su propio formato: *sensor_msgs/Image*. Puesto que OpenCV maneja a las imágenes como tipo: Matriz, CvBridge es una biblioteca de ROS que proporciona una interfaz entre ROS y OpenCV (Ver Figura 29).

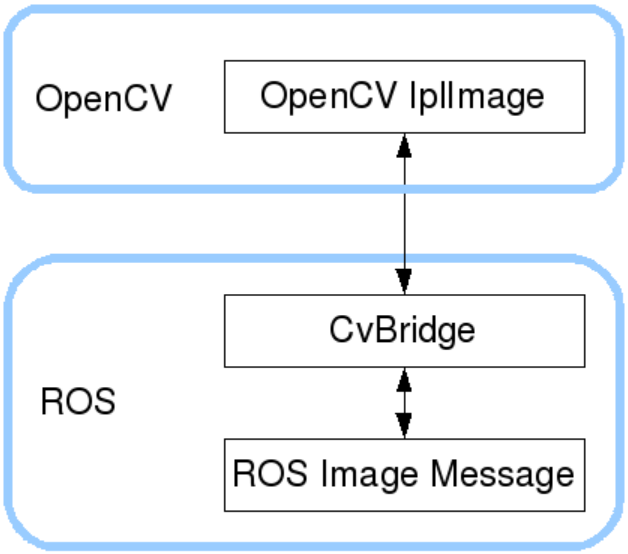

Figura 29. **Interfaz proporcionada por CVBridge**
Fuente: (ROS.org, 2017)

Puesto que la integración y programación del Proyecto será desarrollado en Python, *CvBrigde* proporciona una serie de funciones que serán ejecutadas en este lenguaje de programación.

Inicialmente el entorno de Python debe crear un nodo para establecer una suscripción del nodo a un tópico que publica las imágenes:

```
import roslib
import rospy

image_sub=rospy.Subscriber("/bebop/image_raw",Image,self.callback)
```

En este proceso Python crea un nodo que se suscribe al tópico *image_raw*. Del cual recibirá la trama de imágenes en formato *sensor_msgs/Image*. Para convertir ese formato en un compatible con OpenCV se ejecuta:

```
def callback(self,data):

        cv_image = Cvbridge.imgmsg_to_cv2(data, "bgr8")
```

Esta función *callback* se ejecuta cada vez que el nodo de ROS reciba una imagen del cuadricóptero (30 Hz).

Para convertir un mensaje de imagen ROS en un *cv::Mat* , el módulo CvBridge proporciona la función *imgmsg_to_cv2,* la cual devuelve la imagen codificada según el usuario desee. En este caso la imagen está codificada a *bgr8:* imagen en color con orden de color azul-verde-rojo.

3.4 Control de movimiento del UAV

El driver *bebop_autonomy* publica tópicos que permiten funciones principales del cuadricóptero para el despegue, navegación, aterrizaje. En la tabla Tabla 3 *Tópicos principales para el control de movimiento del UAV* se describe los tópicos necesarios para el control del UAV

Tabla 3
Tópicos principales para el control de movimiento del UAV

TOPIC	TIPO MENSAJE	DESCRIPCIÓN
Takeoff	std_msgs/Empty	Inicia el Despegue
land	std_msgs/Empty	Aterrizaje seguro
reset	std_msgs/Empty	Paro de Emergencia, apaga inmediatamente los motores
cmd_vel	geometry_msgs/Twist	Control de movimientos

Las órdenes publicadas en el tópico cmd_vel deben estar en el rango de [−1 … 1] y son puestos en cero automáticamente al recibir algún comando en tópicos de mayor trascendencia como lo son takeoff, land o reset por razones de seguridad.

En el siguiente proceso se observa la forma correcta de realizar el despegue y aterrizaje del UAV. Desde el programa de Python se crea el objeto que va a ser publicado a los *Topic* correspondientes de *bebop_autonomy*.

```
pub1 = rospy.Publisher('/bebop/takeoff', Empty, queue_size = 1)
pub2 = rospy.Publisher('/bebop/land', Empty, queue_size = 1)

#                       Despegue del UAV
print('despegar')
pub1.publish()

#                       Aterizaje Seguro
print('aterrizando')
pub2.publish()
```

Para enviar comando de movimiento al UAV, una vez que haya despegado, el mensaje a publicar debe ser de tipo *geometry_msgs/ Twist*, El rango de las variables que componen el mensaje Twist está comprendido entre [-1 1], de manera que 0 es la mínima velocidad y 1 en dirección

positiva y -1 la máxima velocidad en dirección negativa. Las velocidades del Bebop 2 en sentido positivo se muestran en la Figura 30. Esquema de desplazamiento positivos del UAV:

***Figura 30*. Esquema de desplazamiento positivos del UAV**

En el programa de Python se crea el objeto *pub*, el cual será publicado en el tópico */bebop/cmd_vel*. Así mismo el objeto twist contiene todos los parámetros para realizar los movimientos del UAV. El proceso para enviar los datos de movimiento se muestra a continuación:

```
pub = rospy.Publisher('/bebop/cmd_vel', Twist, queue_size = 1)
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)
```

CAPITULO IV

4 IMPLEMENTACIÓN DE ALGORITMOS PARA LA DETECCIÓN Y SEGUIMIENTO VISUAL DEL OBJETIVO (OBJETO)

El presente capítulo detalla el desarrollo de las técnicas y algoritmos que se aplicaron para la detección a tiempo real de movimiento en la escena, que es transmitida desde el rango de visión de la cámara hacia el computador portátil.

Al detectar movimiento en la escena, es posible discriminar el objeto que está en movimiento. Cabe mencionar que no es necesario una plantilla definida para localizar el objeto, la plantilla se definirá automáticamente, una vez que se haya producido el movimiento del objeto. Es decir, el objeto no es definido, puede ser cualquiera que cumpla con las características mínimas para su detección.

4.1 Detección de objeto Movimiento en la Escena entre Imágenes Consecutivas

Considerando a una imagen en escala de grises como matriz de *MxN*, si se tiene dos imágenes exactamente iguales, es decir, dos matrices con los valores de sus elementos iguales; Al realizar una diferencia entre las dos matrices, el resultado de cada elemento tendría un valor de cero. En una captura de video (Donde existe movimiento) se obtienen fotogramas secuenciales que son capturados cada cierto tiempo. Al realizar una diferencia entre 2 fotogramas seguidos, tomando en cuenta que son matrices, se podrá notar que no todos los elementos tienen un valor de cero. Entonces, es posible deducir que existe movimiento en la Escena.

En el proyecto para detectar movimiento en videos (Portillo-Portillo, Sánchez-Pérez, Olivares-Mercado, & Pérez-Meana, 2014), el método de diferencia de fotogramas ha sido evaluado, teniendo resultados considerables, para la implementación de este algoritmo.

4.1.1 Diferencia de Fotogramas y estimación de Bordes

Si $I_{(x,y)}(t)$ es la imagen actual que se recibe del UAV, y $I_{(x,y)}(t-1)$ es la imagen anterior. La diferencia absoluta entre la imagen actual y la anterior se define por:

$$\boldsymbol{D_{(x,y)}(t) = |I_t - I_{t-1}|} \quad (27)$$

$\boldsymbol{D_{(x,y)}(t)}$ es una matriz cuyos valores estan entre 0 y 255, esta matriz indica que pixeles han tenido movimiento. Es necesario realizar una imagen en binario, de esta manera se discrimina de movimientos notables y se elimina el ruido que puede existir. La ecuación que discrimina es la siguiente:

$$\boldsymbol{M_{(x,y)}(t)} \begin{cases} \boldsymbol{1}, & \boldsymbol{D_{(x,y)}(t) \geq U(t)} \\ \boldsymbol{0}, & \boldsymbol{D_{(x,y)}(t) < U(t} \end{cases} \quad (28)$$

Donde:

$$\boldsymbol{U(t) = \bar{D}(t) + \sigma\,(t)} \quad (29)$$

$\boldsymbol{U(t)}$ es el valor mínimo que cada pixel debe cumplir para establecer como un valor de interés. $\boldsymbol{\bar{D}(t)}$ es el promedio de diferencias de todos los pixeles de la matriz $\boldsymbol{D_{(x,y)}(t)}$ y $\boldsymbol{\sigma\,(t)}$ es la

desviación estándar de aquellos valores. Es necesario notar que estos valores se calculan a tiempo real.

Una vez obtenida una imagen de valores binarios, hay que tomar en cuenta que los pixeles que comprenden el objeto de interés tienen colores similares, por lo que estos pixeles pueden ser descartados y dividir al objeto en partes. Con a las herramientas que ofrece OpenCV, es posible extraer los bordes del objeto con el algoritmo de detección de bordes de Canny (Canny, 1986). En la Figura 31, la imagen de la izquierda es el resultado binario, se puede apreciar que existe ruido detrás del objeto (Libro) en movimiento.

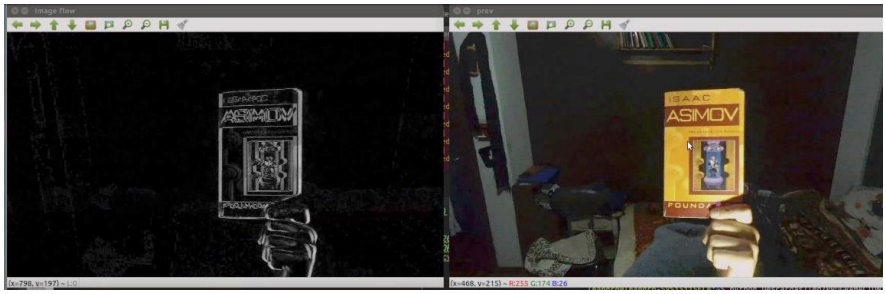

***Figura 31*. Detección de Movimiento a partir de Diferencia de Imágenes**

Para seleccionar los bordes en la región de interés se realiza una operación morfológica de dilatación sobre los pixeles considerados movimiento con un elemento estructural en forma de círculo denominado Kernel correspondiente al 12% aproximadamente de la imagen original. Previo a la dilatación se debe hacer una eliminación de ruido con la herramienta de erosión que es proporcionada por OpenCV. El resultado del algoritmo se puede apreciar en la Figura 32.

***Figura 32*. Filtrado de Imagen Binaria para la extracción de Bordes**

4.1.2 Filtrado de Contornos

Una vez que los bordes son extraídos, en el programa se obtiene un arreglo *array* de los parámetros de todos los bordes encontrados, es decir, su posición inicial y el valor de sus dimensiones. Por lo tanto, se puede reconstruir una forma geométrica, como se aprecia en Figura ***33*** todos los bordes de los objetos encerrados en un cuadrilátero.

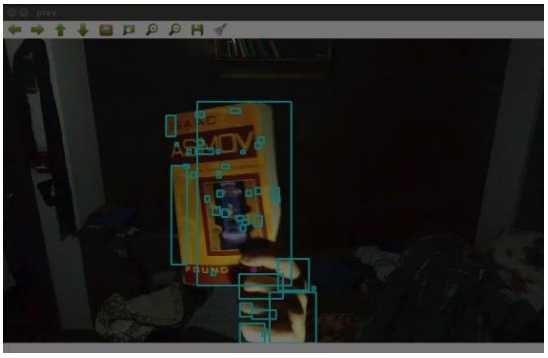

***Figura 33*. Bordes detectados del objeto sobre la imagen original de la cámara**

El Filtrado de los contornos se lo realiza discriminado las áreas de los cuadriláteros que sean muy grandes. Las áreas muy grandes se deben a que el UAV, al estar elevado presenta pequeñas variaciones de movimiento, por lo tanto, estas áreas no deben ser consideradas al momento de encapsular al objeto en una sola forma geométrica. Con una forma similar a la ecuación (33).

4.1.3 Captura de Objeto

El objeto de interés se compone de varios bordes, los cuales están representados por cuadriláteros de diferentes tamaños, es necesario integrar todos estos bordes para que el objeto sea representado por una sola forma geométrica. De esta forma obtener un cuadro del objeto que previamente fue detectado en movimiento.

En la Figura 34 se observa al objeto encerrado en un solo cuadro que será guardado como una nueva imagen, para su posterior procesamiento. Es decir, la imagen del objeto guardado se convierte en una plantilla para realizar un seguimiento visual del mismo.

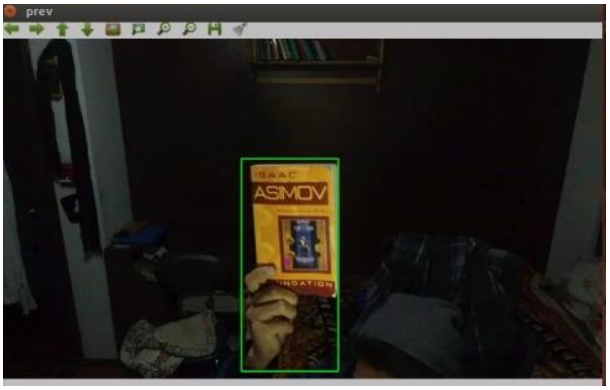

***Figura 34*. Encapsulamiento del objeto en movimiento detectado**

El siguiente diagrama de flujo (), indica el proceso del algoritmo realizado en Python, que describe la captura de las imágenes proporcionadas por el UAV, la detección de movimiento por diferencia de imágenes, las estimación y filtrado de bordes, y finalmente la captura del objeto previamente detectado, salvado su posición en la imagen y sus dimensiones (x,y,h,w).

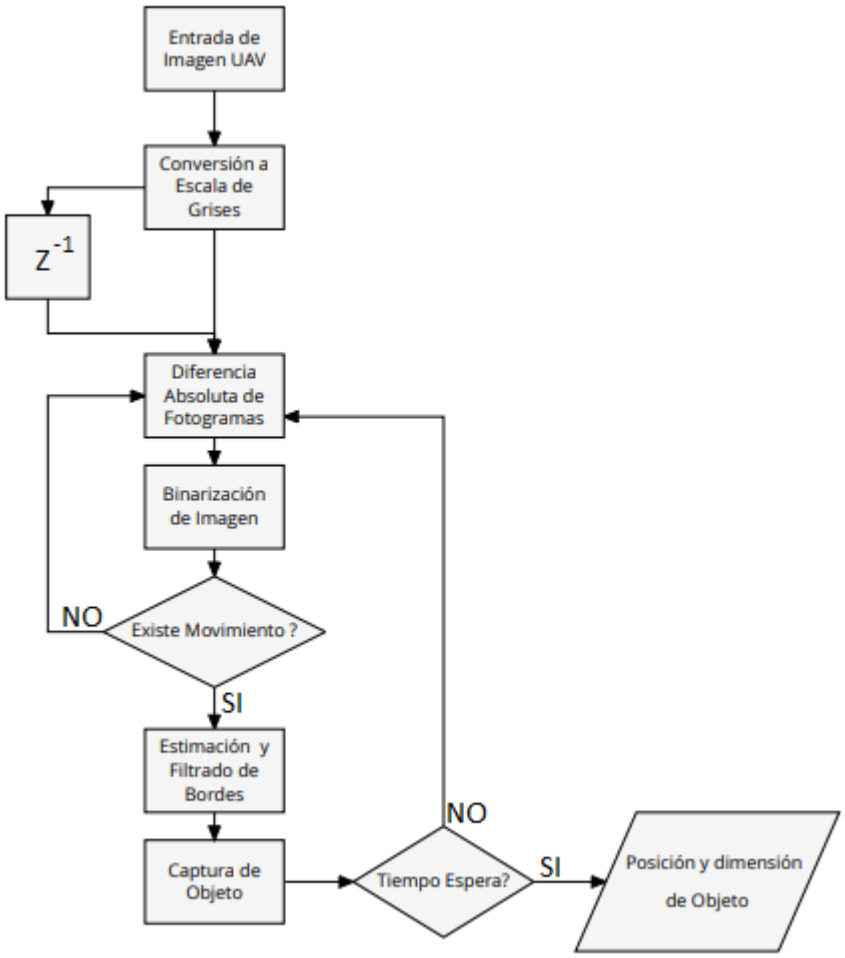

***Figura 35*. Diagrama de Flujo propuesto para la detección y captura de objeto en movimiento**

4.2 Seguimiento del objeto a partir de Estimación de Flujo Óptico

Una vez capturada la plantilla del objeto con la propuesta de detección de movimiento y captura del mismo. Los datos que actualmente están disponibles son, la posición del objeto en la imagen recibida por el UAV y las dimensiones a partir de la posición, es decir, los valores de X,Y para posición y H,W para dimensiones (Figura 36).

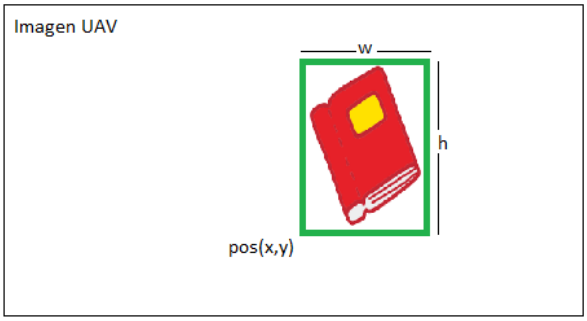

***Figura 36*. ilustración de la información del objeto capturado**

Para realizar el seguimiento del objeto de interés, es necesario extraer las características del objeto, en otras palabras, los puntos de interés de este. Para realizar el seguimiento se propuso algoritmo de flujo óptico Lucas-kanade para seguimiento que será descrito en este capítulo.

4.2.1 Extracción de Características locales del Objeto

Como se indicó en la Detección de Características ideales para el Seguimiento de Objetos, para realizar el seguimiento de un objeto de interés dentro de una escena, es importante tener en cuenta la velocidad que el objeto podría tomar. Por esta razón, se utiliza el algoritmo de Shi-Tomasi para la detección de Esquinas, a este algoritmo se lo denomina "*Good Features to Track*", puesto que hace el seguimiento de los puntos de interés a velocidades considerables.

OpenCV dispone de una función, ***cv2.goodFeaturesToTrack***. Esta nncuentra N esquinas más fuertes en la imagen por el método de Shi-Tomas. Como requisito previo la imagen debe estar en escala de grises. La función permite configurar parámetros como:

- El número de esquinas a encontrar.
- El nivel de calidad de los puntos de interés encontrados, su valor comprende entre 0 y 1.
- Se puede proporcionar la distancia euclidiana mínima entre esquinas detectadas.

- El tamaño del bloque que rodea al pixel

En la siguiente porción de código se puede ver la configuración de los parámetros y la ejecución de la función ***cv2.goodFeaturesToTrack:***

```
# parametros para deteccion de esquinas ShiTomasi
self.feature_params = dict( maxCorners = 500,
                        qualityLevel = 0.01,
                        minDistance = 10,
                        blockSize = 7 )

#Detección de Puntos de Interes de la imagen (Objeto) capturada
p0_aux = cv2.goodFeaturesToTrack(newimage, mask = None,
**self.feature_params)
```

Los valores de los parámetros han sido propuestos y evaluados por . La función ***cv2.goodFeaturesToTrack***, retorna la posición (x,y) de cada punto de interés detectado en la imagen capturada del objeto.

En la Figura 37 se muestra dos ejemplos de objetos capturados con sus respectivos puntos de interés, como la captura de imagen parte desde la posición (0,0), se añadió el desplazamiento de aquellos puntos con la información proporcionada por la captura del objeto (Ver Figura 35)

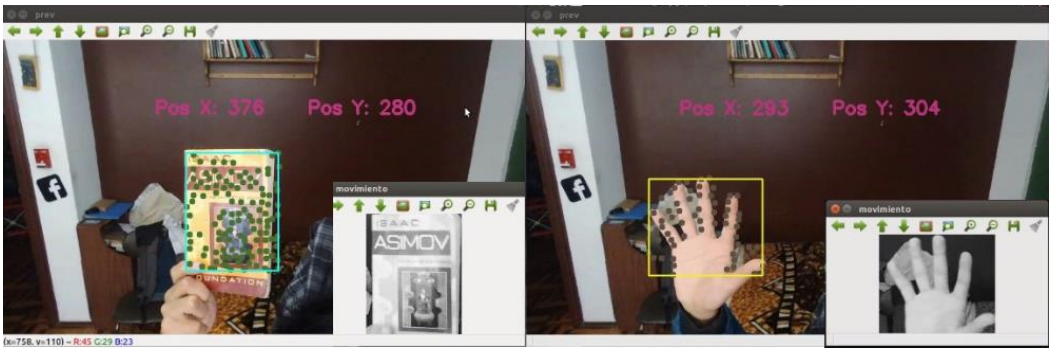

***Figura 37*. Puntos de interés de objetos reales capturados**

4.2.2 Estimación de Velocidad de Puntos de Interés

Volviendo a la definición de Flujo óptico descrito en Detección y Seguimiento de Objetos, es posible estimar el campo vectorial que describe el movimiento de cada una de las características (puntos de interés) solamente del objeto. Partiendo de los criterios supuestos del flujo óptico, de que todos los píxeles vecinos tendrán un movimiento similar. El método de Lucas-Kanade es el método más robusto para este tipo de aplicaciones, junto con los "*Good Features to Track*" que ofrece el algoritmo de Shi-Tomasi.

OpenCV ofrece la función ***cv2.calcOpticalFlowPyrLK***, esta función cual realiza un seguimiento de los puntos de interés previamente detectados por ***cv2.goodFeaturesToTrack***. El rastreo de los puntos comienza desde el segundo fotograma, porque en el primer fotograma hace el reconocimiento de los puntos de interés.

Los datos que ingresan a la función ***cv2.calcOpticalFlowPyrLK*** son: El fotograma actual, el fotograma anterior, el vector de los puntos de interés del fotograma anterior y los criterios de configuración. La función retorna el vector de los puntos de interés del fotograma actual junto número de estado que indica si el punto pertenece a un punto del fotograma anterior. Por último, se realiza una actualización del fotograma y los puntos para la siguiente iteración.

```
# Parametros para lucas kanade
lk_params = dict( winSize  = (15,15),
                  maxLevel = 2,
criteria =(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT,10,0.03))

# Primer Frame
if t==0:
      p0=cv2.goodFeaturesToTrack(cv_image_prev,, mask=None,
                                 **feature_params)
      t++
if t>0:
      frame_gray = cv2.cvtColor(cv_image_next, cv2.COLOR_BGR2GRAY)
      old_gray = cv2.cvtColor(cv_image_prev, cv2.COLOR_BGR2GRAY)
# Calculo de Flujo optico
      p1, st, err = cv2.calcOpticalFlowPyrLK(old_gray, frame_gray,
.                                            p0, None, **self.lk_params)
                       term_crit = (cv2.TERM_CRITERIA_EPS, 30, 0.01))

#Actualizar Frame y puntos
cv_image_prev = cv_image_next
p0=p1
```

En la Figura 38 se aprecia el desplazamiento de un objeto hacia la derecha, el seguimiento de los puntos de interés está representado por líneas de colores. Los datos que arroja esta estimación son de gran interés para realizar el cálculo de la velocidad de cada punto de interés que pertenece al objeto.

***Figura 38*. Seguimiento de los puntos de interés de un objeto desplazado**

La velocidad es la relación que se establece entre el espacio o la distancia que un objeto recorre en un determinado tiempo. Partiendo de esta definición y en relación a este proyecto, es posible

calcular la velocidad de cada punto de interés (pixel), en función de la distancia de pixeles recorridos por cada imagen que transmite el UAV. Sabiendo que la tasa de transmisión es de 30 Hz.

Tabla 4

Planteamiento para calcular la velocidad de los puntos de interés

Parámetro	Fórmula	Descripción	Unidad
Distancia	$p1 - p0$	Es la distancia en pixeles recorrida entre el fotograma actual y el anterior.	$[pixeles]$
Tiempo	1 Frame=0.03[s]	La distancia se puede medir entre cada frame que transmite el UAV, esto es una tasa de 30 Hz	$[segundo]$
Velocidad	$\frac{p1 - p0}{0.03}$	Describe cuantos pixeles a recorrido el punto de interés en un segundo	$\left[\frac{pixeles}{segundo}\right]$

En la Tabla 4 se plantea una forma de cálculo para la velocidad de todos los puntos característicos del objeto en seguimiento.

4.2.3 Agrupación y Encapsulamiento de Puntos de Interés

El problema que se tiene al aplicar esta técnica de flujo óptico es que los puntos de interés, a medida que el objetivo está en movimiento, se confunden con el fondo u otros objetos que se encuentran detrás del objeto de interés. La consecuencia de este problema es que los puntos de interés que pertenecían al objeto se quedan fuera de él. En este proyecto de investigación se plantea una técnica basada en herramientas estadísticas y en un algoritmo de agrupamiento suministrado por OpenCV para reducir este problema.

Un objeto al tener movimiento en la escena, sus puntos característicos se desplazan a velocidad similares. Al determinar un promedio de las velocidades de todos los puntos de interés que comprende el objeto, es posible descartar aquellos puntos de interés que no están en movimiento, dichos puntos sin movimiento son aquellos que no pertenecen al objeto en movimiento.

$$\overline{V_p} = \frac{1}{N}\sum_{i=0}^{N} V_p(i) \quad (30)$$

En la ecuación ((33), $\boldsymbol{V_p(t)}$ es el vector que contiene la velocidad de cada pixel calculado, N es la dimensión de aquel vector. $\overline{V_p}$ es la velocidad promedio de todos los pixeles encontrados. Los umbrales de decisión para estos movimientos se obtienen al calcular la desviación estándar, la fórmula para descartar pixeles no válidos es:

$$pv(t)\begin{cases} V_p(i), & V_p(i) \geq \overline{V_p} - \sigma_p \\ V_p(i), & V_p(i) \leq \overline{V_p} + \sigma_p \\ 0, & fuera\ de\ umbrales \end{cases} \quad (31)$$

Posterior a este método para descartar pixeles que no tienen movimiento, se aplica un último filtro que agrupa los pixeles válidos para lograr encapsularlos. Este criterio se basa en la pertenencia de los puntos de interés de objeto, asegurando que los puntos de interés que pertenecen a un mismo objeto están cercanos.

El algoritmo k-means permite agrupar puntos utilizando como criterio una medida de distancia apropiada. OpenCV maneja la función ***cv2.kmeans***, basada en distancia Euclidiana para conjuntos de puntos que tienen covarianza correlacionada y varianza diferente en sus ejes, esto significa que no deben crecer de la misma manera para cada uno de sus ejes.

Este algoritmo trabaja sobre un conjunto de pixeles aplicando la función y los criterios establecidos en los parámetros de ***cv2.kmeans:***

- **Muestras**: debe ser del tipo de datos *np.float32* , y cada característica debe colocarse en una sola columna.

- **nclusters(K)**: Número de agrupaciones que ser requiere devolver
- **Criterios**: Son los criterios de terminación de iteración. Cuando se cumple este criterio, la iteración del algoritmo se detiene
    - cv2.TERM_CRITERIA_EPS - detiene la iteración del algoritmo si se alcanza la precisión especificada.
    - cv2.TERM_CRITERIA_MAX_ITER - detiene el algoritmo después del número especificado de iteraciones.
    - cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER - detiene la iteración cuando se cumple alguna de las condiciones anteriores.
- **Intentos**: especifica el número de veces que se ejecuta el algoritmo utilizando diferentes etiquetas iniciales. El algoritmo devuelve las etiquetas que producen la mejor compacidad.

El siguiente diagrama de flujo resume todos los métodos y técnicas que intervienen en la estimación de flujo óptico del objeto de interés. Dando como resultado como resultado un objeto encapsulado, adjunto a su velocidad de desplazamiento, su posición y su área.

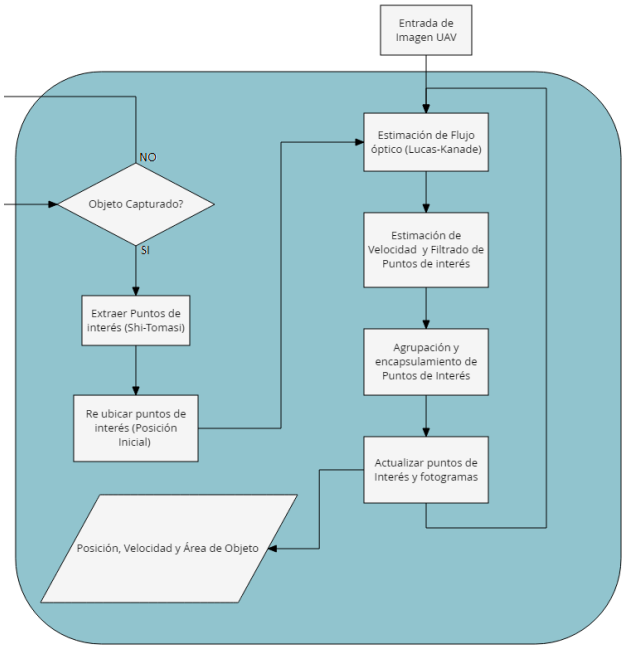

***Figura 39*. Proceso estimación de velocidad para el objeto capturado**

4.3 Regresión Lineal para la estimación de Eventos relacionados al UAV

Una vez que se tiene la información de posición, velocidad y Área del objeto, en relación con el lente de la cámara del UAV cada vez que el computador recibe la imagen. Es posible estimar la tasa de crecimiento del Área y así determinar si el objeto se acerca o se aleja del lente de la cámara. El objeto al acercarse al UAV está comprometiendo estabilidad de vuelo, o su navegación, es importante detectar estos eventos para tomar una acción de control hacía UAV y evitar o minimizar el daño que puede ser causado.

La regresión lineal es una aproximación para modelar la relación entre una variable escalar dependiente, en este caso el área, y una variable independiente que sería el tiempo. En el campo de la robótica la regresión lineal es un algoritmo de aprendizaje supervisado que se utiliza en Machine Learning y Estadística. La ventaja es que al determinar la dependencia de los datos se puede predecir o estimar un valor a partir de una variable que no se encuentra en la distribución.

Python ofrece una librería de aprendizaje automático denominada ***scikit-learn*** que consta de algunos algoritmos de clasificación, agrupamiento y por supuesto regresión. que incluyen máquinas de vectores de soporte, bosques aleatorios, aumento de gradiente, y está diseñado para interactuar con las bibliotecas numéricas y científicas de Python NumPy y SciPy.

Para implementar el algoritmo de regresión lineal es necesario pasar por una etapa de entrenamiento, que será de 15 fotogramas una vez encapsulado el objeto. Posterior a esto el entrenamiento será a tiempo real porque la distribución de datos es una pila que se va alimentando constantemente.

```
#Capturando los 15 primeros datos
if len(self.regresion_Tiempo)<=15:
      self.regresion_Tiempo.append(round(time()-
                              self.tiempo_inicial,2))
      self.regresion_Area.append(self.AreaInstantanea)
#Entrenamiento a tiempo real
else:
      self.regresion_Tiempo.pop(0)
      self.regresion_Tiempo.append(round(time()-
                                  self.tiempo_inicial,2))
      self.regresion_Area.pop(0)
      self.regresion_Area.append(self.AreaInstantanea)
      # Función Entrenar modelo
      self.modelo.fit(t,a)
      #predecir el último valor
      aux=t[len(t)-1]
      aux=aux.reshape(-1,1)
      #Función pronosticar valor
      self.Area_pred = abs(int(self.modelo.predict(aux)))
```

El siguiente código muestra el algoritmo desarrollado para el entrenamiento constante del área del objeto a través del tiempo. Para no superar la tasa de transmisión de las imágenes del UAV se ha seleccionado los últimos 15 datos.

4.3.1 Acercamiento y Alejamiento de Objeto

Partiendo de la fórmula de la recta (33), donde *Y* es el resultado, *X* es la variable, *b* la constante y $\boldsymbol{m}$ la pendiente de la recta.

$$\boldsymbol{Y = mX + b} \qquad \textbf{(32)}$$

Si la regresión es una recta, se puede deducir que los son crecientes o de crecientes según la pendiente de dicha recta. En la Figura 40. Ejemplo de Regresión Lineal se aprecia que los datos son crecientes, por lo tanto, la pendiente tendría un valor positivo, al contrario, si la pendiente tuviera un valor menor a cero los datos estarían decreciendo.

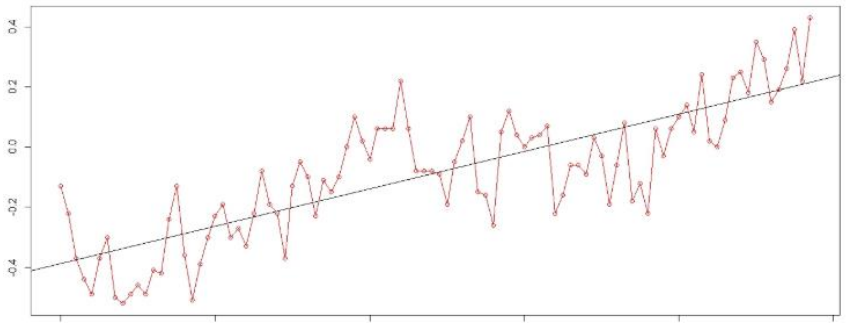

Figura 40. **Ejemplo de Regresión Lineal**

El parámetro que se tomará en cuenta para la estimación de eventos será la pendiente de la recta que el modelo entrenado devuelve al llamar la función **modelo.coef[0].** Entonces para deducir si el objeto se acerca al lente de la cámara el valor de la pendiente será positivo, esta estimación se puede apreciar en la Figura 41, de igual manera se puede estimar el alejamiento del objeto.

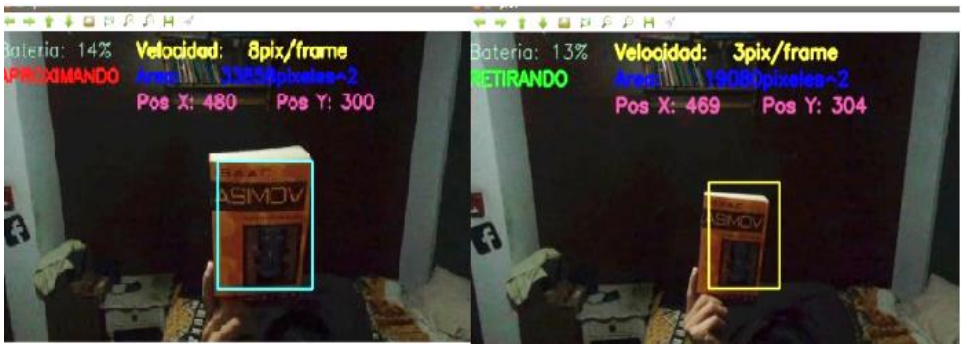

***Figura 41*. Estimación de Acercamiento y Alejamiento de Objeto**

4.4 Esquema General para la estimación de Eventos

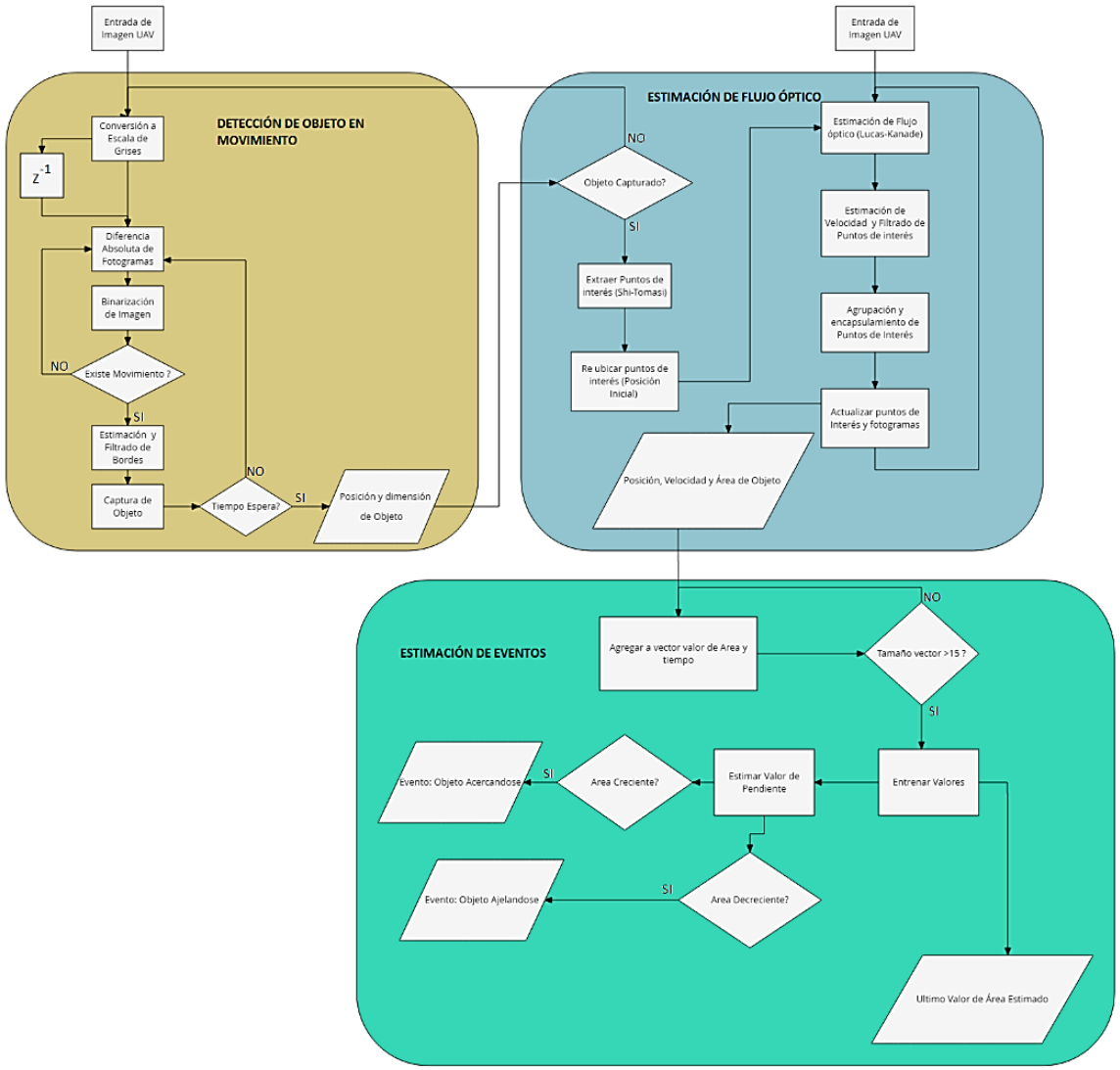

***Figura 42*. Esquema General de Detección, seguimiento de Objeto y estimación de Eventos**

CAPITULO V

5 *EVASIÓN DEL VEHÍCULO AÉREO NO TRIPULADO*

Para diseñar el controlador de un sistema es necesario conocer su modelo matemático. Este capítulo describe el método aplicado para determinar el modelo matemático del UAV a controlar, por ejemplo en las siguientes investigaciones se aplicó la teoría de control para seguimiento de trayectorias (Wilbert G. Aguilar, Angulo, & Costa-Castello, 2017; Wilbert G. Aguilar, Salcedo, Sandoval, & Cobeña, 2017) y evasión de obstáculos (W. Aguilar et al., 2017; Wilbert G. Aguilar, Casaliglla, Pólit, Abad, & Ruiz, 2017), a partir de los datos obtenidos detallados en el capítulo anterior y el modelo matemático se diseña un controlador de lógica difusa. A lo largo del capítulo se explicará las razones por las que se optó e implementar dicho controlador.

5.1 Modelo matemático de Movimiento del UAV

Es posible determinar el modelo matemático del UAV a partir de los datos inerciales que el mismo cuadricóptero proporciona a través del SDK, el único problema es la tasa de muestreo qué es de 5 Hz. Esto es un problema debido a que la respuesta de movimientos del UAV debe ser rápida.

En un proyecto de investigación anterior se propone un método a partir de los movimientos que se puede estimar con la cámara del UAV, se logró obtener el modelo matemático del Parrot Bebop 1 (Salcedo Peña, 2018). En el presente proyecto se aplicará el mismo procedimiento para el Parrot Bebop 2.

5.1.1 Estimación de movimiento entre imágenes

Para estimar el movimiento del UAV se utiliza un concepto descrito en este proyecto, la estimación de flujo óptico sobre las imágenes que son transmitidas. En este caso el objetivo (target) será conocido, pero seleccionado por el usuario con la ayuda de una herramienta ROI. El procedimiento para determinar el modelo matemático en Pitch es el siguiente:

- Ubicar el objetivo (target) a una distancia prudente de modo que el movimiento del dron no pierda de vista el objetivo.
- Comenzar la grabación del video al enviar los comandos de movimiento al dron.
- Los comandos de movimiento del UAV deben ser escalones temporales positivos y negativos con pausas en inercia.
- Luego de terminar con la grabación, es necesario hacer el procesamiento con estimación de flujo óptico.
- Salvar los datos de área, valor de escalón y tiempo en un archivo plano.
- Con la herramienta *ident* de Matlab obtener el modelo matemático.

Figura 43. **Procesamiento del Target para estimación de movimiento de UAV**

En la Figura 43 se muestra el seguimiento del target para la estimación de movimiento del UAV, la característica que se toma en cuenta en el procesamiento es el área del target. La imagen de la derecha muestra un aumento del área debido al movimiento positivo de pitch.

5.1.2 Obtención del modelo matemático

Una vez realizado el procesamiento del video, los datos de área, tiempo y escalón son salvados en un archivo plano. Estos datos deben ser preparados para la identificación del modelo matemático, como se observa en la Figura 44 los datos de interés son: Los valores de los pulsos que se enviaron al UAV en función del tiempo y la diferencia de área entre cada fotograma.

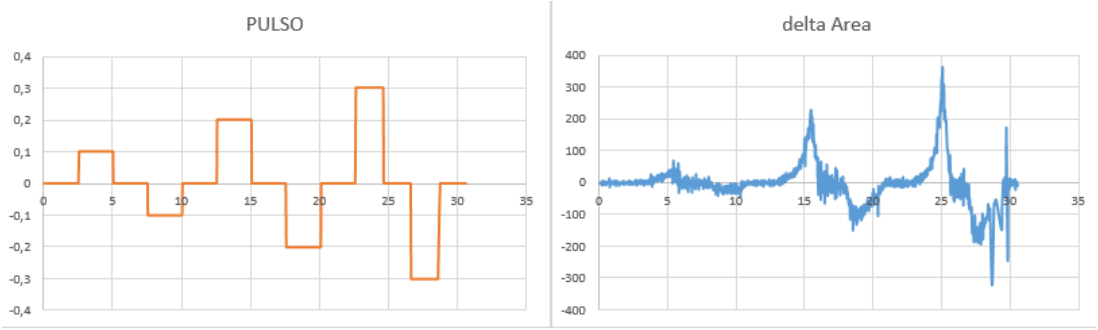

Figura 44. **Datos para la identificación del modelo matemático del eje de rotación Pitch**

La función de transferencia representada va en función del tiempo de muestreo que es posible calcular por la tasa de transmisión del UAV (30 Hz). La función de Transferencia tiene la forma:

$$\mathbf{G(s)} = \frac{\boldsymbol{K_p}}{\mathbf{1} + \boldsymbol{T_p}.\boldsymbol{S}} \tag{33}$$

Gracias a la herramienta de identificación de sistemas *Ident* proporcionada por MATLAB, los valores obtenidos son: $K_p = 1.502$ y $\boldsymbol{T_p} = 0.01249$.

5.2 Implementación del Controlador para la Evasión de Objetivo

Cuando el objetivo es detectado y determinado sus características, se mantiene a cierta distancia del UAV, en otras palabras, el área calculada por el algoritmo se mantiene relativamente en el mismo valor. El propósito de agregar el controlador es que realice una acción de evasión para mantener cierta distancia con el objetivo, siempre y cuando la detección de eventos lo permita.

Se seleccionó el controlador de lógica difusa porque en la evasión del objetivo intervienen dos variables físicas que son, la distancia (para el procesamiento: área) del objetivo hacia el UAV y, la velocidad (tasa de cambio del área) en que se acerca al UAV.

5.2.1 Análisis y Diseño de Controlador Difuso

Para que el controlador difuso funcione de manera correcta la entrada de error será la diferencia del área actual y la primera área calculada al detectar el objetivo. La variación del error se irá determinando en función del tiempo. La salida del control difuso es la velocidad que el UAV debe tomar para realizar la acción de evasión, la dirección del movimiento es negativa en relación con el eje lineal X (Figura 45).

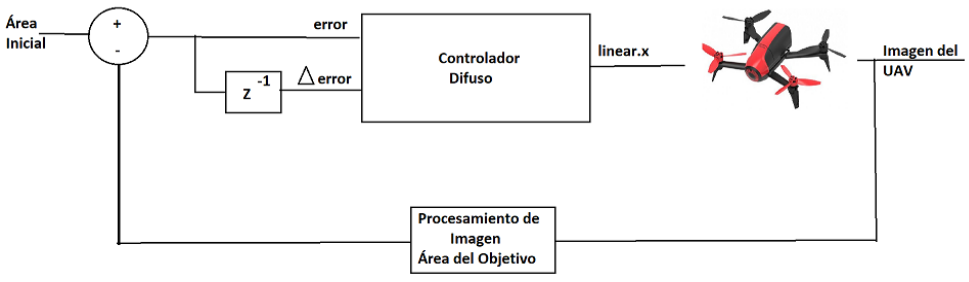

Figura 45. **Esquema de control difuso para evasión del objetivo**

5.2.1.1 Funciones de Membresía

La función de membresía del error se determinó con el criterio mantener la distancia del objetivo, es decir, el error sea cero. Los valores lingüísticos son:

Tabla 5

Descripción de la función de Pertenencia del error

Valor Lingüístico	Valor en Universo de Discurso	Descripción
Z: Cero	[0, 0 ,3]	El error de distancia es nada
PC: Poco Cerca	[2, 3, 5]	El objeto se acercó un poco
C: Cerca	[3, 5, 7.5]	El objeto está acercándose
MC: Muy Cerca	[5, 7.5 10]	El objeto está notablemente cerca
SC: Super Cerca	[7.5, 10, 10]	El objeto se acercó demasiado

El gráfico de la función de pertenencia de la Figura 46, muestra una tolerancia de cercanía del objetivo.

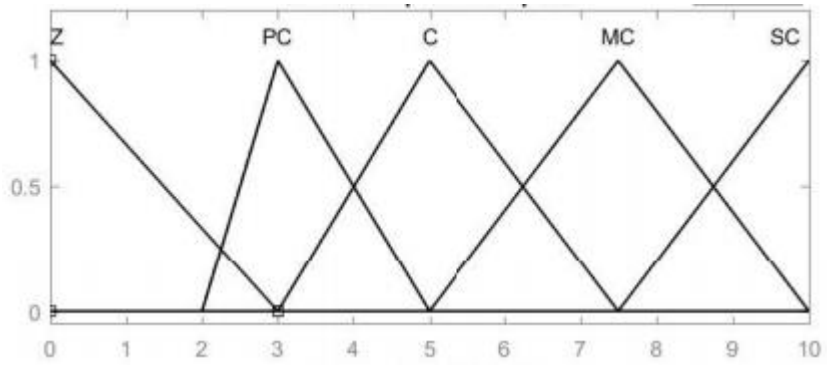

*Figura 46***. Función de membresía del error**

La función de membresía del cambio de error se determinó con el criterio evadir el objetivo dependiendo de la velocidad que se acerca al UAV. Los valores lingüísticos son:

Tabla 6

Descripción de la función de Pertenencia del error

Valor Lingüístico	Valor en Universo de Discurso	Descripción
Z: Cero	[0, 0 ,2]	El objetivo no está en movimiento
PR: Poco Rápido	[1.5, 2, 3]	El objetivo se acerca lentamente
R: Rápido	[2, 3, 4]	El objetivo se está acercando moderadamente
MR: Muy Rápido	[3, 4, 5]	El objetivo se acerca rápidamente
SR: Super Rápido	[4, 6, 6]	El objetivo se está acercando demasiado rápido

El gráfico de la función de pertenencia de la Figura 47, describe que la reacción debe ser más fuerte mientras más rápido se acerca el objetivo.

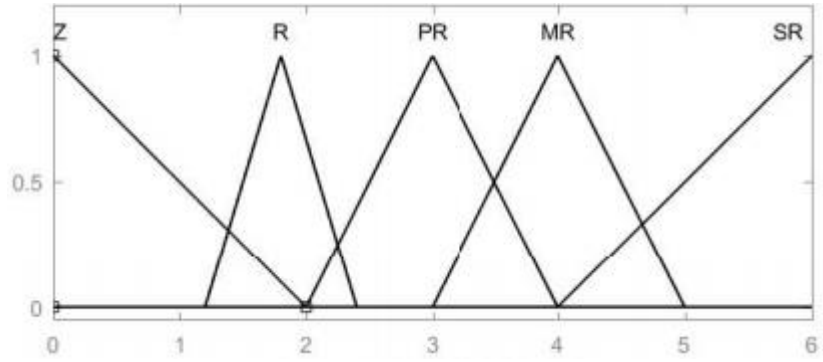

***Figura 47*. Función de membresía del cambio de error**

5.2.1.2 Caracterización Difusa de la Salida

La velocidad de reacción que debe tomar el UAV depende directamente de la velocidad en que el objeto se acerca al mismo. Si el objeto se acerca rápidamente el UAV debe tomar lo más pronto posible una velocidad fuerte de reacción, pero si el objeto está muy cerca al UAV, y su velocidad es muy baja el UAV puede tomar una velocidad prudente para mantener distancia.

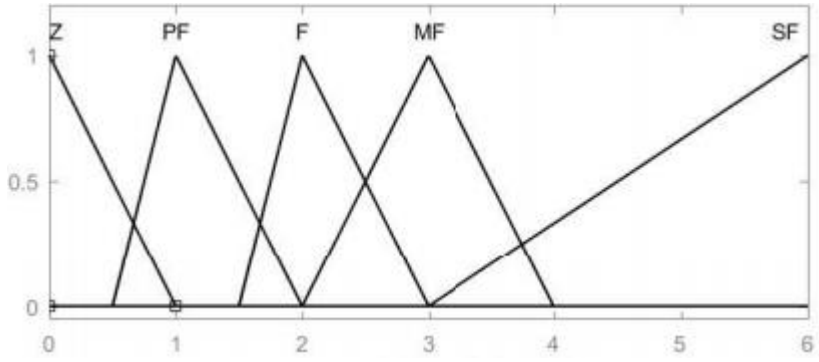

***Figura 48*. Función de pertenencia para la velocidad del UAV**

Como se observa en la Figura 48 se espera tomar rápidamente una acción fuerte mientras más rápido se acerca el objetivo, este criterio también será aplicado en las reglas difusas.

5.2.1.3 Reglas Difusas

Para asociar los conjuntos difusos de error y cambio de error es necesario plantear las reglas difusas, para que el controlador responda de manera correcta frente a las variables que ingresan al mismo. Cada conjunto difuso se compone de 5 valores lingüísticos por lo que se obtiene un total de 25 reglas que se plantean en la Tabla 7

Tabla 7
Reglas Difusas para control de velocidad de UAV

		ERROR				
	VELOCIDAD	Z	PC	C	MC	SC
CAMBIO DE ERROR	Z	Z	Z	F	MF	MF
	PR	Z	F	F	MF	MF
	R	Z	F	MF	SF	SF
	MR	Z	MF	MF	SF	SF
	SR	Z	MF	SF	SF	SF

La velocidad que toma el UAV es muy fuerte cuando el objeto se acerca a mayor velocidad, pero si se acerca a velocidad moderada el UAV puede hacer una evasión suave. Esto se puede comprobar al realizar un análisis de la Superficie de Control de las reglas de control difuso mostradas en la Figura 49

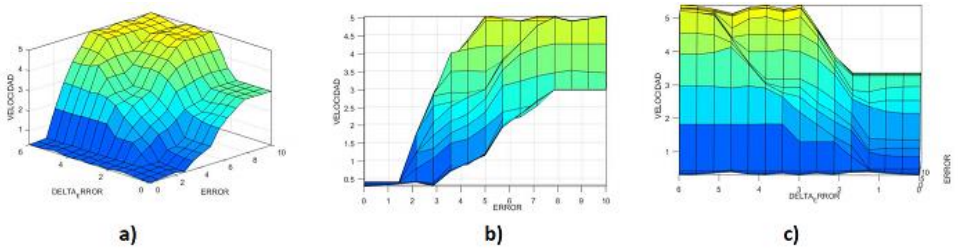

***Figura 49*. Superficie de control difuso de Velocidad para UAV a) Vista 3D b) Vista Error-Velocidad c) Vista Delta Error – Velocidad**

En Python se implementó el controlador difuso con la ayuda de la librería ***skfuzzy***, esta librería proporciona muchas herramientas de utilidad para problemas que involucran lógica difusa.

CAPITULO VI

6 PRUEBAS Y RESULTADOS

Este capítulo tiene el objeto de comprobar el comportamiento del UAV frente a los diferentes escenarios a los que se puede enfrentar. Los métodos y algoritmos planteados en los capítulos anteriores serán evaluados en base a diferentes criterios como los descritos en la

Tabla 8
Criterios de evaluación para el desempeño del UAV

Criterio	Descripción
Distancia de detección	La distancia que hay entre el objeto y el lente de la cámara del UAV al momento de detectar el objetivo.
Cantidad de características	Es el número de puntos de interés que están extraídos del algoritmo.
Tiempo de Respuesta	El tiempo que demora el UAV en reaccionar frente a un evento.
Velocidad	Indica la velocidad que tiene el objeto al momento de acercarse o alejarse

Es fundamental realizar pruebas en diferentes escenarios, porque la influencia de entorno en el comportamiento es directa en relación con el clima iluminación, dimensiones. Se propone diferentes escenarios o pruebas para detección, el seguimiento y la evasión del objetivo. Siempre y cuando el objetivo esté dentro del rango visible de la cámara como indica la Figura 24.

El número de mediciones realizadas para cada escenario fueron 10 veces, por lo cual se podía estimar la desviación estándar de los datos.

6.1 Escenario 1: Máxima distancia de detección

Las pruebas realizadas en este escenario se hicieron con varios objetos de diferentes características como el tamaño, su uniformidad y forma. Los objetos que aplicaron a este escenario son:

- Objeto 1: Pelota, al ser un objeto simétrico, sus características son pocas puesto que las únicas esquinas que tiene son las figuras de este.
- Objeto 2: Libro, este elemento posee muchas características (esquinas) en su portada
- Objeto 3: Caja de Cartón, a este objeto se le agregaron detalles para tener puntos de interés
- Objeto 4: Mano, es importante comprobar una extremidad, ya que también puede comprometer el vuelo del UAV.

Figura 50. **Prueba de máxima distancia de detección**

En la Figura 50 se puede ver la detección de los diferentes objetos, se realizó la detección a 4 distancias. El promedio de puntos de interés a cada distancia es:

Tabla 9

Datos de evaluación Escenario máxima distancia de detección

Puntos de interés	Distancia 0.5[m]	Distancia 1[m]	Distancia 1.5[m]	Distancia 2 [m]
Objeto 1	74 ± 8	43 ± 13	35 ± 5	7 ± 4
Objeto 2	53 ± 7	45 ± 8	22 ± 7	2 ± 2
Objeto 3	38 ± 10	29 ± 5	15 ± 8	3 ± 4
Objeto 4	39 ± 15	28 ± 14	21 ± 11	8 ± 5

6.2 Escenario 2: Tiempos de respuesta

Este escenario evalúa el tiempo de respuesta que el UAV tiene frente al evento de acercamiento de objetivo. Los mismos objetos del escenario anterior serán utilizados para este ejercicio. En la Figura 51 se muestra alguna de las pruebas realizadas

***Figura 51*. Prueba de tiempo de Respuesta al evento de acercamiento**

Todos los objetos que se acercan a la misma velocidad al UAV y llegan hasta la misma distancia que es 0.5 metros. El tiempo que demora el UAV en tomar la acción de evasión se describe en la siguiente tabla:

Tabla 10

Tiempos de Respuesta en Escenario 2

Objetivo	Objeto 1	Objeto 2	Objeto 3	Objeto 4
Tiempo de Respuesta	860 ± 173 [ms]	984 ± 126[ms]	1253 ± 162[ms]	1046 ± 201[ms]

6.3 Escenario 3: Respuesta de controlador a velocidades bajas

La acción de control es habilitada siempre y cuando la detección de eventos indique que se aproxima el objeto, caso contrario el controlador no realiza ninguna acción de control. Esto es un método que se aplicó porque el UAV puede encontrarse en estado de navegación, y no requiere un control para evasión a menos que la estimación lo indique.

La velocidad de acercamiento está en función del área del objetivo, es decir, la entrada del controlador difuso delta error. Se tomó 10 muestras y la velocidad mínima promedio para detectar el evento de acercamiento es: 1562 $\frac{Pixeles^2}{s}$

Se estimó de forma empírica la velocidad de acercamiento mínima que el objeto debe tomar para ser detectado como un evento hostil, el objeto debe superar la velocidad de 0.5 m/s para ser evadido.

6.4 Escenario 4: Prueba en un entorno cerrado

El entorno influye directamente en el desempeño del algoritmo, puesto que un ambiente cerrado consta de paredes, objetos cercanos, iluminación artificial y poco espacio. El lugar donde se realizó las pruebas es un pasillo de dimensiones 4x10 metros, el cual consta de varios objetos visibles al lente de la cámara del UAV (ver Figura 52).

***Figura 52*. Prueba de evasión en entorno cerrado**

En este entorno resulta que la detección del objeto lo hace correctamente, pero al momento de hacer la acción de evasión los puntos de interés del objetivo se van quedando detrás del objeto, es decir, se confunden con otros objetos en la parte posterior.

6.5 Análisis de Resultados

Para el primer escenario se puede extraer que los objetos a evadir deben tener como mínimo 40 puntos de interés con una tolerancia de ± 10 puntos a una distancia de 0.5 m. Esto debido a que al hacer seguimiento del objeto los puntos de interés que son descartados reducen significativamente los puntos pertenecientes al objeto.

La solución para mejorar la respuesta en el segundo escenario sería redefinir las funciones de pertenencia con respecto al cambio de error. Esto aumentaría la reacción inmediata del UAV frente al evento de acercamiento del objeto.

Claramente en el escenario tres el umbral de detección sería un factor clave para la evasión inmediata, es decir, la velocidad mínima para detectar el evento de acercamiento debe ser menor a la encontrada, hasta un cierto valor que evite falsos positivos.

El problema del escenario cuatro es notable, se propone segmentar discriminando para ambientes cerrado como el pasillo, una velocidad de evasión prudente para que los puntos de interés no terminen saliendo el objeto de interés.

CAPITULO VII

CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

Existen varias técnicas y algoritmos para la detección de puntos de interés en una imagen, entre los más relevantes están SIRF, SURF, ORB y el aplicado en este proyecto Shi-Tomasi. El algoritmo ORB tiene una mejor respuesta en cuanto a cantidad de puntos de interés, pero su tiempo de respuesta sobrepasa el necesario para cumplir con la tasa de transmisión de video del UAV, el algoritmo de Shi-Tomasi mantenía el mismo número de puntos interés a haber movimiento en la imagen.

En este proyecto se evaluaron algunos métodos para el seguimiento de objetos entre los cuales está el emparejamiento de imágenes, los seguidores KCF, MOSSE y TLD. Al realizar emparejamiento se encuentra con varios problemas de seguimiento puesto que el objeto presenta cambios de intensidad de luz al tener movimiento, una vez que el emparejamiento pierde de vista al objeto no lo vuelve a encontrar. Los seguidores de objetos mencionados son facilitados por la librería de procesamiento de imágenes OpenCV, dichos algoritmos realizaban seguimiento, pero había momentos que entregaban falsos positivos. En conclusión, se realizó un método a partir de estimación de flujo óptico solo con el seguimiento de los puntos de interés, ya que no se perdían una vez detectados.

Al obtener resultados esperados en el seguimiento de objetos se logró hacer integración con la estimación de proximidad y evasión del objetivo. El porcentaje reducido de falsos positivos era el

suficiente para la estabilidad del método, prestando el tiempo óptimo para estimar la proximidad y tomar decisiones de control ante los eventos detectados.

Para realizar el control del UAV el que mejor se ajustó al proyecto es el controlador difuso, debido a la posibilidad de usar expresiones o eventos imprecisos, por ejemplo, se tiene el caso que el objeto se acerca super rápido, pero no está a una distancia muy cercana al UAV, la reacción de evasión debería ser baja como preparación para una evasión extrema. Los parámetros del controlador se ajustan a este tipo de eventualidades.

Todo el procesamiento de las imágenes y el control del UAV están implementados de manera remota en un computador de rendimiento medio con un sistema operativo multiplataforma, y un lenguaje de programación que permite su portabilidad. Por lo tanto, este sistema puede ser implementado en sistemas embebidos de tamaño reducido, como es el caso de computadores Stick, esto trae consigo una mayor autonomía y movilización.

Las pruebas realizadas en entornos abiertos y cerrados con muy significativas, porque el rendimiento del sistema fue muy diferente en ambos casos, teniendo como conclusión que el nivel de iluminación debe ser muy controlado para tener una respuesta competente a los eventos mencionados en este proyecto. Por lo tanto, el rendimiento del proyecto es mejor en entornos abiertos.

7.2 Recomendaciones

Para proyectos que con UAVs que traten el tema de navegación autónoma, es recomendable aplicarlo en UAVs autocontrolados como el usado en este proyecto, el parrot Bebop 2, es decir, que el control de estabilidad de vuelo esté integrado en el mismo UAV para que el proyecto se enfoque directamente con los temas de navegación como, por ejemplo, planificación de trayectorias, seguimiento, evasión, entre otros.

De igual manera si se requiere hacer procesamiento de imágenes que provienen de una UAV, es recomendable aplicarlo en drones que vienen con herramientas de desarrollo como es el caso del SDK de Parrot.

Al momento de realizar pruebas en diferentes entornos tener en cuenta la seguridad del drone y de las personas, en este caso, se colocó protección en las hélices del drone para interiores.

REFERENCIAS BIBLIOGRÁFICAS

Aguaiza Guerrero, C. H. (2018). Sistema de estimación de número de personas en tiempo real durante misiones de reconocimiento del Ejército Ecuatoriano utilizando vehículos aéreos no tripulados Multirotor. Recuperado a partir de http://repositorio.espe.edu.ec/handle/21000/15022

Aguilar, W., & Angulo Bahón, C. (2014). Estabilización de vídeo en micro vehículos aéreos y su aplicación en la detección de caras. *Revista Digital Congreso de Ciencia y Tecnología : Memorias. Sesiones Técnicas*, 155–160. Recuperado a partir de https://upcommons.upc.edu/handle/2117/23100

Aguilar, W., Casaliglla, V., Pólit, J., Aguilar, W. G., Casaliglla, V. P., & Pólit, J. L. (2017). Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles. *Electronics*, *6*(1), 10. https://doi.org/10.3390/electronics6010010

Aguilar, W. G., Abad, V., Ruiz, H., Aguilar, J., & Aguilar-Castillo, F. (2017). RRT-Based Path Planning for Virtual Bronchoscopy Simulator (pp. 155–165). Springer, Cham. https://doi.org/10.1007/978-3-319-60928-7_13

Aguilar, W. G., & Angulo Bahón, C. (2013). Estabilización robusta de vídeo basada en diferencia de nivel de gris. *Memorias del VIII Congreso de Ciencia y Tecnología*. Recuperado a partir de https://upcommons.upc.edu/handle/2117/19878

Aguilar, W. G., & Angulo, C. (2012). Compensación de los Efectos Generados en la Imagen por el Control de Navegación del Robot Aibo ERS 7. En *Memorias del VII Congreso de Ciencia y Tecnología ESPE 2012* (pp. 165–170).

Aguilar, W. G., & Angulo, C. (2012). Compensación y Aprendizaje de Efectos Generados en la Imagen durante el Desplazamiento de un Robot. En *X Simposio CEA de Ingeniería de Control*.

Aguilar, W. G., & Angulo, C. (2014a). Real-time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing*, *2014*(1), 46.

Aguilar, W. G., & Angulo, C. (2014b). Real-time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing*, *2014*(1), 46. https://doi.org/10.1186/1687-5281-2014-46

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

Aguilar, W. G., & Angulo, C. (2016). Real-Time Model-Based Video Stabilization for Microaerial Vehicles. *Neural Processing Letters*, *43*(2), 459–477. https://doi.org/10.1007/s11063-015-9439-0

Aguilar, W. G., Angulo, C., & Costa-Castello, R. (2017). Autonomous Navigation Control for Quadrotors in Trajectories Tracking (pp. 287–297). Springer, Cham. https://doi.org/10.1007/978-3-319-65298-6_27

Aguilar, W. G., Angulo, C., & Pardo, J. A. (2017). Motion intention optimization for multirotor robust video stabilization. En *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)* (pp. 1–4). IEEE. https://doi.org/10.1109/CHILECON.2017.8229689

Aguilar, W. G., Casaliglla, V. P., Pólit, J. L., Abad, V., & Ruiz, H. (2017). Obstacle Avoidance for

Flight Safety on Unmanned Aerial Vehicles (pp. 575–584). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_49

Aguilar, W. G., Cobeña, B., Rodriguez, G., Salcedo, V. S., & Collaguazo, B. (2018). SVM and RGB-D Sensor Based Gesture Recognition for UAV Control (pp. 713–719). Springer, Cham.

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Parra, H., & Ruiz, H. (2017). Pedestrian Detection for UAVs Using Cascade Classifiers with Meanshift. En *2017 IEEE 11th International Conference on Semantic Computing (ICSC)* (pp. 509–514). IEEE. https://doi.org/10.1109/ICSC.2017.83

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Angulo, C. (2017). Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps (pp. 563–574). Springer, Cham.

Aguilar, W. G., Luna, M. A., Moya, J. F., Abad, V., Ruiz, H., Parra, H., & Lopez, W. (2017). Cascade Classifiers and Saliency Maps Based People Detection (pp. 501–510). Springer, Cham. https://doi.org/10.1007/978-3-319-60928-7_42

Aguilar, W. G., Manosalvas, J. F., Guillén, J. A., & Collaguazo, B. (2018). Robust Motion Estimation Based on Multiple Monocular Camera for Indoor Autonomous Navigation of Micro Aerial Vehicle (pp. 547–561). Springer, Cham. https://doi.org/10.1007/978-3-319-95282-6_39

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017a). RRT* GL Based Optimal Path Planning for Real-Time Navigation of UAVs (pp. 585–595). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_50

Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017b). RRT* GL Based Path Planning for Virtual Aerial Navigation (pp. 176–184). Springer, Cham. https://doi.org/10.1007/978-3-319-60922-5_13

Aguilar, W. G., Quisaguano, F. J., Rodríguez, G. A., Alvarez, L. G., Limaico, A., & Sandoval, D. S. (2018). Convolutional Neuronal Networks Based Monocular Object Detection and Depth Perception for Micro UAVs (pp. 401–410). Springer, Cham.

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). On-Board Visual SLAM on a UGV Using a RGB-D Camera (pp. 298–308). Springer, Cham. https://doi.org/10.1007/978-3-319-65298-6_28

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing (pp. 410–419). Springer, Cham. https://doi.org/10.1007/978-3-319-60928-7_35

Aguilar, W. G., Rodríguez, G. A., Álvarez, L., Sandoval, S., Quisaguano, F., & Limaico, A. (2017). Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing (pp. 596–606). Springer, Cham. https://doi.org/10.1007/978-3-319-59147-6_51

Aguilar, W. G., Salcedo, V. S., Sandoval, D. S., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation (pp. 94–105). Springer, Cham. https://doi.org/10.1007/978-3-319-71011-2_8

Aguilar, W. G., Sandoval, D. S., Caballeros, J., Alvarez, L. G., Limaico, A., Rodríguez, G. A., & Quisaguano, F. J. (2018). Graph Based RRT Optimization for Autonomous Mobile Robots (pp. 12–21). Springer, Cham. https://doi.org/10.1007/978-3-030-02698-1_2

Aguilar, W., Morales, S., Aguilar, W. G., & Morales, S. G. (2016). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics*, *5*(4), 70. https://doi.org/10.3390/electronics5040070

Andrea, C. C., Byron, J. Q., Jorge, P. I., Inti, T. C., & Aguilar, W. G. (2018). Geolocation and Counting of People with Aerial Thermal Imaging for Rescue Purposes (pp. 171–182). Springer, Cham. https://doi.org/10.1007/978-3-319-95270-3_12

Aplicaciones y usos, intelligenia DYNAMICS. (s/f).

Barron, J. L., Fleet, D. J., & Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, *12*(1), 43–77.

Bauer, P., Hiba, A., Vanek, B., Zarandy, A., & Bokor, J. (2016). Monocular image-based time to collision and closest point of approach estimation. En *2016 24th Mediterranean Conference on Control and Automation (MED)* (pp. 1168–1173). IEEE.

BEAUDET, & R., P. (1978). Rotationally invariant image operators. *Proc. 4th Int. Joint Conf. Pattern Recog, Tokyo, Japan, 1978*.

Bertozzi, M., Broggi, A., Cellario, M., Fascioli, A., Lombardi, P., & Porta, M. (2002, julio). Artificial vision in road vehicles. *Proceedings of the IEEE*.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698.

Cognifit. (2017). Percepción Visual - Habilidad Cognitiva.

Comaniciu, D., Ramesh, V., & Meer, P. (s/f). Real-time tracking of non-rigid objects using mean shift. En *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR*

2000 (Cat. No.PR00662) (Vol. 2, pp. 142–149). IEEE Comput. Soc.

Control Luis Eduardo Romero Paredes, E. Y. (2014). *Escuela Politécnica Nacional Facultad De Ingeniería Eléctrica Y Electrónica Diseño Y Construcción De Un Módulo De Control PID Para La Estabilización De Un Cuadricóptero*.

Crespi de los Reyes, X. (1000). UAV y sus aplicaciones en obra civil.

Emmanuel Jiménez Camacho. (2009). *Generación, Representación y Principios de Procesamiento Digital de Imágenes 2.1 Cámara pinhole y formación de una imagen*.

Fabio Nelli. (2017). OpenCV & Python - Harris Corner Detection - method to detect corners.

FernándezFern, S., Directores, L., Macho, Z., & Luis Porras Gaí an, J. (2014). *Modelado y control de un cuadricóptero*.

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.

Harris, C., & Stephens, M. (s/f). *A COMBINED CORNER AND EDGE DETECTOR*.

Hassaballah, M., Abdelmgeid, A. A., & Alshazly, H. A. (2016). Image Features Detection, Description and Matching (pp. 11–45). Springer, Cham.

Hassanalian, M., & Abdelkefi, A. (2017). Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, *91*, 99–131.

Jara-Olmedo, A., Medina-Pazmiño, W., Mesías, R., Araujo-Villaroel, B., Aguilar, W. G., & Pardo, J. A. (2018). Interface of Optimal Electro-Optical/Infrared for Unmanned Aerial Vehicles (pp. 372–380). Springer, Cham. https://doi.org/10.1007/978-3-319-78605-6_32

Jara-Olmedo, A., Medina-Pazmino, W., Tozer, T., Aguilar, W. G., & Pardo, J. A. (2018). E-services from Emergency Communication Network: Aerial Platform Evaluation. En *2018 International Conference on eDemocracy & eGovernment (ICEDEG)* (pp. 251–256). IEEE. https://doi.org/10.1109/ICEDEG.2018.8372336

JEAN-YVES, & B. (1999). Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. *Intel Corporation, OpenCV Documents*.

Jianbo Shi, & Tomasi. (1994). Good features to track. En *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94* (pp. 593–600). IEEE Comput. Soc. Press.

José Muñoz Pérez. (2010a). *Capítulo 1 Imágenes digitales y sus propiedades 1. Introducción*.

José Muñoz Pérez. (2010b). *Capítulo 2 Representación de imágenes digitales 2.1 Representación de una imagen digital: matrices vinculadas*.

Kendoul, F. (2012). Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, *29*(2), 315–378.

Kheng, L. W. (s/f). *Principal Component Analysis Leow Wee Kheng CS4243 Computer Vision and Pattern Recognition CS4243 Principal Component Analysis 1*.

Kote, P. (2016). Parrot Bebop 2, análisis: autonomía y facilidad de vuelo ideales para entrar en el mundo de los drones.

Lindeberg, T., & Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of applied statistics*, *21*, 224--270.

Liu, Z., Zhang, Y., Yu, X., & Yuan, C. (2016). *Unmanned surface vehicles: An overview of*

developments and challenges. *Annual Reviews in Control* (Vol. 41).

Lucas, B. D., & Kanadc, T. (s/f). *Optical Navigation by the Method of Differences*.

Mani Monajjemi. (2015a). bebop_autonomy - ROS Driver for Parrot Bebop Drone (quadrocopter) 1.0 & 2.0 — bebop_autonomy indigo-devel documentation.

Mani Monajjemi. (2015b). Installation — bebop_autonomy indigo-devel documentation.

Marr, D., & Poggio, T. (1979). A Computational Theory of Human Stereo Vision. *Proceedings of the Royal Society B: Biological Sciences*, *204*(1156), 301–328.

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.

Mora Mariño, D. L., & Páez Melo, A. O. (2012). Detección de objetos móviles en una escena utilizando flujo óptico.

Ocegueda, O. (2003). *Flujó optico*.

Olivares-Mendez, M. A., Mejias, L., Campoy, P., & Mellado-Bataller, I. (2012). Quadcopter see and avoid using a fuzzy controller (pp. 1239–1244).

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., León, G., & Jara-Olmedo, A. (2017). Math Model of UAV Multi Rotor Prototype with Fixed Wing Aerodynamic Structure for a Flight Simulator (pp. 199–211). Springer, Cham. https://doi.org/10.1007/978-3-319-60922-5_15

Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., Reyes, R. P., & Montoya, L. (2017). Vertical

take off and landing with fixed rotor. En *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)* (pp. 1–6). IEEE. https://doi.org/10.1109/CHILECON.2017.8229691

Parrot. (2015). Dron Parrot Bebop 2 | Sitio Web Official de Parrot.

Parrot Developers. (s/f). Referencia de API ARDroneSDK3.

Portillo-Portillo, J., Sánchez-Pérez, G., Olivares-Mercado, J., & Pérez-Meana, H. (2014). Detección de Movimiento de Vehículos en Secuencias de Video Basados en la Diferencia Absoluta entre Fotogramas y la Combinación de Bordes. *Información tecnológica*, *25*(5), 129–136.

Procisur, & Di Leo, N. C. (2015). *Drones: nueva dimensión de la teledetección agroambiental y nuevo paradigma para la agricultura de precisión*. *Agromensajes - Facultad de Ciencias Agrarias . Universidad Nacional del Rosario* (Vol. 41).

Quan, Q. (2017a). Basic Composition. En *Introduction to Multicopter Design and Control* (pp. 31–55). Singapore: Springer Singapore.

Quan, Q. (2017b). Introduction. En *Introduction to Multicopter Design and Control* (pp. 1–27). Singapore: Springer Singapore.

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

Radim Stuchlík, Z. S., & , Kamil Láska, P. K. (s/f). Unmanned Aerial Vehicle – Efficient mapping tool available for recent research in polar regions.

Riascos Segura, J. S., Cardona Gallego, J. A., & Electrónico, I. (2015). determinación de la cinemática de objetos móviles bajo condiciones controladas mediante imágenes empleando

técnicas de flujo óptico.

Richter, C., Bry, A., & Roy, N. (2016). Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments (pp. 649–666). Springer, Cham.

Robotics: Science and Systems VI - Yoky Matsuoka, Hugh Durrant-Whyte, José Neira - Google Libros. (s/f).

Rojas Hernández, I. R., Ramón, S., Ortigoza, M., María, A., & Molina, V. ♦. (2007). *La visión Artificial en la Robótica*.

Rojas, R. (2010). *Lucas-Kanade in a Nutshell*.

ROS.org. (2017). cv_bridge-Tutoriales.

Rosten, E., & Drummond, T. (2005). Fusing points and lines for high performance tracking. En *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (p. 1508–1515 Vol. 2). IEEE.

Salahat, E., & Qasaimeh, M. (2017). Recent advances in features extraction and description algorithms: A comprehensive survey. En *Proceedings of the IEEE International Conference on Industrial Technology* (pp. 1059–1063).

Salcedo Peña, V. S. (2018). Aterrizaje automático de un vehículo aéreo no tripulado basado en seguimiento de puntos de interés para superficies móviles.

Salgado, M. F., Tierra, A., Sandoval, D. S., & Aguilar, W. G. (s/f). Travel Time Estimation of Public Transport Networks Based on Commercial Incidence Areas in Quito Historic Center.

Sayem, A. (2016). Vision-Aided Navigation for Autonomous Vehicles Using Tracked Feature

Points. *Dissertations and Theses*.

Shapiro, L. (2000). *Fundamentos de Visión por Computador*.

Thesis, C. K.-M., University, T., & 2012, U. (s/f). Odometry from rgb-d cameras for autonomous quadrocopters. *Citeseer*.

Tomasi, C., & Kanade, T. (1991). Shape and Motion from Image Streams: a Factorization Method—Part 3 Detection and Tracking of Point Features.

Valero Lavid, C. (2017). Desarrollo de aplicaciones basadas en visión con entorno ROS para el dron Bebop 2.

Vedaldi, A. (2008). An implementation of SIFT detector and descriptor. *International Journal*, *1*, 1–7.

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

Zhou, H., Yuan, Y., & Shi, C. (2009). Object tracking using SIFT features and mean shift. *Computer Vision and Image Understanding*, *113*(3), 345–352.

Zingg, S., Scaramuzza, D., Weiss, S., & Siegwart, R. (2010). MAV navigation through indoor corridors using optical flow. En *2010 IEEE International Conference on Robotics and Automation* (pp. 3361–3368). IEEE.