



**ESPE**  
UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**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: SISTEMA DE NAVEGACIÓN AUTÓNOMO BASADO EN  
VISIÓN PARA PISTAS DE CARRERAS DE DRONES CON MARCAS DE  
SUPERFICIE**

**AUTOR:**

**GRIJALVA CAISACHANA, SANTIAGO ANDRÉS**

**DIRECTOR:**

**AGUILAR CASTILLO, WILBERT GEOVANNY**

**SANGOLQUÍ**

**2019**



**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA,  
AUTOMATIZACIÓN Y CONTROL**

**CERTIFICACIÓN**

Certifico que el trabajo de titulación, “**SISTEMA DE NAVEGACIÓN AUTÓNOMO BASADO EN VISIÓN PARA PISTAS DE CARRERAS DE DRONES CON MARCAS DE SUPERFICIE**” fue realizado por el señor **GRIJALVA CAISACHANA , SANTIAGO ANDRÉS** 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.

Sangolqui, 16 de octubre de 2019

Wilbert Geovanny Aguilar Castillo

CC: 0703844696

**AUTORÍA DE RESPONSABILIDAD**



**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, **GRIJALVA CAISACHANA, SANTIAGO ANDRÉS**, declaro que el contenido, ideas y criterios del trabajo de titulación: “**SISTEMA DE NAVEGACIÓN AUTÓNOMO BASADO EN VISIÓN PARA PISTAS DE CARRERAS DE DRONES CON MARCAS DE SUPERFICIE**” es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Una firma manuscrita en tinta azul, que parece ser 'S. Grijalva', sobre una línea horizontal que sirve como línea de firma.

Sangolquí, 16 de octubre de 2019

Santiago Andrés Grijalva Caisachana

CC: 1721705612



**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y  
TELECOMUNICACIONES**

**CARRERA DE INGENIERÍA EN ELECTRÓNICA,  
AUTOMATIZACIÓN Y CONTROL**

**AUTORIZACIÓN**

Yo, **GRIJALVA CAISACHANA, SANTIAGO ANDRÉS** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: “**SISTEMA DE NAVEGACIÓN AUTÓNOMO BASADO EN VISIÓN PARA PISTAS DE CARRERAS DE DRONES CON MARCAS DE SUPERFICIE**” en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.



Sangolquí, 16 de octubre de 2019

Santiago Andrés Grijalva Caisachana

CC: 1721705612

## DEDICATORÍA

*A mi padre César y a mi madre Ximena,*

*ellos son el motor de mi vida,*

*este es su logro.*

*A cada una de las personas que forman parte de mi vida.*

*Por último, a todas las personas*

*que nunca se rinden y creen en el trabajo duro.*

*Santiago Andrés Grijalva Caisachana*

## AGRADECIMIENTO

*Agradezco a mis padres por brindarme las condiciones necesarias,  
la mejor vida posible.*

*Agradezco a la vida por rodearme de las personas indicadas,  
en especial a mis amigos,  
indispensables para ser lo que soy ahora.*

*Santiago Andrés Grijalva Caisachana*

## ÍNDICE DE CONTENIDOS

<b>CERTIFICACIÓN .....</b>	<b>i</b>
<b>AUTORÍA DE RESPONSABILIDAD .....</b>	<b>ii</b>
<b>AUTORIZACIÓN .....</b>	<b>iii</b>
<b>DEDICATORÍA .....</b>	<b>iv</b>
<b>AGRADECIMIENTO.....</b>	<b>v</b>
<b>ÍNDICE DE CONTENIDOS .....</b>	<b>vi</b>
<b>ÍNDICE DE TABLAS .....</b>	<b>xi</b>
<b>ÍNDICE DE FIGURAS.....</b>	<b>xiii</b>
<b>RESUMEN.....</b>	<b>xix</b>
<b>ABSTRACT .....</b>	<b>xx</b>
<b>GLOSARIO .....</b>	<b>xxi</b>
<b>CAPÍTULO I .....</b>	<b>1</b>
<b>1. DESCRIPCIÓN DEL PROYECTO DE INVESTIGACIÓN.....</b>	<b>1</b>
1.1 Planteamiento del problema .....	1
1.2 Antecedentes.....	2



1.3 Justificación e importancia .....	5
1.4 Alcance del proyecto .....	9
1.5 Objetivos .....	11
1.5.1 Objetivo General.....	11
1.5.2 Objetivos Específicos.....	11
<b>CAPÍTULO II.....</b>	<b>12</b>
<b>2. MARCO TEÓRICO.....</b>	<b>12</b>
2.1 Vehículos aéreos no tripulados.....	12
2.1.1 Dinámica del movimiento de un cuadricóptero.....	13
2.1.2 Control de UAVs.....	15
2.1.3 Estimación de estado con Filtro de Kalman .....	17
2.2 Formación de imágenes .....	17
2.3 Caracterización de imágenes .....	19
2.3.1 Detección de características.....	20
2.4 Procesamiento de imágenes.....	26
2.4.1 Modelo de color.....	27
2.4.2 Suavizado de imágenes.....	29



2.4.3 Transformaciones morfológicas .....	31
2.4.4 Transformaciones geométricas .....	32
<b>CAPÍTULO III .....</b>	<b>37</b>
<b>3. DESCRIPCIÓN DEL SISTEMA .....</b>	<b>37</b>
3.1 Descripción del hardware del sistema .....	37
3.1.1 Estación terrestre .....	37
3.1.2 Estación aérea (UAV).....	37
3.2 Descripción del software del sistema .....	42
3.2.1 <i>Robot Operating System (ROS)</i> .....	42
3.2.2 Comunicación entre <i>ROS</i> y <i>Bebop 2</i> .....	44
<b>CAPÍTULO IV .....</b>	<b>58</b>
<b>4. DETECCIÓN Y SEGUIMIENTO DE LA TRAYECTORIA .....</b>	<b>58</b>
4.1 Pre procesamiento de las imágenes terrestres.....	59
4.1.1 Conversión a espacio de color HSV .....	60
4.1.2 Control de iluminación presente en las imágenes de la cámara .....	61
4.1.3 Enmascaramiento de la trayectoria.....	65
4.2 Detección de trayectoria .....	68

4.2.1 Detección de puntos de interés .....	68
4.2.2 Estimación del punto medio de la trayectoria .....	72
4.2.3 Seguimiento de trayectoria .....	81
<b>CAPÍTULO V .....</b>	<b>89</b>
<b>5. DETECCIÓN Y EVASIÓN DE OBSTÁCULOS .....</b>	<b>89</b>
5.1 Pre procesamiento de imágenes frontales.....	89
5.1.1 Enmascaramiento del obstáculo .....	90
5.2 Detección de obstáculo.....	93
5.2.1 Detección de contornos .....	93
5.2.2 Estimación del centro de masa del obstáculo .....	95
5.2.3 Evasión del obstáculo .....	104
<b>CAPÍTULO VI .....</b>	<b>119</b>
<b>6. EXPERIMENTOS Y ANÁLISIS DE RESULTADOS .....</b>	<b>119</b>
6.1 Pruebas experimentales de seguimiento de trayectoria .....	120
6.1.1 Tiempo de seguimiento de trayectoria.....	122
6.1.2 Error de seguimiento de la trayectoria.....	123
6.2 Pruebas experimentales en trayectoria con presencia de obstáculos .....	128

6.2.1 Seguimiento de trayectoria en presencia de obstáculos.....	130
6.2.2 Tiempo de ejecución del circuito.....	136
6.2.3 Tiempo de evasión de obstáculos.....	137
6.2.4 Observaciones en tercera persona.....	139
<b>CAPÍTULO VII.....</b>	<b>140</b>
<b>7. CONCLUSIONES Y RECOMENDACIONES .....</b>	<b>140</b>
7.1 Conclusiones.....	140
7.2 Recomendaciones .....	141
<b>8. REFERENCIAS .....</b>	<b>143</b>

## ÍNDICE DE TABLAS

<b>Tabla 1</b> <i>Movimientos de un cuadricóptero según la velocidad de los rotores</i> .....	15
<b>Tabla 2</b> <i>Transformaciones geométricas en coordenadas de dos dimensiones</i> .....	36
<b>Tabla 3</b> <i>Características del sistema interno, construcción y de conectividad del Bebop 2</i> .....	38
<b>Tabla 4</b> <i>Parámetros de movimiento configurables del Bebop 2</i> .....	40
<b>Tabla 5</b> <i>Características de la cámara, captura de imagen y video</i> .....	41
<b>Tabla 6</b> <i>Parámetros configurables de pilotaje</i> .....	46
<b>Tabla 7</b> <i>Parámetros de movimiento configurables</i> .....	47
<b>Tabla 8</b> <i>Parámetros de red configurables</i> .....	48
<b>Tabla 9</b> <i>Parámetros de imagen configurables</i> .....	48
<b>Tabla 10</b> <i>Parámetros de GPS configurables</i> .....	49
<b>Tabla 11</b> <i>Parámetros de pilotaje, velocidad e imagen reconfigurados</i> .....	50
<b>Tabla 12</b> <i>Publicadores de la clase "bebop2"</i> .....	52
<b>Tabla 13</b> <i>Valores para el control de movimiento de la cámara virtual</i> .....	54
<b>Tabla 14</b> <i>Parámetros para el control de movimiento del UAV</i> .....	54
<b>Tabla 15</b> <i>Suscriptores de la clase "bebop2"</i> .....	55
<b>Tabla 16</b> <i>Rango de valores para enmascaramiento de trayectoria</i> .....	65
<b>Tabla 17</b> <i>Tiempos de detección y número de puntos característicos para aproximadamente</i> <i>2000 lux</i> .....	71
<b>Tabla 18</b> <i>Tiempos de detección y número de puntos característicos para aproximadamente</i> <i>5000 lux</i> .....	71
<b>Tabla 19</b> <i>Error medio cuadrático entre mediana y punto marcado</i> .....	74

<b>Tabla 20</b> <i>Error medio cuadrático entre media y punto marcado</i> .....	76
<b>Tabla 21</b> <i>Error medio cuadrático entre estimación con Filtro de Kalman y punto marcado</i> .....	80
<b>Tabla 22</b> <i>Rango de valores para enmascaramiento del obstáculo</i> .....	90
<b>Tabla 23</b> <i>Error medio cuadrático para la estimación de posición del obstáculo</i> .....	98
<b>Tabla 24</b> <i>Varianza de las mediciones de la posición del obstáculo</i> .....	98
<b>Tabla 25</b> <i>Tiempo de seguimiento de trayectoria para tres intentos</i> .....	122
<b>Tabla 26</b> <i>Error de seguimiento promedio para los circuitos 1, 2 y 3</i> .....	134
<b>Tabla 27</b> <i>Tiempo de ejecución de circuito en presencia de obstáculos</i> .....	136
<b>Tabla 28</b> <i>Tiempos de evasión de obstáculos para el Circuito 1</i> .....	138
<b>Tabla 29</b> <i>Tiempos de evasión de obstáculos para el Circuito 2</i> .....	138
<b>Tabla 30</b> <i>Tiempos de evasión de obstáculos para el Circuito 3</i> .....	138
<b>Tabla 31</b> <i>Visualizaciones en tercera persona para tele-operadores</i> .....	139

## ÍNDICE DE FIGURAS

<i>Figura 1.</i> Parrot Bebop 2 .....	5
<i>Figura 2.</i> Tipos de obstáculos definidos en la IROS 2018 .....	8
<i>Figura 3.</i> Esquema del circuito de la competición Autonomous Drone Racing IROS 2017 .....	8
<i>Figura 4.</i> Diagrama esquemático del sistema.....	10
<i>Figura 5.</i> Eje de referencia y eje del cuerpo de un cuadricóptero .....	13
<i>Figura 6.</i> Esquema de control PID .....	16
<i>Figura 7.</i> Diferencia de incidencia de rayos en la barrera con y sin una apertura.....	18
<i>Figura 8.</i> Esquema de proceso de formación de imagen en una cámara digital .....	18
<i>Figura 9.</i> Proceso de formación de una imagen digital, sin post procesamiento .....	19
<i>Figura 10.</i> Diferencia de Gaussianas adyacentes para espacios de escala diferentes .....	22
<i>Figura 11.</i> De izquierda a derecha: Derivadas Gaussianas de segundo orden en y (Dyy), dirección xy (Dxy) y su aproximación en la misma dirección, respectivamente .....	23
<i>Figura 12.</i> Punto de interés bajo test y los 16 píxeles adyacentes para el algoritmo FAST.....	24
<i>Figura 13.</i> Representación del modelo de color RGB.....	28
<i>Figura 14.</i> Modelo de color HSV .....	29
<i>Figura 15.</i> Transformaciones morfológicas .....	32
<i>Figura 16.</i> Deformaciones de imágenes causadas por la modificación del dominio .....	33
<i>Figura 17.</i> Transformaciones geométricas para el plano bidimensional .....	33
<i>Figura 18.</i> Parrot Bebop 2 vista frontal.....	38
<i>Figura 19.</i> Movimientos independientes del Bebop 2.....	40
<i>Figura 20.</i> Campo de visión del streaming de video del Bebop 2.....	41

<b>Figura 21.</b> Grafo de computación de ROS .....	43
<b>Figura 22.</b> Intercambio de información entre nodos en ROS.....	44
<b>Figura 23.</b> Proceso de conversión de mensaje de ROS a una imagen.....	56
<b>Figura 24.</b> Detección de trayectoria .....	59
<b>Figura 25.</b> a) Imagen en BGR, b) Canal H, c) Canal S, d) Canal V.....	61
<b>Figura 26.</b> Diagrama de flujo de subproceso de control de luminosidad de imagen .....	62
<b>Figura 27.</b> Interpolación lineal de la señal de control y el porcentaje de iluminación.....	64
<b>Figura 28.</b> Frame con control de iluminación (izquierda) vs. frame sin control de iluminación (derecha).....	64
<b>Figura 29.</b> Matiz de colores del canal H.....	65
<b>Figura 30.</b> Máscaras para el frame 1, rango 1 (izquierda), rango 2 (derecha) .....	66
<b>Figura 31.</b> Máscaras para el frame 2, rango 1 (izquierda), rango 2 (derecha) .....	66
<b>Figura 32.</b> Máscara sin y con transformación morfológica.....	67
<b>Figura 33.</b> Frames 1 y 2 enmascarados .....	67
<b>Figura 34.</b> Detección de trayectoria .....	68
<b>Figura 35.</b> Región de interés para la detección de puntos de interés .....	69
<b>Figura 36.</b> Rango visible para la región de interés.....	70
<b>Figura 37.</b> Tiempo de detección medio de puntos característicos de la trayectoria.....	71
<b>Figura 38.</b> Media de número de puntos característicos detectados en la trayectoria .....	72
<b>Figura 39.</b> Diagrama de flujo de proceso de estimación de punto medio con mediana.....	73
<b>Figura 40.</b> Error medio cuadrático de la mediana de la trayectoria .....	74
<b>Figura 41.</b> Diagrama de flujo de proceso de estimación de punto medio con media.....	75
<b>Figura 42.</b> Error medio cuadrático de la media de la trayectoria .....	76



<b>Figura 43.</b> Diagrama de flujo de proceso de estimación de punto medio con Filtro de Kalman..	80
<b>Figura 44.</b> Error medio cuadrático de la Estimación con Filtro de Kalman de la trayectoria .....	81
<b>Figura 45.</b> Diagrama de flujo de seguimiento de trayectoria .....	82
<b>Figura 46.</b> Tren de impulsos para la obtención del modelo servo visual .....	84
<b>Figura 47.</b> Estimación de movimiento angular acumulado.....	85
<b>Figura 48.</b> Comparación de modelo estimado para Yaw vs. Datos de validación del movimiento angular acumulado .....	86
<b>Figura 49.</b> Respuesta al escalón unitario del conjunto Planta-Controlador para el movimiento angular alrededor del eje Z (Yaw).....	87
<b>Figura 50.</b> Interpolación lineal para el control de velocidad lineal en x .....	88
<b>Figura 51.</b> Máscaras para diferentes frames que contienen un obstáculo .....	91
<b>Figura 52.</b> Comparación entre frames con y sin transformación morfológica.....	92
<b>Figura 53.</b> Frames 1 y 2 enmascarados .....	92
<b>Figura 54.</b> Proceso de detección de obstáculos .....	93
<b>Figura 55.</b> Detección de contornos.....	94
<b>Figura 56.</b> Encapsulamiento del obstáculo.....	95
<b>Figura 57.</b> Centro del recuadro de encapsulamiento .....	96
<b>Figura 58.</b> Detecciones erróneas del centro del obstáculo por cambios de iluminación.....	99
<b>Figura 59.</b> Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 1.....	100
<b>Figura 60.</b> Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 2.....	100

<b>Figura 61.</b> Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 3.....	101
<b>Figura 62.</b> Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 4.....	101
<b>Figura 63.</b> Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 5.....	102
<b>Figura 64.</b> Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 6.....	102
<b>Figura 65.</b> Proceso de estimación de centro de masa del obstáculo.....	103
<b>Figura 66.</b> Dimensiones del obstáculo .....	104
<b>Figura 67.</b> Distancias umbral para atravesar el obstáculo .....	105
<b>Figura 68.</b> Distancia entre el dron y el obstáculo cuando se ha alcanzado el área umbral .....	107
<b>Figura 69.</b> Algoritmo de evasión de obstáculos .....	109
<b>Figura 70.</b> Estimación de movimiento acumulado en el eje Y.....	111
<b>Figura 71.</b> Estimación de movimiento acumulado en el eje Z.....	112
<b>Figura 72.</b> Comparación de modelo estimado para Roll vs. Datos de validación del movimiento a lo largo del eje Y acumulado.....	113
<b>Figura 73.</b> Respuesta al escalón unitario del conjunto Planta-Controlador para el movimiento lineal a lo largo del eje Y (Roll).....	114
<b>Figura 74.</b> Comparación de modelo estimado para altitud vs. Datos de validación del movimiento a lo largo del eje Z acumulado .....	115
<b>Figura 75.</b> Respuesta al escalón unitario del conjunto Planta-Controlador para el movimiento lineal a lo largo del eje Z (Altitud) .....	116

<b>Figura 76.</b> Transición Obstáculo-Trayectoria .....	117
<b>Figura 77.</b> Algoritmo de navegación autónoma para detección y seguimiento de trayectoria, y detección y evasión de obstáculos.....	118
<b>Figura 78.</b> Trayectoria de pista de carreras .....	121
<b>Figura 79.</b> Perspectiva para análisis de seguimiento de trayectoria.....	121
<b>Figura 80.</b> Tiempo de seguimiento promedio para diferentes niveles de experticia.....	122
<b>Figura 81.</b> Trayectoria de referencia .....	123
<b>Figura 82.</b> Trayectoria recorrida por el dron con el algoritmo de seguimiento de trayectoria....	124
<b>Figura 83.</b> Trayectoria recorrida por el dron bajo el mando del tele-operador experto .....	124
<b>Figura 84.</b> Trayectoria recorrida por el dron bajo el mando del tele-operador intermedio.....	125
<b>Figura 85.</b> Trayectoria recorrida por el dron bajo el mando del tele-operador novato .....	125
<b>Figura 86.</b> Error medio cuadrático de seguimiento de trayectoria por niveles de experticia.....	126
<b>Figura 87.</b> Trayectoria realizada por los diferentes niveles de experticia .....	127
<b>Figura 88.</b> Seguimiento de Set Point por nivel de experticia .....	127
<b>Figura 89.</b> Trayectoria con presencia de 2 obstáculos .....	128
<b>Figura 90.</b> Trayectoria con presencia de 3 obstáculos .....	129
<b>Figura 91.</b> Trayectoria con presencia de 4 obstáculos .....	129
<b>Figura 92.</b> Trayectorias trazadas por diferentes niveles de experticia en el Circuito 1.....	130
<b>Figura 93.</b> Error de seguimiento de trayectoria desde el obstáculo 1 al 2 en el Circuito 1 .....	131
<b>Figura 94.</b> Trayectorias trazadas por diferentes niveles de experticia en el Circuito 2.....	132
<b>Figura 95.</b> Error de seguimiento de trayectoria desde el obstáculo 2 al 3 en el Circuito 2.....	132
<b>Figura 96.</b> Trayectorias trazadas por diferentes niveles de experticia en el Circuito 3.....	133
<b>Figura 97.</b> Error de seguimiento de trayectoria desde el obstáculo 2 al 3 en el Circuito 3.....	133

<b>Figura 98.</b> Trayectoria trazada por el algoritmo de navegación autónoma en el circuito 3 .....	134
<b>Figura 99.</b> Trayectoria trazada por el tele-operador experto en el circuito 3 .....	135
<b>Figura 100.</b> Trayectoria trazada por el tele-operador intermedio en el circuito 3.....	135
<b>Figura 101.</b> Trayectoria trazada por el tele-operador novato en el circuito 3 .....	136
<b>Figura 102.</b> Tiempo medio de ejecución de los circuitos 1, 2 y 3 para diferentes niveles de experticia.....	137
<b>Figura 103.</b> Tiempo de evasión medio de obstáculos de los circuitos 1, 2 y 3 para diferentes niveles de experticia.....	139

## RESUMEN

La navegación de vehículos aéreos no tripulados en ambientes GPS-denegados representa un reto para los tele-operadores y sistemas autónomos de navegación, ya que al no contar con la localización del GPS se debe recurrir a otros sensores tales como cámara para percibir el ambiente. Las carreras de drones autónomos engloban este problema y constituyen un primer acercamiento a la navegación autónoma al ser un ambiente parcialmente controlado. En el trabajo de titulación se utilizó el dron de Parrot Bebop 2, este al ser de código abierto permite obtener las imágenes de la cámara monocular en el computador a través de una red Wi-Fi. Para desarrollar el sistema de navegación autónoma, se obtienen simultáneamente imágenes frontales y terrestres, para poder detectar y atravesar el obstáculo mientras que se sigue detectando y siguiendo la pista delimitada por marcas de superficie. Para la detección de los obstáculos se emplean filtros de color y detección de contornos, para la detección de la pista de igual manera se utilizan filtros de color en conjunto con la detección de puntos de interés. Además, se obtiene el modelo matemático del dron basado en video mediante la acumulación de movimiento. Se comparan los resultados obtenidos frente a tele-operadores de diferentes niveles de experticia y se logra mejorar los tiempos de los tele-operadores novato e intermedio en un 73.17% y 130.49% respectivamente.

### **PALABRAS CLAVE:**

- **NAVEGACIÓN AUTÓNOMA**
- **EVASIÓN DE OBSTÁCULOS**
- **SEGUIMIENTO DE TRAYECTORIA**
- **CONTROL SERVO VISUAL**

## **ABSTRACT**

Navigation of unmanned aerial vehicles in GPS-denied environments represents a challenge for tele-operators and autonomous navigation systems, since not having the GPS location, other sensors such as camera are used to perceive the environment. Autonomous drone racing encompasses this problem and constitutes a first approach to autonomous navigation, as it is a partially controlled environment. In this work the Parrot Bebop 2 drone was used, as it is open source allows obtaining the images from the monocular camera to the computer or ground station through a Wi-Fi network. To develop the autonomous navigation system, front and land images are simultaneously obtained, in order to detect and traverse the obstacle while still detecting and following the track delimited by landmarks. For obstacle detection, color filters and contour detection are used, on the other hand for trajectory tracking, color filters are used in conjunction with feature points detection. In addition, a video-based math model of the drone is obtained through the accumulation of movement and geometrical transformations. Obtained results are compared with tele-operators of different expertise's level and the beginner and intermediate tele-operators execution times are outperformed by 73.17% and 130.49% respectively.

### **KEYWORDS:**

- **AUTONOMOUS NAVIGATION**
- **OBSTACLE AVOIDANCE**
- **TRAJECTORY TRACKING**
- **VISUAL SERVOING**

## GLOSARIO

<b>GPS</b>	Global Positioning System
<b>BGR</b>	Blue Green Red
<b>FAST</b>	Features from accelerated segment test
<b>FPS</b>	Frames per second
<b>GPU</b>	Graphics processing unit
<b>HAL</b>	Hardware abstraction layer
<b>HSV</b>	Hue Saturation Value
<b>IMU</b>	Inertial measurement unit
<b>LIDAR</b>	Light Detection and Ranging
<b>MAV</b>	Micro Air Vehicle
<b>ORB</b>	Oriented FAST and rotated BRIEF
<b>PID</b>	Proportional Integral Derivative
<b>RANSAC</b>	Random sample consensus
<b>RGB</b>	Red Blue Green
<b>RMSE</b>	Root mean squared error
<b>ROS</b>	Robot operating system
<b>SIFT</b>	Scale-invariant feature transform
<b>SISO</b>	Single input, single output
<b>SURF</b>	Speeded up robust features
<b>UAV</b>	Unmanned aerial vehicle



## CAPÍTULO I

### 1. DESCRIPCIÓN DEL PROYECTO DE INVESTIGACIÓN

#### 1.1 Planteamiento del problema

Los vehículos aéreos no tripulados de micro escala todavía presentan inconvenientes con maniobras agresivas y ágiles, ambientes dinámicos, sensado erróneo y estimaciones de estado. Las carreras de drones autónomos acarrearán todos los problemas antes mencionados y presentan un primer reto al ser un ambiente parcialmente controlado.

La mayoría de los trabajos sobre navegación autónoma presentan alternativas en ambientes previamente explorados, con cámaras externas o algoritmos de aprendizaje que requieren de altas cantidades de información para tener una confianza aceptable.

Otros enfoques plantean alternativas para la navegación autónoma basada en GPS, ya que una de las principales aplicaciones de los vehículos aéreos no tripulados es la exploración y reconocimiento, sin embargo, existen ambientes GPS-denegados en los cuales esta alternativa no podría realizar una navegación precisa. Además, en este tipo de aplicaciones se depende altamente de la experiencia del tele-operador.

El presente proyecto de investigación busca solucionar la problemática de la navegación autónoma en ambientes GPS-denegados a través de sensores que no dependan de señales externas, como una cámara monocular en un ambiente parcialmente controlado.

## 1.2 Antecedentes

Los vehículos aéreos no tripulados, por sus siglas en inglés (*Unmanned Aerial Vehicle*), han tenido gran acogida en los últimos años en diferentes disciplinas, mapeo 3D (Aguilar, y otros, Visual SLAM with a RGB-D Camera on a Quadrotor UAV Using on-Board Processing, 2017), (Aguilar, y otros, Real-Time 3D Modeling with a RGB-D Camera and On-Board Processing, 2017), (Aguilar, y otros, On-Board Visual SLAM on a UGV Using a RGB-D Camera, 2017), levantamientos topográficos (Basantes, y otros, 2018), agricultura de precisión, sistemas de vigilancia, rescate (Andrea, Byron, Jorge, Inti, & Aguilar, 2018), (Pardo, Aguilar, & Toulkeridis, 2017), sistemas de navegación autónomos, etc. (Nex & Fabio, 2013; Liu & Chen Y., 2014; Zhang & Kovacs, 2012).

Como menciona Karpenko et al. (Karpenko & Konovalenko), los sistemas de navegación convencionales dependen de una Unidad de Medida Inercial (*IMU*) y de un Sistema de Posicionamiento Global (*GPS*) para cumplir con las funciones de localización (Orbea, y otros, Vertical take off and landing with fixed rotor, 2017), (Orbea, y otros, Math Model of UAV Multi Rotor Prototype with Fixed Wing Aerodynamic Structure for a Flight Simulator, 2017). No obstante, existen ambientes en los cuales no se puede obtener una señal fidedigna en el *GPS* y el sistema es susceptible a errores o fallo total. Existen sistemas de localización que utilizan únicamente la *IMU* incorporada en el sistema (Yi & Junjie, 2007), pero esta contiene ruido, y al ser procesada conlleva a una propagación exponencial del error. Si se desea obtener la posición a través de los datos obtenidos de un acelerómetro, esta señal ruidosa aumentará el error en la posición por la doble integración del error de la aceleración.

En la literatura se pueden encontrar sistemas de navegación autónoma basados en sensores externos al *UAV*. En el trabajo de Gageik et al. (Gageik & Benz, 2015) se utilizaron sensores ultrasónicos y sensores infrarrojos. Trabajos como, (Hrabar & Sukhatme, 2005; Huang & Bachrach, 2016), basan sus sistemas en diferentes tipos de cámaras como estéreo, omnidireccionales, RGB-D, etc., que no son comunes en los *UAVs* comerciales. Sin embargo, estos sensores no pueden ser añadidos en *UAVs* de micro escala por sus siglas en inglés *MAVs* (*Micro Aerial Vehicles*) que poseen hasta 100g de carga útil. La autonomía de estos está basada en imágenes obtenidas por una cámara monocular. A través de las imágenes se estima la pose y movimiento del *MAV* (*egomotion estimation*) (Galarza, Pérez, Serrano, Tapia, & Aguilar, 2018), y se realiza un control servo visual.

La evasión de obstáculos en los sistemas de navegación autónoma en tiempo real basada únicamente en imágenes monoculares ha representado un reto y uno de los problemas a enfrentar es la detección del obstáculo (Aguilar, Casaliglla, Pólit, Abad, & Ruiz, Obstacle Avoidance for Flight Safety on Unmanned Aerial Vehicles, 2017), (Aguilar, Casaliglla, & Pólit, Obstacle Avoidance for Low-Cost UAVs, 2017). Mori et al. (Mori & Scherer, 2013) en un primer enfoque presentan un método para detectar cambios relativos en el tamaño de las imágenes obtenidas por la cámara monocular, basado en los puntos de interés. Este método utiliza *SURF* (*Speeded-Up Robust Features*) que es un algoritmo basado en visión, capaz de detectar y describir estos puntos, invariante a la rotación y escala de la imagen.

Los errores de *matching* son un problema a afrontar en los detectores de puntos de interés, esto genera detecciones erróneas del obstáculo, por esta razón se desestiman valores atípicos en el modelo. Aguilar et al. (Aguilar & Angulo, Real-Time Model-Based Video Stabilization for

Microaerial Vehicles, 2015) usa una combinación de transformaciones geométricas y rechazo de falsos positivos (*outliers rejection*) basado en el Consenso de Muestreo Aleatorio por sus siglas en inglés *RANSAC* (*RANdom SAmples Consensus*), que es un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos observados que contiene valores atípicos (*outliers*), para diseñar un estabilizador de video en tiempo real para *MAVs*. Esto puede ser extrapolado a una robusta detección de obstáculos sin errores de *matching*.

Otro de los problemas que se ha afrontado es la estimación de la pose del *MAV* en ambientes sin señal GPS. Varios estudios, (Sa, Hrabar, & Corke, 2015; Soloviev & Rutkowski, 2009; Lippiello & Mebarki, 2013), utilizan un filtro de Kalman para fusionar los datos de la *IMU* y las imágenes monoculares de la cámara que tienen incorporada, de esta manera obtienen una estimación óptima de la pose actual del *MAV*.

En cuanto a carreras de drones, en la actualidad existen propuestas basadas en aprendizaje automático (Kaufmann, y otros, 2019; Kaufmann, y otros, 2018), dichas propuestas funcionan con considerables cantidades de datos para su entrenamiento y aunque tengan un desempeño superior en ambientes dinámicos, están diseñadas para un solo tipo de obstáculo.

El *Bebop 2.0* es un *MAV* comercial de la empresa *Parrot* que tiene una cámara monocular de 14 megapíxeles con una velocidad de obturación de 30 cuadros por segundo (FPS). Adicionalmente esta cámara es tipo ojo de pez (*fish-eye*), lo que significa que tiene un ángulo de visión de 180° permitiendo obtener desde imágenes frontales hasta imágenes de la superficie sobre la que se encuentra sobrevolando. Además, cuenta con un altímetro que proporciona información sobre la distancia que existe entre el *MAV* y la superficie terrestre.



**Figura 1.** Parrot Bebop 2

Fuente: (Parrot, 2019)

Los drones de *Parrot* son de código abierto, por lo que son altamente utilizados en investigación. *Parrot* presenta compatibilidad con las plataformas *Paparazzi UAV*, *Dronedcode* y *ROS (Robot Operating System)*, este último, es un conjunto de librerías y herramientas básicas para desarrollar aplicaciones con robots.

Una de estas librerías es *bebop\_autonomy*, desarrollada por Mani Monajjemi de la *Simon Fraser University* y otros colaboradores. Esta herramienta es compatible con el *Parrot Bebop 1.0* y el *Parrot Bebop 2.0* y permite la comunicación para controlar a estos *MAVs*. Es importante recalcar que *bebop\_autonomy* no provee al usuario de herramientas de control de bajo nivel para estos *MAVs*, ya que *Parrot* no ha liberado esa funcionalidad en sus productos.

### **1.3 Justificación e importancia**

Los sistemas de navegación autónoma comprenden un área de investigación extensa, existen desde sistemas basados en señales GPS, hasta sistemas basados en visión con diferentes tipos de

cámara, omnidireccionales, estéreo, RGB-D, monoculares, etc. Las aplicaciones de estos sistemas son diversas:

- En interiores (entornos controlados)
- En exteriores sin señal GPS (entornos complejos)
- Exploración de zonas desconocidas (cuevas, bosques, etc.)
- Identificación de plantaciones ilegales en zonas fronterizas
- Vigilancia privada
- Detección de eventos anormales en aglomeraciones
- Detección de plagas en plantaciones
- Agricultura de precisión
- Búsqueda y rescate de personas
- Rastreo y seguimiento de objetos
- Servicio de mensajería y entregas

Estos sistemas deben poseer un grado de autonomía elevado. La alternativa convencional es utilizar sistemas basados en GPS, esta solución es favorable cuando el área de navegación está cubierta por esta señal y de no ser así el sistema es inmune a errores de posicionamiento. La importancia del desarrollo de sistemas de navegación autónomos basados en visión es suministrar independencia de señales externas al *MAV* de forma que su navegación dependa de señales internas que en ambientes complejos posean menos posibilidad de perderse. Además, con el desarrollo de este se ponen en práctica los conceptos de la Ingeniería en Electrónica, Automatización y Control. Principalmente se utilizan conceptos de Sistemas de Control, Control Digital, Robótica Industrial y Control Inteligente.

En este proyecto de investigación se plantea el desarrollo de un sistema de navegación autónomo basado en visión para pistas de carreras de drones. Estas pistas a pesar de ser ambientes parcialmente controlados están diseñadas para probar la destreza del sistema y representan un primer reto para medir la robustez del controlador y el nivel de autonomía del *MAV*. En base a los resultados obtenidos en la investigación, se plantea extrapolar el sistema a las aplicaciones listadas en la literatura.

Adicionalmente, existen competiciones a nivel internacional de Sistemas Robóticos Autónomos. Una de ellas es la IROS (International Conference on Intelligent Robots). Esta se desarrolló el 3 de octubre de 2018 en la ciudad de Madrid – España. Existe un evento específico para carreras de drones autónomos (*Autonomous Drone Racing*), en este se presentan diferentes etapas que prueban la destreza del controlador. En la Figura 3, se muestra un esquema del circuito presentada en la IROS 2017, en este se incluyen diferentes secciones para medir la destreza del sistema:

- Sección recta
- Sección curva
- Sección giro en curva
- Sección dinámica

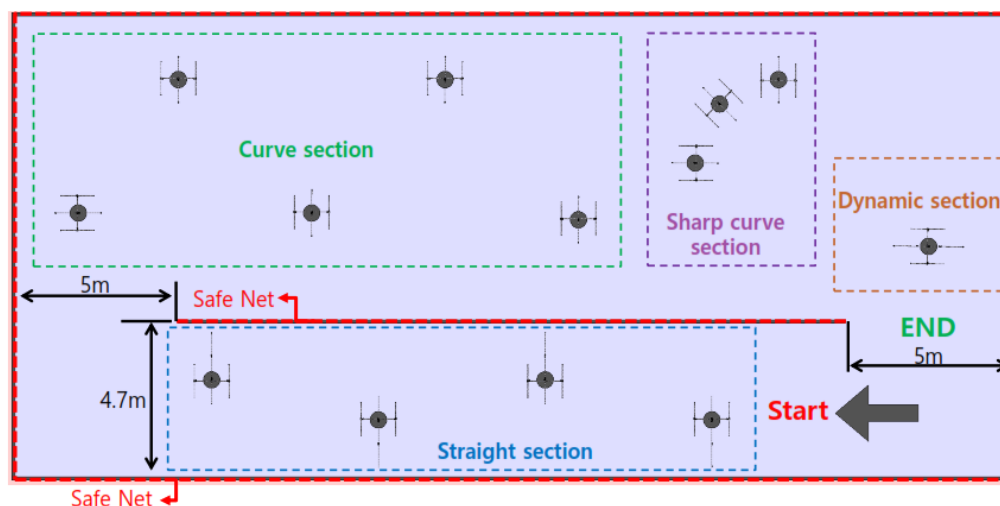
En la Figura 2 se muestran los obstáculos definidos en la IROS 2018.





**Figura 2.** Tipos de obstáculos definidos en la IROS 2018

Competition area



**Figura 3.** Esquema del circuito de la competición Autonomous Drone Racing IROS 2017

En adición, el proyecto se presenta como primer antecedente para la organización de una competición interna de sistemas robóticos autónomos a nivel superior.

## 1.4 Alcance del proyecto

En el proyecto de investigación se desarrollará un sistema de navegación autónomo basado en visión para pistas de carreras de drones con marcas de superficie (*landmarks*), tomando como referencia algoritmos de reconocimiento de puntos de interés (*feature points*) para estimar la posición de la trayectoria con respecto al UAV. Adicionalmente se diseñará e implementará un controlador servo visual (*visual servoing*) para navegar a través de los obstáculos y conjuntamente seguir la trayectoria.

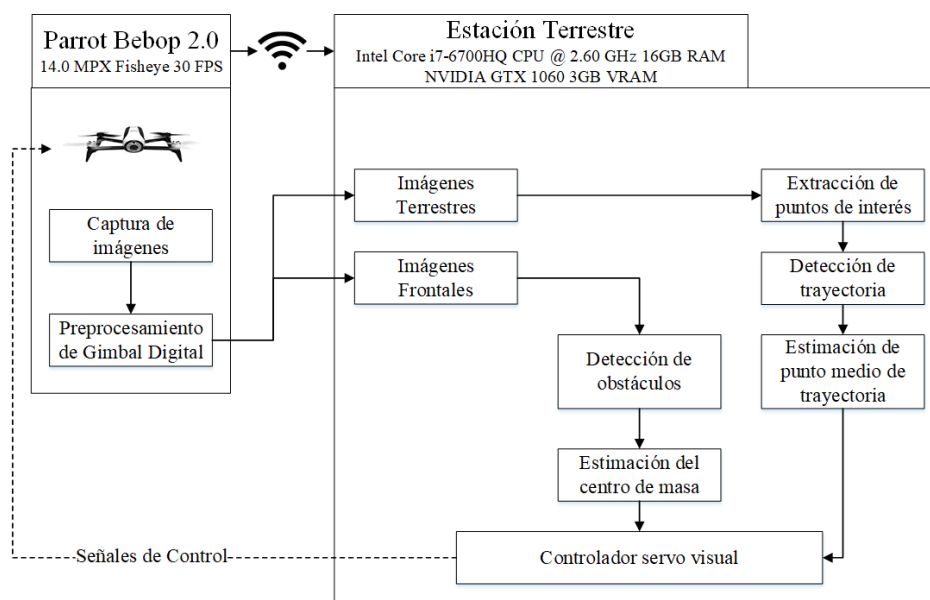
Para el circuito de la carrera, se tomarán en cuenta los obstáculos de la sección recta del esquema de la pista de carrera de la IROS 2017, con el objetivo de una futura participación en competiciones a nivel internacional a largo plazo.

El sistema estará constituido por una estación aérea Bebop 2.0 y una estación terrestre Intel Core i7-6800HQ @ 2.60 GHz con 16 GB DDR4 de memoria RAM y una tarjeta gráfica dedicada NVIDIA GTX 1060 de 3 GB de VRAM encargada de procesar las imágenes monoculares y datos inerciales, para estimar la pose del MAV y enviar las señales de control a este.

Las herramientas y software necesarios para el procesamiento de la información obtenida desde el MAV son los siguientes:

- Sistema Operativo: Ubuntu 14.04.5 LTS (Xenial Xerus)
- Robot Operating System: Distribución ROS Kinetic
- Driver bebop\_autonomy
- OpenCV y Python 3.4

Los módulos funcionales comandados por el software y herramientas listadas están organizados como se muestra en la Figura 4:



**Figura 4.** Diagrama esquemático del sistema

1. Comunicación inalámbrica con la estación terrestre: El Parrot Bebop 2.0 crea una red con SSID Bebop-XXXX, a la cual se conecta la estación terrestre.
2. Extracción de datos de la cámara (Digitalización de imágenes): ROS levanta un nodelet y a través de este publica la captura del video en el tópico /bebop/image\_raw como un mensaje de ROS de tipo sensor\_msgs/Image. Para seguir la trayectoria y evadir obstáculos al mismo tiempo se observa hacia la superficie y hacia al frente.
3. Detección de trayectoria: Se extraen puntos interés para las imágenes terrestres, determinando el punto medio para el seguimiento de la ruta marcada.
4. Detección de obstáculos: Se detectan obstáculos previamente definidos y se obtiene su centro de masa.

6. Control servo visual: Con el punto medio de la trayectoria y el centro de masa del obstáculo, se envían las señales de control hacia el MAV para navegar a través de este, siguiendo la ruta delimitada por *landmarks*.

## **1.5 Objetivos**

### **1.5.1 Objetivo General**

Diseñar un sistema de navegación autónomo basado en visión para pistas de carreras de drones delimitadas con marcas de superficie.

### **1.5.2 Objetivos Específicos**

- Realizar el estudio del sobre los tipos de algoritmos de evasión de obstáculos, seguimiento de trayectoria y control servo visual.
- Estudiar la dinámica y parámetros físicos del *MAV* Parrot Bebop 2.0
- Desarrollar el algoritmo de detección de obstáculos basado en imágenes monoculares.
- Desarrollar el algoritmo de seguimiento de trayectorias a través de marcas de superficie.
- Diseñar el controlador tanto para la evasión de los obstáculos y el seguimiento de la trayectoria.
- Realizar pruebas experimentales para el seguimiento de la trayectoria y de forma colectiva el seguimiento de la trayectoria en presencia de obstáculos.
- Evaluar el desempeño del sistema frente a diferentes niveles de experticia en circuitos de destreza de drones.

## CAPÍTULO II

### 2. MARCO TEÓRICO

#### 2.1 Vehículos aéreos no tripulados

Los UAVs por sus siglas en inglés (*Unmanned Aerial Vehicle*) o vehículos aéreos no tripulados se pueden encontrar en numerosas aplicaciones, la mayoría de estas son militares (Amaguaña, Collaguazo, Tituaña, & Aguilar, 2018), pero también existen aplicaciones civiles, aunque la mayoría de estas están orientadas a la recreación (Kendoul, 2012; Nonami, Kendoul, Suzuki, Wang, & Nakazawa, 2010). Ofrecen muchas ventajas al ser utilizados para vigilancia aérea (Jara-Olmedo A. , y otros, 2018), (Jara-Olmedo A. , Medina-Pazmiño, Tozer, Aguilar, & Pardo, 2018), reconocimiento (Aguilar W. G., y otros, Pedestrian Detection for UAVs Using Cascade Classifiers and Saliency Maps, 2017), (Aguilar W. G., y otros, 2017), (Aguilar W. G., y otros, Pedestrian Detection for UAVs Using Cascade Classifiers with Meanshift, 2017), (Aguilar W. G., y otros, 2017), (Aguilar W. G., y otros, 2017), inspección en ambientes complejos y peligrosos. La mayoría de estas aplicaciones requieren de un aterrizaje y despegue vertical, que resulta ser una de las mayores ventajas de estos sistemas. Además, están equipados con varios sensores (cámaras, acelerómetros, LIDAR, GPS, etc.) que permiten tener una percepción del entorno en el que se encuentran navegando, (Aguilar, Morales, Ruiz, & Abad, RRT\* GL Based Optimal Path Planning for Real-Time Navigation of UAVs, 2017), (Aguilar, Morales, Ruiz, & Abad, RRT\* GL Based Path Planning for Virtual Aerial Navigation, 2017), (Aguilar & Morales, 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms, 2016).

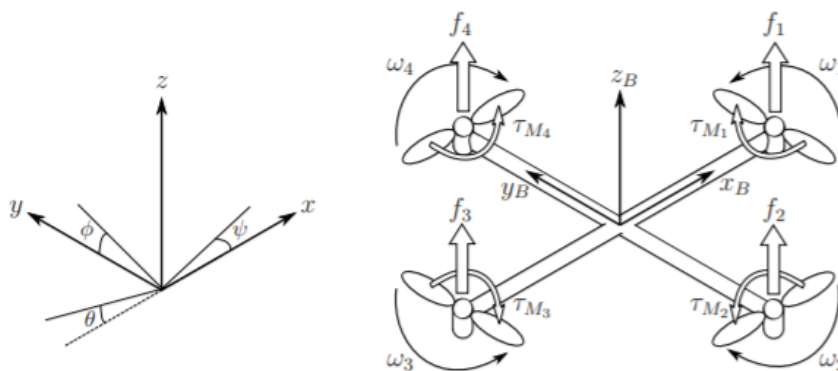
Las categorías en las que se clasifican los *UAVs* son extensas, entre ellas se encuentran: 1) por su carga útil, 2) por su uso, 3) por su diseño, 4) por su nivel de autonomía, etc. Para este trabajo de investigación se tomarán en cuenta las categorías de carga útil y diseño (Dalamagkidis, 2014; Penguin Engine, 2017; Kendoul, 2012).

La carga útil clasifica a los *UAVs* según el peso que sean capaces de llevar en un vuelo. Los vehículos aéreos de micro escala por sus siglas en inglés *MAVs*, son drones que tienen una carga útil menor a 100g y su peso total bordea los cientos de gramos.

Los *MAVs* se pueden recategorizar según su diseño, estos pueden ser de ala fija o basados en rotores, en este proyecto de investigación se analizarán los *MAVs* con cuatro rotores, también llamados cuadricópteros.

### 2.1.1 Dinámica del movimiento de un cuadricóptero

La estructura de un cuadricóptero se presenta en la Figura 5, en esta se pueden apreciar las velocidades angulares, torques y fuerzas creadas por los rotores.



**Figura 5.** Eje de referencia y eje del cuerpo de un cuadricóptero

Fuente: (Luukkonen, 2011)

La posición lineal absoluta del cuadricóptero está definida en el eje de referencia en los ejes  $x$ ,  $y$ ,  $z$  con  $\xi$ . La pose, es decir, la posición angular, en el eje de referencia está definida por los tres ángulos de Euler  $\eta$ . El ángulo *pitch*  $\theta$  determina la rotación del cuadricóptero alrededor del eje  $y$ . El ángulo *roll*  $\phi$  determina la rotación alrededor del eje  $x$  y el ángulo *yaw*  $\psi$  determina la rotación alrededor del eje  $z$ . El vector  $q$  representa entonces, la posición del cuadricóptero en referencia al eje referencial.

$$\xi = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \eta = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad q = \begin{bmatrix} \xi \\ \eta \end{bmatrix} \quad (1)$$

Se asume que el cuadricóptero tiene una estructura simétrica, es decir, que los cuatro brazos están alineados en referencia al eje  $x$  y al eje  $y$ . Por lo tanto, la matriz inercial del cuadricóptero corresponde a una matriz diagonal  $I$  en la cual  $I_{xx} = I_{yy}$ .

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (2)$$

La velocidad angular del rotor  $i$ , está denotada por  $\omega_i$ , esta crea una fuerza  $f_i$  en la dirección del eje del rotor. En adición, la velocidad angular y la aceleración del rotor crean un torque  $\tau_{M_i}$  alrededor del eje del rotor.

$$f_i = k\omega_i^2, \quad \tau_{M_i} = b\omega_i^2 + I_M\dot{\omega}_i, \quad (3)$$

En la cual la constante de elevación es  $k$ , la constante de arrastre es  $b$  y el momento de inercia del motor corresponde a  $I_M$ . Usualmente el efecto de  $\dot{\omega}_i$  es muy pequeño y por lo tanto se lo puede omitir.



La combinación de las fuerzas crea el empuje  $T$  en la dirección del eje del cuerpo del cuadricóptero. Por lo tanto, la rotación en roll se adquiere disminuyendo la velocidad de rotación del rotor 2 e incrementando la velocidad de rotación del rotor 4. De igual forma, la rotación en pitch se adquiere disminuyendo la velocidad de rotación del rotor 1 e incrementando la velocidad del rotor 3. La rotación en yaw se adquiere incrementando las velocidades de dos rotores opuestos y disminuyendo las velocidades de los otros dos. En la Tabla 1 se resumen los movimientos que puede adquirir el UAV (Salcedo, 2018), (Aguilar, Salcedo, Sandoval, & Cobeña, 2017), en donde  $\omega_0$  es la velocidad de los rotores en las cuales las fuerzas  $f_i$  y los torques  $\tau_{M_i}$  quedan anulados y se produce el fenómeno en el cual el cuadricóptero está suspendido en el aire y permanece inmóvil.

**Tabla 1**

*Movimientos de un cuadricóptero según la velocidad de los rotores*

Rotación	Rotor 1	Rotor 2	Rotor 3	Rotor 4	Desplazamiento
Yaw horario	$\omega_0 - \Delta$	$\omega_0 + \Delta$	$\omega_0 - \Delta$	$\omega_0 + \Delta$	Ninguno
Yaw anti horario	$\omega_0 + \Delta$	$\omega_0 - \Delta$	$\omega_0 + \Delta$	$\omega_0 - \Delta$	Ninguno
Roll derecha	$\omega_0 - \Delta$	$\omega_0 - \Delta$	$\omega_0$	$\omega_0$	Derecha (Positivo)
Roll izquierda	$\omega_0$	$\omega_0$	$\omega_0 - \Delta$	$\omega_0 - \Delta$	Izquierda (Negativo)
Pitch adelante	$\omega_0 - \Delta$	$\omega_0$	$\omega_0$	$\omega_0 - \Delta$	Adelante (Negativo)
Pitch atrás	$\omega_0$	$\omega_0 - \Delta$	$\omega_0 - \Delta$	$\omega_0$	Atrás (Negativo)
Ninguna	$\omega_0 + \Delta$	$\omega_0 + \Delta$	$\omega_0 + \Delta$	$\omega_0 + \Delta$	Arriba (Positivo)
Ninguna	$\omega_0 - \Delta$	$\omega_0 - \Delta$	$\omega_0 - \Delta$	$\omega_0 - \Delta$	Abajo (Negativo)

### 2.1.2 Control de UAVs

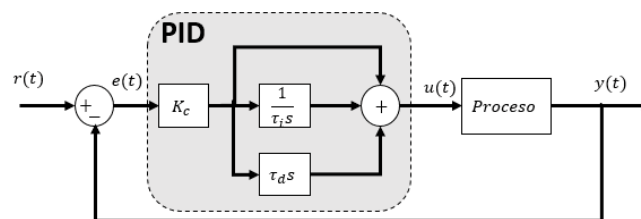
Los UAVs pertenecen a la clase de sistemas mecánicos sub-actuados, es decir, que el número de actuadores sobre los cuales se puede tomar una acción de control, es menor al número de salidas

que presenta el sistema, para el caso de un cuadricóptero se tiene cuatro actuadores, pero se pueden obtener seis salidas que corresponden a posición y orientación en el espacio. El diseño del controlador para estos sistemas generalmente se basa en su modelo dinámico, que presenta características altamente no lineales, por lo que el modelamiento del sistema constituye una parte fundamental en el control del UAV.

Existen tres tipos de control para *UAVs* según (Kendoul, 2012):

- Controladores de vuelo basados en aprendizaje
- Controladores de vuelo lineales
- Controladores no lineales basados en modelos

Si bien los métodos de aprendizaje han tenido gran acogida en los últimos años (Aguilar, Álvarez, Grijalva, & Rojas, 2019), no son utilizados en aplicaciones comerciales, por el contrario, el control lineal se prefiere en la mayoría de aplicaciones, específicamente se utiliza comúnmente el controlador *PID* (Aguilar, Angulo, & Costa-Castello, Autonomous Navigation Control for Quadrotors in Trajectories Tracking, 2017), (Aguilar, Manosalvas, Guillén, & Collaguazo, 2018) aunque el modelo del sistema se pueda obviar y las ganancias del controlador se sintonicen manualmente. En la Figura 6 se muestra el esquema de control *PID*.



**Figura 6.** Esquema de control PID

### 2.1.3 Estimación de estado con Filtro de Kalman

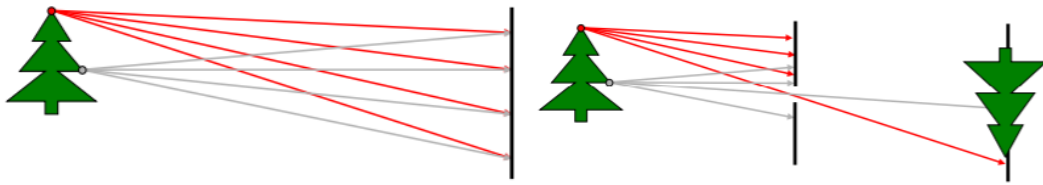
Uno de los mayores desafíos en robótica es proveer de información precisa sobre variables que presentan incertidumbre. Por ejemplo, en las señales *GPS*, las mediciones dependen de muchos factores externos como, condiciones térmicas, efectos atmosféricos, pequeños cambios en la posición del satélite, etc.

El Filtro de Kalman (Kalman, 1960) es uno de los algoritmos de estimación de estados más utilizados. El filtro de Kalman produce la estimación de variables basada en mediciones imprecisas e inciertas. Además, el filtro de Kalman provee una predicción de estados futuros del sistema, basada en estimaciones pasadas (Becker, 2018).

## 2.2 Formación de imágenes

Las imágenes no pueden existir sin luz. Para producir una imagen, la escena a reproducir debe estar iluminada. Existen dos tipos de fuentes de luz, puntuales o difusas. La luz puntual se origina en un punto reducido del espacio, además tiene una intensidad y un espectro de colores, es decir una distribución sobre las longitudes de onda. Las fuentes de luz difusas son más complicadas de modelar y no son usadas a menudo.

Cuando la luz *golpea* a un objeto, esta se refleja y se dispersa. La forma más básica para captar la luz que un objeto 3D refleja correspondería a una superficie plana en frente del objeto, pero esto causa que la imagen se distorsione debido a que la luz se dispersa en varias direcciones. Para solucionar esta distorsión se añade una apertura en la superficie plana, de manera que se bloquean la mayoría de los rayos incidentes en la superficie, este modelo se conoce como *pin hole* y es uno de los modelos más utilizados en visión por computadora.

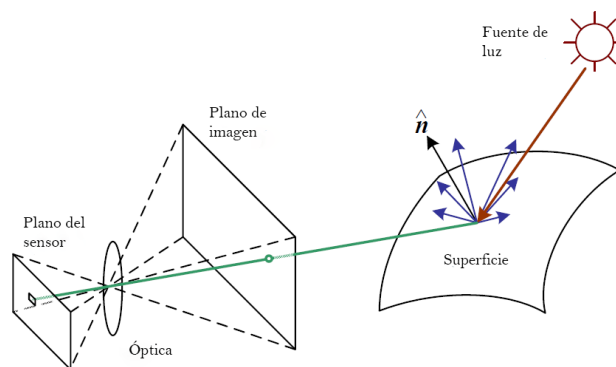


**Figura 7.** Diferencia de incidencia de rayos en la barrera con y sin una apertura

Los rayos que pasan a través de la apertura forman una imagen invertida (Figura 7) y de diferentes dimensiones, que están en función de la distancia focal  $f$  la cual es distancia entre la barrera y la película donde se refleja la imagen.

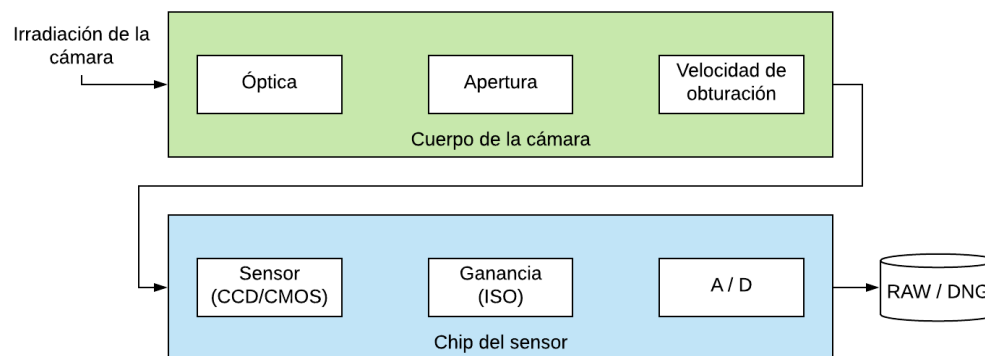
En el modelo *pin hole* existe un inconveniente ya que el diámetro de la apertura puede producir distorsiones en cuanto a las variaciones de longitud de onda de los rayos que inciden en la misma. Para solucionar esto se incluyen lentes ópticos que permiten que más luz de cada punto en la escena se refleje sobre la imagen (Bedros, 2017).

El principio de formación de una imagen digital sigue la misma línea que se ha descrito en los párrafos anteriores, pero en el proceso de formación de esta se sustituye a la película fina en el plano de la imagen donde se reflejaba la imagen invertida por un sensor (Figura 8).



**Figura 8.** Esquema de proceso de formación de imagen en una cámara digital

La luz que incide sobre el sensor es recogida por un área de sensado activo, integrada por la duración de la exposición. Convencionalmente se utilizan dos tipos de sensores en las cámaras modernas, los sensores *CCD* por sus siglas en inglés (*Charge-coupled device*) y los sensores *CMOS* (*Complementary metal oxide on silicon*). Una vez que el sensor ha capturado las intensidades se pasa esta señal analógica a un conversor A/D y se produce la imagen *RAW*, esto se resume en la Figura 9.



**Figura 9.** Proceso de formación de una imagen digital, sin post procesamiento

Tradicionalmente, los sensores *CCD* superan a los sensores *CMOS* en aplicaciones que requieren calidad, mientras que los sensores *CMOS* se usan en aplicaciones de baja potencia (Szeliski, 2010).

### 2.3 Caracterización de imágenes

Para realizar tareas de procesamiento de imágenes en aplicaciones de visión por computadora, se necesita representar a la imagen por sus características, ya que la imagen completa está llena de información que puede ser procesada por el hombre mas no por un computador. Existen dos tipos de descripción de características de imágenes (Hassaballah, Abdelmgeid, & Alshazly, 2016):

- Características globales: En el cual se extrae un solo vector con características correspondientes a todas las regiones de la imagen como una sola.
- Características locales: En el cual se extraen varios vectores con la información característica de cada región.

Si se comparan los dos tipos de caracterización de imágenes, existe una desventaja de las características globales y es que no son invariantes a una significativa cantidad de transformación y son sensibles al ruido y oclusión.

### **2.3.1 Detección de características**

Un punto característico o de interés puede ser definido como un punto de una imagen que se distingue de sus vecinos inmediatos, por lo tanto, su propósito es proveer una representación que permita realizar un emparejamiento de características entre imágenes eficiente (Hassaballah, Abdelmgeid, & Alshazly, 2016), (Aguilar & Angulo, Real-time video stabilization without phantom movements for micro aerial vehicles, 2014), (Aguilar & Angulo, Robust video stabilization based on motion intention for low-cost micro aerial vehicles, 2014), (Aguilar, Angulo, & Pardo, Motion intention optimization for multicopter robust video stabilization, 2017).

Las siguientes características son importantes al momento de utilizar un algoritmo de detección de características o puntos de interés en aplicaciones de visión por computadora:

- Robustez
- Repetibilidad
- Precisión
- Generalidad

- Eficiencia
- Cantidad

A continuación, se describe el proceso de detección de puntos de interés mas no el de descripción, debido a que en el presente proyecto de titulación no se hace uso de los descriptores obtenidos con los algoritmos.

### 2.3.1.1 SIFT

Es un método de detección y descripción de puntos de interés invariantes a la orientación y escala desarrollado por David G. Lowe (Lowe, 2004). La primera etapa de computación busca sobre todas las escalas y las ubicaciones de la imagen. Se implementa eficientemente utilizando la función *DoG* (Diferencia de gaussianas) para identificar potenciales puntos de interés que sean invariantes a la escala y orientación.

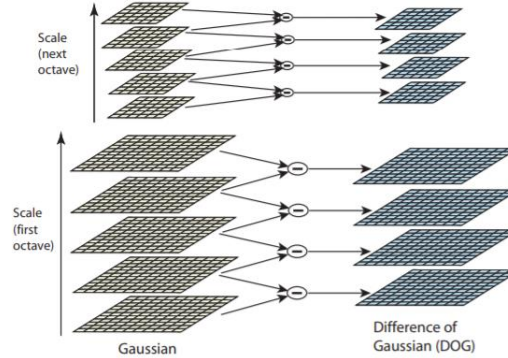
En este caso se va a entender como escala al valor de  $\sigma$ , en la función:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (4)$$

Cada escala que se encuentra una octava arriba de la otra será una imagen suavizada de la anterior y viceversa, de esta manera tomando la diferencia de gaussianas de la ecuación ( 5):

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (5)$$

Se pueden obtener puntos de interés invariantes a la escala. El proceso descrito se resume en la Figura 10.



**Figura 10.** Diferencia de Gaussianas adyacentes para espacios de escala diferentes

Fuente: (Lowe, 2004)

### 2.3.1.2 SURF

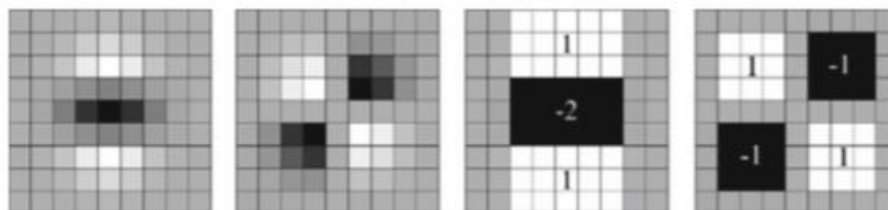
El algoritmo *SURF* por sus siglas en inglés (*Speeded Up Robust Features*) para la detección de puntos de interés desarrollado por (Bay, Tuytelaars, & Van Gool, 2006) como una alternativa eficiente a *SIFT*, al contrario de este, es mucho más rápido y robusto. Para la detección de puntos de interés, en lugar de apoyarse en derivadas ideales de Gaussianas, el cálculo se basa en simples *cajas 2D* como filtros; en donde, se usa un detector *blob* invariante a la escala basado en el determinante de la matriz Hessiana para la selección de la escala y la ubicación de los puntos de interés.

La idea básica es aproximar de manera eficiente la segunda derivada de una función Gaussiana con la ayuda de imágenes integrales usando un conjunto de filtros. Las *cajas* de  $9 \times 9$  de la Figura 11, son aproximaciones de una Gaussiana con  $\sigma = 1.2$  y representan la escala más baja para calcular los mapas de respuesta *blob*. Estas aproximaciones están denotadas por  $D_{xx}$ ,  $D_{yy}$  y  $D_{xy}$ . Por lo tanto, El determinante aproximado de la matriz Hessiana se puede representar por:

$$\det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \quad (6)$$



En donde,  $w$  es el peso relativo para la respuesta del filtro y es usado para balancear la expresión para el determinante Hessiano.



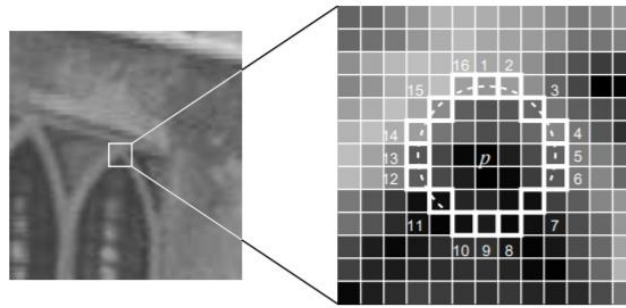
**Figura 11.** De izquierda a derecha: Derivadas Gaussianas de segundo orden en  $y$  ( $D_{yy}$ ), dirección  $xy$  ( $D_{xy}$ ) y su aproximación en la misma dirección, respectivamente

Fuente: (Bay, Tuytelaars, & Van Gool, 2006)

Esta aproximación representa la respuesta *blob* de la imagen. Estas respuestas son almacenadas en un mapa de respuesta *blob* y el máximo local es detectado y refinado usando una interpolación cuadrática, de la misma manera que con *DoG*. Finalmente, se realiza una supresión no máxima en una región de  $3 \times 3 \times 3$  para obtener puntos de interés estables.

### 2.3.1.3 FAST

El algoritmo FAST por sus siglas en inglés (*Features from Accelerated Segment Test*) propuesto en (Rosten & Drummond, 2006) es un detector de puntos de interés. Un punto de interés es un pixel que tiene una posición definida y se puede detectar robustamente. Los puntos de interés tienen una alta información local y deben ser idealmente repetibles entre diferentes imágenes. La idea de FAST surge de utilizar la detección de puntos de interés al tiempo real para aplicaciones de visión por computador.



**Figura 12.** Punto de interés bajo test y los 16 pixeles adyacentes para el algoritmo FAST  
Fuente: (Rosten & Drummond, 2006)

El algoritmo FAST se detalla a continuación:

1. Seleccionar un pixel  $p$  en la imagen. Asumir que la intensidad de este pixel corresponde a  $I_p$ . Este es el pixel que va a ser definido o no como un punto de interés.
2. Definir una intensidad umbral  $T$ .
3. Considerar un círculo de 16 pixeles rodeando al pixel  $p$ .
4.  $N$  pixeles adyacentes fuera de los 16 pixeles necesitan estar arriba o abajo del valor umbral  $I_p$  por el valor de  $T$ . Los autores usaron el valor de 12 en la primera versión del algoritmo.
5. Para acelerar el algoritmo, primero se debe comparar las intensidades de los pixeles 1, 5, 9 y 13 del círculo con  $I_p$ .
6. Si por lo menos tres de los cuatro valores de los pixeles  $-I_1, I_5, I_9, I_{13}$ , no están debajo o arriba de  $I_p + T$ , entonces  $p$ , no es un punto de interés. En este caso rechazar  $p$  como un posible punto de interés. Mas si al menos tres de los cuatro pixeles están arriba o debajo de  $I_p + T$ , entonces verificar para todos los 16 pixeles y asegurar que los 12 pixeles contiguos cumplan con el criterio.
7. Repetir este procedimiento para todos los pixeles en la imagen.

### 2.3.1.4 ORB

ORB (*Oriented FAST and Rotated BRIEF*) fue desarrollado por (Rublee, Rabaud, Konolige, & Bradski, 2011) en *OpenCV Labs* como una alternativa eficiente y viable a los algoritmos ya patentados SURF y SIFT, cabe mencionar que ORB es un algoritmo de libre uso.

Como se mencionó en el anterior apartado, el algoritmo FAST es de bajo costo computacional, desafortunadamente, no cuenta con orientación y características multiescala. Para resolver esto, ORB propone una pirámide de imágenes multiescala. La pirámide de imágenes multiescala consiste de una secuencia de imágenes en la que todas ellas son una representación de cada una a diferentes resoluciones. Cada nivel de la pirámide es una representación escalada de su imagen previa. Una vez que ORB ha creado la pirámide este usa el algoritmo FAST para detectar puntos de interés en la imagen. Seguido de esto, se descarta algunos puntos de interés con una medición de esquinas de Harris (Harris & Stephens, 1988), ya que el algoritmo FAST tiende a producir muchos puntos redundantes en bordes.

Después de localizar los puntos de interés ORB asigna una orientación a cada punto dependiendo de cómo los niveles de intensidad cambian alrededor de dicho punto. Para detectar el cambio de intensidad se utiliza el centroide de intensidad, este asume que la intensidad tiene un *offset* de su centro, y este vector se usa para calcular una orientación.

En primer lugar, se calculan los momentos definidos como:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \quad (7)$$

Con estos momentos, se puede calcular el centroide o el centro de masa como:

$$C = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (8)$$

De esta manera se puede construir un vector del centro del punto de interés  $O$  al centroide –  $OC$ . Entonces, la orientación del punto de interés está dada por:

$$\theta = \text{atan 2}(m_{01}, m_{10}) \quad (9)$$

## 2.4 Procesamiento de imágenes

El procesamiento de imágenes hace referencia a la técnica utilizada por los humanos para poder obtener información relevante de una imagen como características, puntos de interés, colores, reconocimiento de objetos, rastreo de objetos, etc.

Los humanos son seres principalmente visuales (Russ, 2011), la percepción del espacio y su entorno la realiza a través de la vista. Naturalmente todos los resultados que se producen a través de un determinado proceso en cualquier área del desarrollo de su vida son productos visuales, ya que la capacidad de interpretación de los humanos a través de una imagen es mucho mayor a la de productos auditivos o tangibles.

En las últimas décadas la idea de dotar a robots del sentido primario del ser humano ha sido ampliamente usada, esto ha sido posible gracias a dos condiciones:

- Crecimiento exponencial de la capacidad de procesamiento de las unidades de procesamiento gráfico (*GPU*)
- Desarrollo de nuevas tecnologías en cámaras

Sin las condiciones mencionadas, el procesamiento de imágenes en tiempo real para aplicaciones de robótica no podría ser llevado a cabo, teniendo que buscar otras alternativas que se ajusten a la tecnología existente.

### **2.4.1 Modelo de color**

Es una representación matemática abstracta de los colores, a diferencia de un espacio de color que es una organización específica de los colores, lo que sugiere que se pueden tener varios espacios de colores para un modelo de color (Skrede, 2017); entre estos los más usados según su aplicación son:

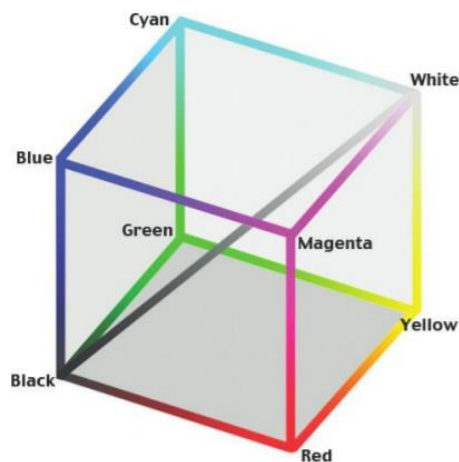
- *RGB*: Visualización en pantallas
- *HSV*: Selección intuitiva de colores
- *CMYK*: Impresoras a color

#### **2.4.1.1 Modelo de color *RGB***

Es un modelo de color empleado en todos los dispositivos que muestran imágenes: televisores, monitores, proyectores, etc. Este se basa en la teoría de Young-Helmholtze propuesta en el siglo XIX que tiene como sustento que las longitudes de onda de los tres colores primarios en este modelo de color (Rojo, Verde y Azul) corresponden a la sensibilidad de cada uno de los conos sensitivos del ojo humano (Alonso, 2009) que son de aproximadamente 559 nm, 531 nm y 419 nm, respectivamente.

Este modelo hace referencia al sistema de coordenadas cartesianas, en el cuál cada eje corresponde a la intensidad de un color primario, los vértices opuestos al origen corresponden a los colores secundarios cian, magenta y amarillo, específicamente en el vértice en donde las

intensidades de rojo y azul son máximas se encuentra el color magenta, de la misma manera, el vértice en donde los colores azul y verde tienen intensidad máxima se encuentra el color cian y finalmente para el vértice en donde el color rojo y verde son máximos se encuentra el color amarillo. Estos colores secundarios tienen como origen el color blanco que es opuesto al origen de los colores primarios, el negro, y en cuya diagonal se encuentra la escala de grises (Ver Figura 13).



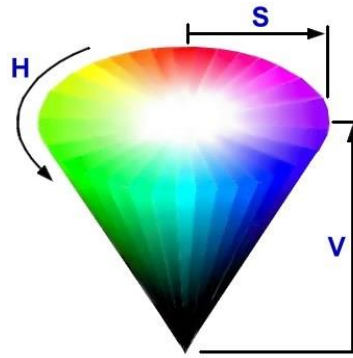
**Figura 13.** Representación del modelo de color RGB

#### 2.4.1.2 Modelo de color *HSV*

En el modelo de color *HSV* se distinguen los colores de una forma más intuitiva, el color está definido por:

- **Matiz:** Es la cantidad que define la tonalidad del color, está definido por el ángulo de  $0^\circ$  a  $360^\circ$
- **Saturación:** Se refiere a la cantidad de luz blanca presente en un color, define la temperatura del color.
- **Valor (Intensidad):** Representa la iluminación del color, permite tener la sensación de un color con más o menos luz reflejada.

El modelo es una transformación no lineal del tetraedro RGB, y a diferencia de este está representado en coordenadas cilíndricas mas no en cartesianas.



*Figura 14. Modelo de color HSV*

#### 2.4.2 Suavizado de imágenes

Cuando los colores que se muestran en una imagen no tienen una distribución uniforme es posible que esta se haya transmitido con ruido, esto se puede deber a varias causas:

- Malas detecciones de la cámara
- Introducción de ruido en los circuitos electrónicos
- Interferencias electromagnéticas en la transmisión de la imagen

La relación señal/ruido es un indicador de la presencia de ruido en la imagen, cuando esta relación es baja, la cantidad de ruido presente puede pasar desapercibida por el ser humano. Cuando se trata de procesamiento de imágenes es ideal tener imágenes sin ruido, para esto se pueden aplicar filtros lineales de la forma:

$$g(i, j) = \sum_{k, l} f(i + k, j + l) * h(k, l) \quad (10)$$

En donde:

- $g(i, j)$  es el pixel de salida
- $f(i + k, j + l)$  es la suma ponderada de los valores del pixel de entrada
- $h(k, l)$  llamado kernel, hace referencia a los coeficientes del filtro

#### 2.4.2.1 Filtro de promedio

Es un filtro en el que cada pixel de salida es la media de sus “vecinos” o “píxeles cercanos” definidos por el *kernel*. Este se define de la siguiente manera:

$$K = \frac{1}{K_{ancho} \cdot K_{altura}} \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix} \quad (11)$$

Este tipo de filtro corresponde a la forma más sencilla de filtros de suavizado, con la desventaja que no es robusto a la introducción de ruido considerado elevado, esto se debe a que al tratarse de la media de una región de la imagen esta medida estadística tiende a ser desviada (*biased*) por el ruido.

#### 2.4.2.2 Filtro Gaussiano

También conocido como *Gaussian Blur*, es uno de los filtros de suavizado más utilizado. El filtrado de la imagen se la realiza como una convolución de cada pixel con un *kernel* Gaussiano definido de la siguiente forma:

$$G_{(x,y)} = Ae^{\left[ \frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2} \right]} \quad (12)$$

en donde:

- $\mu$  es la media para cada variable  $(x,y)$
- $\sigma$  es la desviación estándar para cada variable  $(x,y)$



### 2.4.2.3 Filtro de Media

El filtro de media corresponde a un filtro no lineal que a diferencia de los filtros antes mencionados no se desviado por la media establecida por el tamaño del kernel. Este filtro recorre por cada pixel de la imagen y cambia su valor por el valor de la media de sus píxeles vecinos.

### 2.4.3 Transformaciones morfológicas

Mientras que los filtros no lineales son usados para mejorar la calidad de imágenes a color o en escala de grises, existe otra aplicación para estos. La más común es la transformación morfológica de una imagen binaria correspondiente a una imagen después de una operación de *thresholding*, denotada por:

$$\theta(f, t) = \begin{cases} 1 & \text{si } f \geq t, \\ 0 & \text{cc} \end{cases} \quad (13)$$

En donde:

- $f$  corresponde a cada pixel de la imagen de entrada
- $t$  corresponde al valor umbral de la intensidad del pixel

Para realizar la transformación morfológica primero se convoluciona la imagen binaria con un elemento de estructura binaria y luego se selecciona el valor de salida dependiendo del valor umbral seleccionado para la salida (Szeliski, 2010), la convolución está denotada por:

$$c(i, j) = \sum_{k,l} f(i + k, j + l) * s(k, l) \quad (14)$$

en donde:

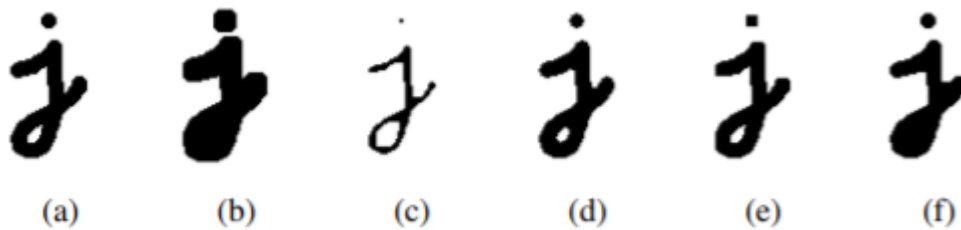
- $c$  es el valor de salida

- $f$  es la imagen resultante de la operación de *thresholding*
- $s$  es el elemento de estructura binaria

En la mayoría de los casos  $s$  corresponde a un *kernel* cuadrado de  $N \times N$  elementos. Las transformaciones morfológicas estándar corresponden a:

- Dilatación:  $dilate(f, s) = \theta(c, 1)$
- Erosión:  $erode(f, s) = \theta(c, S)$
- Mayoría:  $maj(f, s) = \theta(c, S/2)$
- Apertura:  $opening(f, s) = dilate(erode(f, s), s)$
- Cierre:  $closing(f, s) = erode(dilate(f, s), s)$

En la Figura 15 se muestran las transformaciones morfológicas para un *kernel* de  $3 \times 3$  elementos.

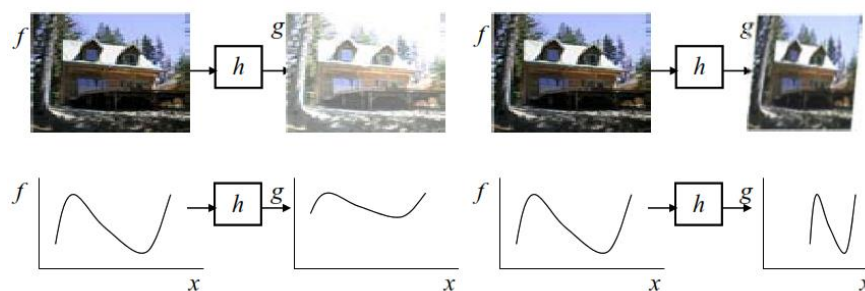


**Figura 15.** Transformaciones morfológicas (Szeliski, 2010)

#### 2.4.4 Transformaciones geométricas

Una transformación geométrica es una función que mapea todos los puntos de una imagen  $P$  a otro dominio  $P'$ , esto se expresa con la siguiente ecuación:

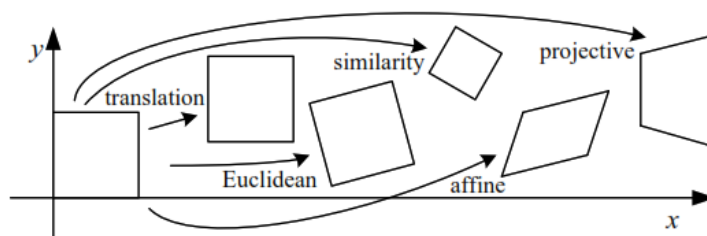
$$g(x) = f(h(x)) \quad (15)$$



**Figura 16.** Deformaciones de imágenes causadas por la modificación del dominio

Fuente: (Szeliski, 2010)

Las transformaciones geométricas en el plano más simples se muestran en la Figura 17:



**Figura 17.** Transformaciones geométricas para el plano bidimensional

#### 2.4.4.1 Traslación

Corresponde a una traslación bidimensional en el plano que se puede expresar como  $x' = x + t$ , o en su forma matricial como:

$$x' = [I \quad t]x \quad (16)$$

Donde  $I$  corresponde a una matriz identidad de dimensiones  $2 \times 2$  y  $t$  al vector de traslación de dimensiones  $2 \times 1$ , esta forma aunque es compacta no es útil al momento de realizar operaciones matriciales como la multiplicación, es por eso que se usan coordenadas homogéneas para poder representar las transformaciones en el plano, entonces adjuntando una matriz de forma  $[0^T \quad 1]$ ,

donde  $0^T$  corresponde a un vector de ceros de dimensiones  $1 \times 2$  y uno corresponde a la variable  $z$  constante, resultando en:

$$x' = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} x \quad (17)$$

#### 2.4.4.2 Transformación de cuerpo rígido

También conocida como transformación euclidiana, ya que las distancias euclidianas se conservan entre puntos correspondientes, se escribe de la forma:  $x' = Rx + t$  o en su forma matricial

$$x' = [R \quad t] x \quad (18)$$

Donde

$$R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (19)$$

corresponde a una matriz de rotación ortonormal que cumple con  $RR^T = 1$  y  $|R| = 1$

#### 2.4.4.3 Transformación de similitud

También conocida como rotación escalada, esta transformación puede ser expresada de la forma  $x' = sRx + t$  donde  $s$  corresponde a un factor de escala arbitrario. Se puede representar en forma matricial como:

$$x' = [sR \quad t]x = \begin{bmatrix} s \cos \theta & -s \sin \theta \\ s \sin \theta & s \cos \theta \end{bmatrix} x + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} x \quad (20)$$

Donde no se requiere que se cumpla con  $a^2 + b^2 = 1$ , es decir que las escalas pueden ser diferentes. La transformación de similitud preserva los ángulos entre líneas.

#### 2.4.4.4 Transformación afín

Una transformación afín, o también conocida como transformación de afinidad es una transformación lineal no singular seguida de una traslación. Su representación matricial es:

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

O en bloque:

$$x' = H_A x = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} x \quad (22)$$

Donde  $A$  es una matriz singular de dimensiones  $2 \times 2$ . Una transformación afín plana o de dos dimensiones tiene seis grados de libertad correspondiendo a los seis elementos de la matriz  $H_A$ . La transformación se puede calcular con tres puntos correspondientes entre imágenes (Hartley & Zisserman, 2003).


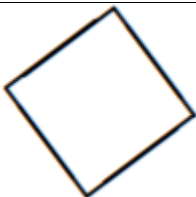
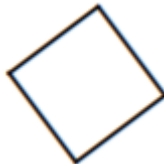


#### 2.4.4.5 Transformación proyectiva

Esta transformación también es conocida como homografía o transformación de perspectiva, trabaja con coordenadas homogéneas, además es una transformación no singular lineal, se representa de la forma

$$x' = \tilde{H} x \quad (23)$$

Donde  $\tilde{H}$  es una matriz arbitraria de dimensiones  $3 \times 3$ . La transformación de perspectiva preserva líneas rectas, es decir, permanecen rectas después de la transformación.

**Tabla 2***Transformaciones geométricas en coordenadas de dos dimensiones*

<b>Transformación</b>	<b>Matriz</b>	<b>Grados de libertad</b>	<b>Preserva</b>	<b>Representación</b>
Traslación	$[I   t]_{2 \times 3}$	2	Orientación	
Euclidiana	$[R   t]_{2 \times 3}$	3	Tamaño	
Similitud	$[sR   t]_{2 \times 3}$	4	Ángulos	
Afín	$[A]_{2 \times 3}$	6	Paralelismo	
Proyectiva	$[\tilde{H}]_{3 \times 3}$	8	Líneas rectas	

Fuente: (Szeliski, 2010)

## CAPÍTULO III

### 3. DESCRIPCIÓN DEL SISTEMA

#### 3.1 Descripción del hardware del sistema

El sistema consta de dos unidades de hardware, la unidad o estación terrestre que se denominará *GS (Ground Station)* y la unidad aérea que se denominará *UAV*. Estas se comunican a través de un enlace Wi-Fi que es generado por el *UAV*.

##### 3.1.1 Estación terrestre

La estación terrestre la conforma un ordenador con procesador Core i7 6700HQ con memoria RAM de 16 GB DDR4 y una unidad de procesamiento gráfico NVIDIA GeForce GTX 1060.

##### 3.1.2 Estación aérea (UAV)

El *UAV* corresponde a un vehículo aéreo no tripulado *Bebop 2* de la marca francesa *Parrot*, es un dron recreativo y de código abierto con un peso de 500 gramos y una autonomía de vuelo de 25 minutos entrando en la categoría de *MAV* según (Kendoul, 2012). Aunque este dron fue diseñado para ser recreativo, al ser de código abierto ha llamado la atención de los desarrolladores, llegando a ser así uno de los *MAVs* más utilizados para desarrollo de algoritmos o incluso para participar en carreras de drones autónomos a nivel internacional, un ejemplo de esto es la participación del equipo *QuetzalC++* del *INAOE* de México en la *IROS 2017*, llegando a obtener el primer lugar en dicha competición.



**Figura 18.** Parrot Bebop 2 vista frontal

Fuente: (Parrot, 2019)

### 3.1.2.1 Características generales

El *Bebop 2* está diseñado para velar por la seguridad, y aunque no disponga de protectores de hélices como su antecesor el *Bebop 1*, cuenta con hélices flexibles que se bloquean en caso de contacto o colisión. Además, cuenta con un *LED* que le permite ser visualizado y localizado a grandes distancias. A continuación, en la Tabla 3 se detallan las características del sistema interno, construcción y conectividad del *Bebop 2*.

**Tabla 3**

*Características del sistema interno, construcción y de conectividad del Bebop 2*

<b>Sistema Interno</b>	
<b>Sensores</b>	GPS integrado, IMU
<b>CPU</b>	Dual core con GPU quad-core
<b>Almacenamiento</b>	8 GB de almacenamiento flash
<b>Construcción</b>	
<b>Motores</b>	4 motores sin escobillas Outrunner
<b>Estructura</b>	PA12 estructura reforzada de fibra de vidrio (20%) y Grilamid (casco)
<b>Peso</b>	500 gramos, batería: 192 gramos
<b>Dimensiones</b>	38 x 33 x 9 cm

CONTINÚA



### Conectividad

<b>Estándar</b>	Wi-Fi IEEE 802.11 a /b /g /n /ac
<b>Antena Wi-Fi</b>	MIMO de banda dual con 2 antenas dipolo dual de 2.4 y 5 GHz
<b>Alcance de la señal</b>	300m con dispositivo móvil o hasta 2km con Skycontroller 2

#### 3.1.2.2 Características de movimientos


El *Bebop 2* puede ser controlado en 4 grados de libertad, estos corresponden a un movimiento horizontal en el eje X, movimiento horizontal en el eje Y, movimiento angular alrededor del eje Z y movimiento vertical a lo largo del eje Z según se visualiza en la Figura 19, además en la Tabla 4, se muestran los parámetros configurables para los diferentes movimientos, es importante mencionar que no se pueden controlar directamente los ángulos de rotación *roll* y *pitch*, es decir, el *UAV* tiene un control de alto nivel.

(a) Movimiento lineal en el eje X



(b) Movimiento lineal en el eje Y



CONTINÚA 

(c) Movimiento angular alrededor del eje Z



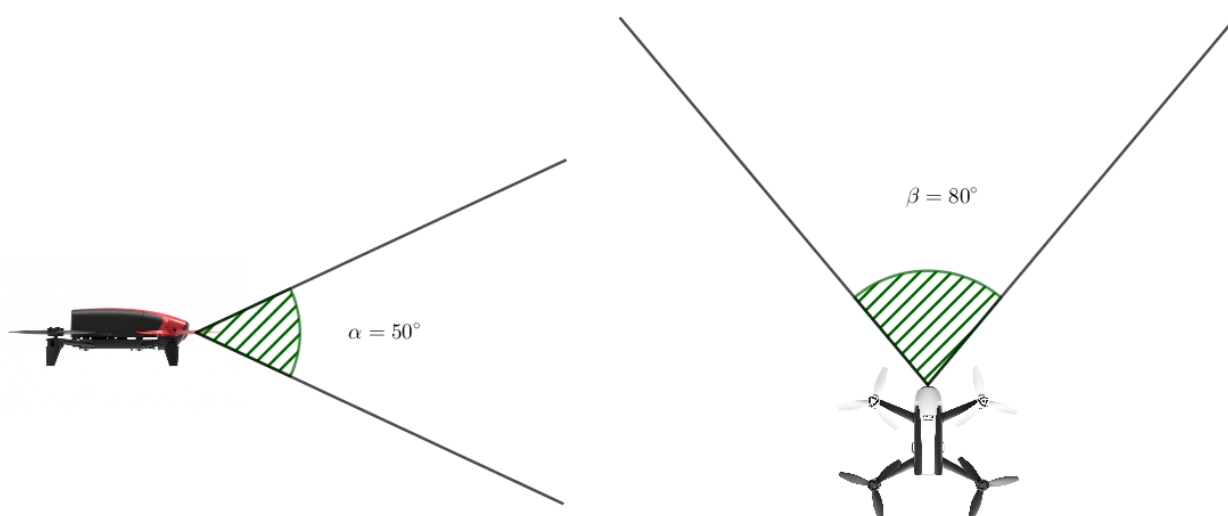
(d) Movimiento lineal en el eje Z

**Figura 19.** Movimientos independientes del Bebop 2**Tabla 4***Parámetros de movimiento configurables del Bebop 2*

Parámetros de movimiento	
<b>Inclinación</b>	Mínima: 5° - Máxima: 35°
<b>Velocidad de inclinación</b>	Mínima: 80°/s – Máxima: 300°/s
<b>Velocidad vertical</b>	Mínima: 0.5 m/s – Máxima: 6 m/s
<b>Velocidad de rotación</b>	Mínima: 10°/s – Máxima: 200°/s
<b>Velocidad horizontal</b>	Máxima: 16 m/s

### 3.1.2.3 Características de la cámara

La cámara del *Bebop 2* permite tener una vista cenital desde la perspectiva del dron lo que en teoría permitiría a este poder realizar una observación completa de su entorno con un campo de visión de 180° tanto para el eje horizontal o eje x (*pan*) y para el eje vertical o eje y (*tilt*), las características técnicas de la cámara y captura de imagen y video se describen en la Tabla 5. Lamentablemente en el streaming de video no se permite observar la imagen global, ya que Parrot utiliza la imagen global de la cámara para la estabilización digital de video, limitando al usuario a un streaming de video con un campo de visión de 80° en el eje horizontal y 50° en el eje vertical, esto se ilustra en la Figura 20.



**Figura 20.** Campo de visión del streaming de video del Bebop 2

**Tabla 5**

*Características de la cámara, captura de imagen y video*

<b>Imagen</b>	
<b>Cámara</b>	14 megapíxeles con lente gran angular
<b>Estabilización de video</b>	Si, estabilizado digitalmente en 3 ejes con definición Full HD 1080p
<b>Streaming de video</b>	Si
<b>Ángulo de visión</b>	180 grados
<b>Formato de fotos</b>	RAW, JPEG, DNG
<b>Resolución de imagen</b>	3800 x 3188
<b>Video</b>	
<b>Lente objetivo</b>	14 megapíxeles con sensor CMOS y lente “fish eye” de la empresa Sunny
<b>Ángulo de apertura</b>	180 grados, apertura de 1/2.3” (6 elementos de óptica)
<b>Resolución de video</b>	1920x1080p (30 fps)
<b>Codificación de video</b>	H264

### 3.2 Descripción del software del sistema

Las herramientas de software utilizadas en este proyecto de investigación fueron instaladas en la distribución de *Linux Ubuntu* 14.04.6 LTS la cual es una versión estable, se instaló previamente la versión 16.04 LTS, sin embargo, se presentaron incompatibilidades de software con el controlador del *UAV*. Además, las librerías y herramientas necesarias para el control del *UAV* están desarrolladas y cuentan con documentación más ampliada para esta versión de *Linux*, por lo que se encuentra pertinente realizar el proyecto sobre esta plataforma.

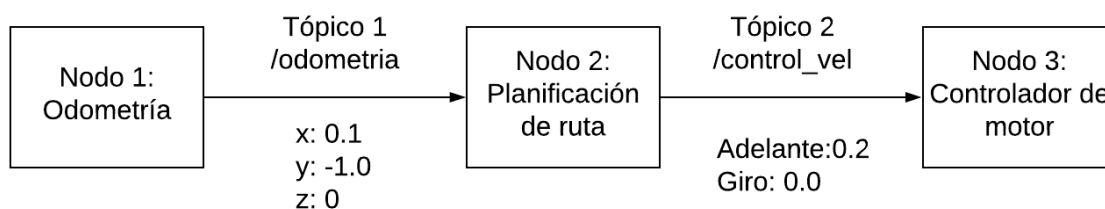
#### 3.2.1 *Robot Operating System (ROS)*

Es un sistema operativo de código abierto diseñado específicamente para el desarrollo de aplicaciones robóticas. Provee servicios similares a un sistema operativo común, incluyendo abstracción de hardware, por sus siglas en inglés *HAL*, control de bajo nivel de los dispositivos vinculados, implementación de funcionalidades comúnmente usadas por robots, pase de mensajes entre procesos y el manejo de paquetes. Está basado en una arquitectura de grafos que le permite la multiplexación de la información, es decir el procesamiento se lleva a cabo en “nodos” independientes, mismo que pueden publicar y suscribirse a mensaje de otros nodos, sean estos de sensores, señales de control, estados, planificaciones, entre otros. Una de las principales ventajas de *ROS*, es que tiene independencia de lenguaje de programación, actualmente existen implementaciones de *ROS* en *Python*, *C++* y *Lisp*, y se están realizando experimentaciones con Java y Lua.

##### 3.2.1.1 Nivel de computación gráfica de *ROS*

La computación gráfica es una red peer-to-peer de procesos de *ROS* que procesan datos juntos. *ROS* utiliza nodos que son procesos que realizan computación de datos. Se utilizan para distribuir

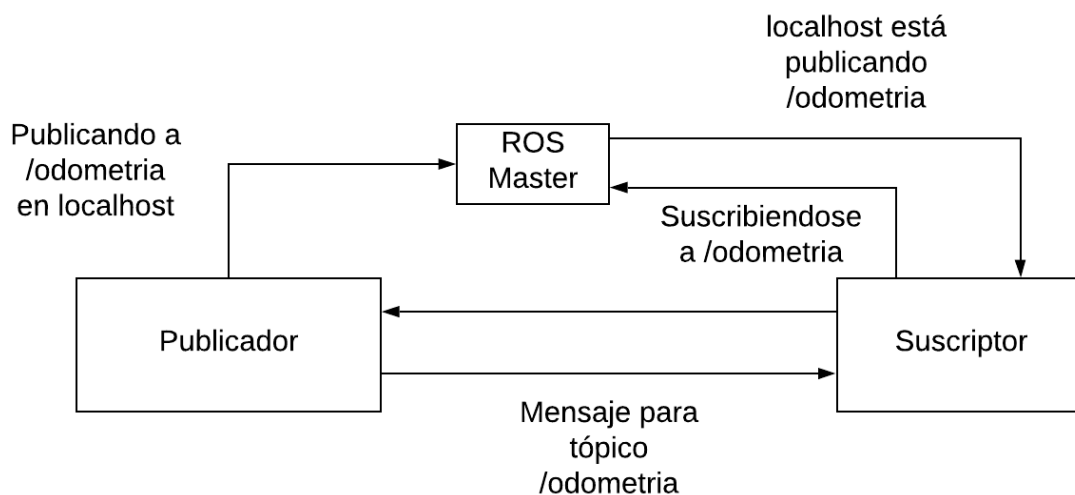
el flujo de información y permitir que la computación se descentralice, por ejemplo, un nodo puede manejar los datos provenientes del *GPS*, otro nodo maneja la velocidad de giro de las ruedas del robot, otro realiza la localización del robot, etc. (Ver Figura 21)



**Figura 21.** Grafo de computación de ROS

Estos se comunican a través de un nodo master llamado *ROS Master*, que permite el registro de los nodos para que estos puedan comunicarse entre sí, sin el nodo master el intercambio de datos no sería posible, ya que los nodos no podrían establecer comunicaciones.

La información que se intercambia entre nodos es llamada Mensaje y puede corresponder a cualquier tipo de dato o una composición de ellos. Estos mensajes se transportan a través de un sistema de transporte con semántica de publicadores y suscriptores, es decir, un nodo publica un mensaje publicándolo a un tópico específico. El nombre del tópico debe ser único y además debe identificar al tipo de dato, un nodo que esté interesado en un tipo de dato se suscribirá a un tópico específico. En la Figura 21, se muestra una arquitectura básica del intercambio de información entre nodos en *ROS*.



**Figura 22.** Intercambio de información entre nodos en ROS

### 3.2.2 Comunicación entre ROS y Bebop 2

ROS al contar con una rama para el desarrollo de nuevos paquetes (*ros\_pkg*) cuenta con varias librerías que hacen compatibles a otros robots con el mismo. Este es el caso del *Bebop 2*, que cuenta con el *driver bebop\_autonomy* compatible también con la versión anterior *Bebop 1*, está basado en el SDK oficial de *Parrot ARDroneSDK3*. El driver ha sido desarrollado en *Autonomy Lab* de la *Simon Fraser University* por Mani Monajjemi y otros contribuyentes. Este software se mantiene bajo mantenimiento de *Autonomy Lab*, *Dynamic Systems Lab* de la Universidad de Toronto y *Advanced Interactive Technologies Lab* del *ETH* de Zúrich.

El *driver* tiene la posibilidad de levantar tanto un nodo como un *nodelet*, la diferencia entre estos es que un nodo puede trabajar sobre una red de ordenadores y un *nodelet* solo puede intercambiar la información en el mismo ordenador, la ventaja que tiene un *nodelet* sobre un nodo es que provee una forma más rápida de pasar la información, es decir, si se trabaja con grandes flujos de información, como la imagen de una cámara o puntos en la nube, se recomienda usar un *nodelet*.

Antes de levantar el *nodelet*, es necesario:

- 1) Instalación del *driver bebop\_autonomy*
- 2) Que el ordenador se encuentre conectado a la red *Wi-Fi* del *UAV*

Una vez conectado el ordenador al *UAV*, se abre un terminal en *Ubuntu* y se añade como fuente al paquete del *driver* de *bebop\_autonomy* con la línea de comando:

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

Una vez añadida la fuente, con el comando *roslaunch* se lanzan los archivos con extensión *.launch*, tanto el nodo como *nodelet* se encuentran en el directorio *bebop\_driver*, además para tener un *streaming* de video de la cámara del dron en una ventana emergente, se puede lanzar el archivo *bebop\_nodelet\_iv.launch* que se encuentra en el directorio *bebop\_tools*. Se levanta el *nodelet* del directorio *bebop\_driver* con la siguiente línea de comando:

```
roslaunch bebop_driver bebop_nodelet.launch
```

Una vez levantado el *nodelet*, los tópicos que este se encuentra publicando se pueden verificar con el comando *rostopic list*.

### 3.2.2.1 Configuración de los parámetros del Bebop 2

Como se mencionó en la sección 3.2.2, el *driver* del *Bebop 2* es configurable, para levantar el nodo o *nodelet* se utiliza el archivo *defaults.yaml* ubicado en el directorio *bebop\_ws/src/bebop\_autonomy/bebop\_driver/cfg* que contiene la configuración del *driver*, en este

archivo se permite configurar parámetros de pilotaje, parámetros de movimiento, parámetros de red, parámetros de imagen y parámetros del *GPS*, la descripción y valores permitidos se describen en las Tablas 6 – 10.

**Tabla 6**  
*Parámetros configurables de pilotaje*

**Parámetros de pilotaje**

<b>Parámetro</b>	<b>Detalles</b>	<b>Tipo</b>
~PilotingSettingsMaxAltitudeCurrent	Máxima altitud en metros	Double
~PilotingSettingsMaxTiltCurrent	Máximo ángulo de inclinación en grados	Double
~PilotingSettingsAbsolutControlOn	1: Habilitado 0: Deshabilitado	Int
~PilotingSettingsMaxDistanceValue	Máxima distancia en metros desde el punto de partida	Double
~PilotingSettingsNoFlyOverMax DistanceShouldnotflyover	1: Si el dron no puede volar más allá de la distancia máxima 0: Si no existe limitaciones	Int
~PilotingSettingsBankedTurnValue	Habilita el giro con freno 1: Habilitado 0: Deshabilitado	Int
~PilotingSettingsMinAltitudeCurrent	Mínima altitud en metros	Double
~PilotingSettingsCirclingDirectionValue	Dirección de giro 1: Dirección de giro horaria 0: Dirección de giro antihoraria	Int
~PilotingSettingsCirclingRadiusValue	Radio de giro en metros	Int
~PilotingSettingsCirclingAltitudeValue	Altura de giro en metros	Int



~PilotingSettingsPitchModeValue	<p>Modo de pitch</p> <p>1: Comandos de pitch invertidos, valores positivos levantan la nariz del dron, valores negativos bajan la nariz del dron.</p> <p>0: Valores negativos levantan la nariz del dron, valores positivos bajan la nariz del dron.</p>	Int
---------------------------------	--	-----

**Tabla 7**  
*Parámetros de movimiento configurables*

**Parámetros de movimiento**

Parámetro	Detalles	Tipo
~SpeedSettingsMaxVerticalSpeedCurrent	Máxima velocidad vertical en m/s	Double
~SpeedSettingsMaxRotationSpeedCurrent	Máxima velocidad de rotación en grados por segundo	Double
~SpeedSettingsHullProtectionPresent	<p>Presencia de protectores para el Bebob 1</p> <p>1: Si tiene protección</p> <p>0: Si no tiene protección</p>	Int
~SpeedSettingsOutdoorOutdoor	<p>Vuelo en exteriores o interiores</p> <p>1: Vuelo en exteriores</p> <p>0: Vuelo en interiores</p>	Int
~SpeedSettingsMaxPitchRollRotationSpeedCurrent	Máxima velocidad de rotación en pitch y roll en grados por segundo	Double

**Tabla 8**  
*Parámetros de red configurables*

**Parámetros de red**

<b>Parámetro</b>	<b>Detalles</b>	<b>Tipo</b>
~NetworkSettingsWifiSelectionType	Selección del tipo de red Wi-Fi 1: Selección manual 0: Selección automática	Int
~NetworkSettingsWifiSelectionBand	Bandas permitidas: 2.4GHz, 5 GHz o todas 2: 2.4 GHz y 5 GHz 1: 5 GHz 0: 2.4 GHz	Int
~NetworkSettingsWifiSelectionChannel	Selección del canal	Int

**Tabla 9**  
*Parámetros de imagen configurables*

**Parámetros de imagen**

<b>Parámetro</b>	<b>Detalles</b>	<b>Tipo</b>
~PictureSettingsVideoStabilizationModeMode	Estabilización de video 3: Imagen sigue los movimientos del dron 2: Video estabilizado solamente en roll 1: Video estabilizado solamente en pitch 0: Video estabilizado en roll y pitch	Int
~PictureSettingsVideoRecordingModeMode	Modo de grabación de video	Int

	<p>1: Maximiza el tiempo de grabación</p> <p>0: Maximiza la calidad de la grabación</p>	
~PictureSettingsVideoFramerateFramerate	<p>Tasa de cuadros por segundo</p> <p>2: 29.97 cuadros por segundo</p> <p>1: 25 cuadros por segundo</p> <p>0: 23.976 cuadros por segundo</p>	Int
~PictureSettingsVideoResolutionsType	<p>Resolución de streaming de video y grabación</p> <p>1: 720p grabación, 720 streaming</p> <p>0: 1080p grabación, 480p streaming</p>	Int

**Tabla 10**  
*Parámetros de GPS configurables*

**Parámetros de GPS**

<b>Parámetro</b>	<b>Detalles</b>	<b>Tipo</b>
~GPSSettingsHomeTypeType	<p>Tipo de posición Home</p> <p>2: El dron tratará de regresar a la posición de la actual (o última) “follow me”</p> <p>1: El dron tratará de regresar a la posición del piloto</p> <p>0: El dron tratará de regresar a la posición de despegue</p>	Int
~GPSSettingsReturnHomeDelayDelay	Retraso de la señal del GPS	Int

De los parámetros listados en las tablas, se han configurado parámetros tanto para el pilotaje, movimiento e imagen, en la Tabla 11 se muestran los valores actuales para cada uno de los parámetros.

**Tabla 11**  
*Parámetros de pilotaje, velocidad e imagen reconfigurados*

Parámetro	Valor predeterminado	Valor configurado
PilotingSettingsMinAltitudeCurrent	0.3 m	1.0 m
PilotingSettingsMaxTiltCurrent	5°	8°
SpeedSettingsMaxRotationSpeedCurrent	13°/s	100°/s
SpeedSettingsMaxVerticalSpeedCurrent	1 m/s	1 m/s
SpeedSettingsMaxPitchRollRotationSpeedCurrent	80°/s	100°/s
PictureSettingsVideoStabilizationModeMode	0	0
PictureSettingsVideoRecordingModeMode	1	0
PictureSettingsVideoResolutionsType	0	1

Los valores de los parámetros de movimiento se han tomado en base a los valores predeterminados por la aplicación *FreeFlight Pro* para vuelos de grabación, sin embargo, se ha prescindido del parámetro de velocidad de giro angular ya que esta al ser de 13 grados por segundo no es suficiente para tomar curvas cerradas en la trayectoria definida en los siguientes apartados.

### 3.2.2.2 Tipos de mensaje

Los mensajes son un componente principal dentro de la arquitectura de computación de grafos de *ROS*. Estos están definidos por un *header* o cabecera que especifican la estructura del mensaje

y los tipos de datos que contiene el mismo. A continuación, se detalla la cabecera de los mensajes que se van a utilizar para la creación de los publicadores y suscriptores de clase:

- 1) Mensaje std\_msgs/Bool.msg

```
{bool data}
```

- 2) Mensaje std\_msgs/Empty.msg

- 3) Mensaje std\_msgs/Float32.msg

```
{float32 data}
```

- 4) Mensaje geometry\_msgs/Twist.msg

```
{{vector3 linear: {float32 x: , float32 y: , float32 z: }},{vector3 angular:{ float32
x: , float32 y: , float32 z: }}}
```

- 5) Mensaje nav\_msgs/Odometry.msg

```
{PoseWithCovariance pose:{Pose pose:{Point position:{float64 x: ,float64 y:
,float64 z: }, Quaternion orientation:{float64 x: ,float64 y: ,float64 z: ,float64 w:
}}}}
```

- 6) Mensaje sensor\_msgs/Image.msg

```
{uint32 height, uint32 width, string encoding, uint8 is_bigendian, uint32 step,
uint8[] data}
```

- 7) Mensaje bebop\_msgs/CommonCommonStateBatteryStateChanged.msg

```
{uint8 percent}
```

### 3.2.2.3 Publicadores de clase

La creación de una clase con publicadores y suscriptores es imprescindible para la escalabilidad del sistema, por lo que, se ha creado la clase “*bebop2*” con sus respectivos publicadores, que le

permitirán al dron navegar de manera autónoma. La librería *rospy* permite la creación de publicadores en el lenguaje de programación Python en cualquiera de sus versiones, estos necesitan de tres parámetros para poder crearse.

- 1) Tópico en el cual se va a publicar el mensaje
- 2) Tipo de mensaje
- 3) Tiempo de espera para la publicación del mensaje (*queue\_size*)

En la Tabla 12 se listan los publicadores creados para la clase “*bebop2*”, el tópico en el cual publican, el tipo de mensaje y el tiempo de espera para cada uno.

**Tabla 12**  
*Publicadores de la clase "bebop2"*

Publicador	Tópico	Mensaje	queue_size
Pub_Twist	‘/bebop/cmd_vel’	geometry_msgs.msg.Twist	10 ms
Pub_Takeoff	‘/bebop/takeoff’	std_msgs.msg.Empty	100 ms
Pub_Land	‘/bebop/land’	std_msgs.msg.Empty	100 ms
Pub_Reset	‘/bebop/reset’	std_msgs.msg.Empty	100 ms
Pub_CamTwist	‘/bebop/camera_control’	geometry_msgs.msg.Twist	10 ms
Pub_Exposure	‘/bebop/set_exposure’	std_msgs.msg.Float32	10 ms
Pub_Record	‘/bebop/record’	std_msgs.msg.Bool	100 ms

A continuación, se detallan cada uno de los publicadores listado en la tabla anterior necesarios para el control del UAV.

## a) Pub\_Takeoff

Publicador necesario para el despegue del UAV. Para publicar el mensaje vacío se invoca a función *publish()* de la librería *rospy* sin argumentos.

## b) Pub\_Land

Publicador necesario para el aterrizaje del UAV. Para publicar el mensaje vacío se invoca a función *publish()* de la librería *rospy* sin argumentos.

## c) Pub\_Reset

Publicador necesario para el aterrizaje en caso de emergencia. Para publicar el mensaje vacío se invoca a función *publish()* de la librería *rospy* sin argumentos.

## d) Pub\_Record

Publicador necesario para grabar las misiones del UAV en primera persona. Para publicar un mensaje de tipo *Bool* se invoca a la función *publish()* de la librería *rospy* con los argumentos: *False* si se requiere detener la grabación y *True* si se requiere iniciar la grabación.

## e) Pub\_Exposure

Publicador necesario para controlar la exposición a luz que es de utilidad para controlar la iluminación en una imagen. Para publicar un mensaje de tipo *Float32* se invoca a la función *publish()* de la librería *rospy* con los argumentos del rango de  $[-3.0 \dots 3.0]$  siendo el valor de  $-3.0$  la mínima cantidad de exposición a la luz que significaría una imagen oscura y el valor de  $3.0$  la máxima cantidad de exposición a la luz resultando en una imagen con alto brillo.

## f) Pub\_CamTwist

Publicador que controlar la posición de la cámara virtual. Para publicar un mensaje de tipo Twist, se debe especificar los valores de *angular.y* para el movimiento en *tilt* y *angular.z* para el movimiento en *pan* de la cámara, esto se detalla en la Tabla 13. Similar a los anteriores publicadores se requiere la invocación de la función *publish()*, y el rango de valores es [-80.0 ... 80.0].

**Tabla 13**

*Valores para el control de movimiento de la cámara virtual*

Parámetro	Valor negativo	Valor positivo
Angular.y	Tilt hacia abajo	Tilt hacia arriba
Angular.z	Panning hacia la derecha	Panning hacia la izquierda


## g) Pub\_Twist

Publicador que controla los movimientos del UAV. Como se explicó en los tipos de mensajes el mensaje a ser publicado en el tópico */bebop/cmd\_vel* corresponde a un mensaje de tipo *Twist* que contiene tres valores lineales y tres valores angular. En la Tabla 14 se resume el movimiento que se produce al publicar valores en el rango de [-1.0 ... 1.0].

**Tabla 14**

*Parámetros para el control de movimiento del UAV*

Parámetro	Valor negativo	Valor positivo
Linear.x	Hacia atrás	Hacia adelante
Linear.y	Hacia la derecha	Hacia la izquierda
Linear.z	Hacia abajo	Hacia arriba

CONTINÚA 



Angular.x	Inclinación en roll en sentido horario	Inclinación en roll en sentido anti horario
Angular.y	Inclinación en pitch en sentido horario	Inclinación en pitch en sentido anti horario
Angular.z	Giro en el propio eje en sentido horario	Giro en el propio eje en sentido anti horario

### 3.2.2.4 Suscriptores de clase

Al igual que los publicadores, los suscriptores pertenecen a la clase “*bebop2*”. Estos se encargan de recibir los datos de la imagen de la cámara y la pose del *UAV*. La librería *rospy* permite la creación de suscriptores, estos requieren de los siguientes parámetros:

- 1) Tópico al cual se va a suscribir
- 2) Tipo de mensaje
- 3) Función *callback*

La función de *callback* se refiere a la función que se ejecuta desde la creación del suscriptor, ya que cada suscriptor crea un hilo que se está ejecutando de forma paralela al programa principal.

**Tabla 15**

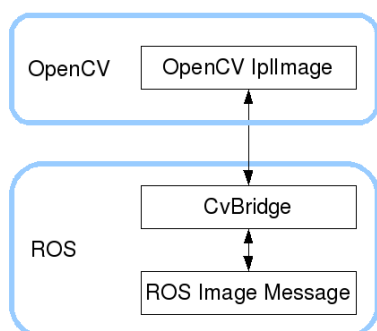
*Suscriptores de la clase "bebop2"*

Suscriptor	Tópico	Mensaje
Sub_Battery	‘/bebop/states/common/CommonState /BatteryStateChanged’	CommonCommonStateBattery StateChanged
Sub_Odom	‘/bebop/odom’	nav_msgs.msg.Odometry
Sub_Image	‘/bebop/image_raw’	sensor_msgs.msg.Image

A continuación, se detalla cada uno de los suscriptores necesarios para el control del *UAV*.

a) Suscriptor de imágenes

El suscriptor de imágenes obtiene las imágenes de la cámara a bordo del *UAV*. Estas son publicadas en el tópico */bebop/image\_raw* desde el momento en el que se levanta el *nodelet*. Los datos que provienen del mensaje *sensor\_msgs/Image.msg* son un mensaje de *ROS* y no puede ser interpretado por los lenguajes de programación *C++* o *Python*. Para lograr que los mensajes de *ROS* sean interpretados por *Python* existe una librería llamada *cv\_bridge*, que permite la conversión de mensajes de *ROS* a un array de *NumPy*, específicamente a un array *UNIT8* de tres dimensiones, el primer y segundo corresponden a las intensidades de cada uno de los píxeles, dependiendo del tamaño de la imagen obtenida del streaming de video, y el tercer valor corresponde al número de canales, en este caso es de tres canales (*BGR*) ya que la mayoría de aplicaciones de visión por computadora utilizan este orden de los canales. En la Figura 23 se muestra el proceso de conversión del mensaje de *ROS* a una imagen que puede ser procesada con algoritmos incluidos en las librerías de *OpenCV*.



**Figura 23.** Proceso de conversión de mensaje de ROS a una imagen

La función *callback\_image* se encarga del proceso antes descrito, a esta función se le pasa una instancia de la clase *self* y los datos “*data*” que provienen del mensaje de *ROS*.

b) Suscriptor de estado de batería

El suscriptor de batería obtiene el porcentaje de batería disponible en el dron, es de utilidad para evitar que el dron caiga en caída libre en el momento de realizar una misión. El tópico al cual se suscribe es */bebop/states/common/CommonStates/BatteryStateChanged* y a través de la función *callback\_battery* se obtiene el porcentaje de batería y el valor que corresponde a este es *data.percent* en donde “*data*” corresponde al mensaje de *ROS* que se obtiene el tópico.

c) Suscriptor de odometría

El suscriptor de odometría es útil al inicio del trayecto del dron, ya que una vez que despegue este debe alcanzar una altura predeterminada para tener la capacidad de observar los obstáculos. El tópico al cual se suscribe es */bebop/odom* y a través de la función *callback\_odom* se obtienen los datos de odometría. El valor de la altura actual del dron corresponde a *data.pose.pose.position.z*.

## CAPÍTULO IV

### 4. DETECCIÓN Y SEGUIMIENTO DE LA TRAYECTORIA

En este capítulo se trata la problemática sobre la detección y seguimiento de la trayectoria a través de las imágenes terrestres que se obtienen desde el *UAV* en la estación terrestre.

Una detección correcta disminuirá el ruido introducido por interferencias en la transmisión, además utilizar técnicas que optimicen el tiempo de procesamiento para mantener la tasa de refrescamiento de imágenes alrededor de 30 Hz es primordial para tomar acciones de control en tiempo real. Es de igual importancia la estimación del punto medio de la trayectoria, si se obtiene una mala estimación del punto de referencia el controlador no será capaz de mantener al *UAV* dentro de la trayectoria trazada en el piso.

Para la detección y seguimiento de la trayectoria se han establecido dos etapas:

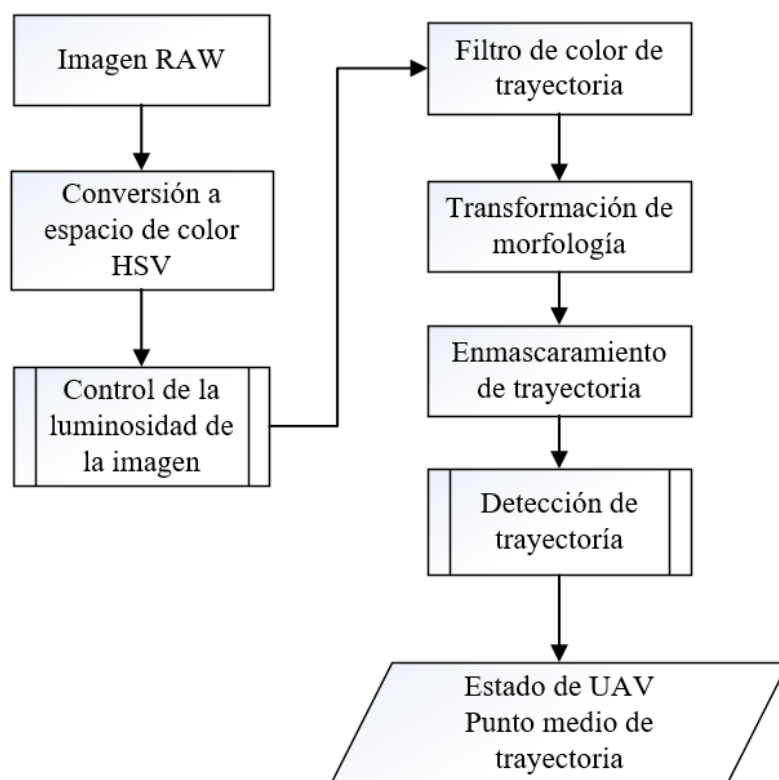
- 1) Pre procesamiento de las imágenes terrestres
- 2) Detección de trayectoria
  - a. Detección de puntos de interés
  - b. Estimación de punto medio de la trayectoria como punto de referencia para el controlador
  - c. Seguimiento de la trayectoria

En los siguientes apartados se detallan las etapas establecidas, se comparan tanto eficiencia de los algoritmos de detección de puntos de interés como el número de características que resultan de cada uno. En adición, se compara la estimación de la referencia de la trayectoria con medidas

estadísticas clásicas y las mismas a diferencia de utilizar el filtro de *Kalman* como estimador de la referencia de la trayectoria.

#### 4.1 Pre procesamiento de las imágenes terrestres

Como se menciona en el trabajo de (Scaramuzza & Fraundorfer, 2011) para poder distinguir movimientos en un entorno este debe ser rugoso y tener cambios de saturaciones, con esta premisa se ha seleccionado el color rojo para distinguir la trayectoria en comparación con el color de la superficie en donde se encuentra. En la Figura 24 se resumen el proceso de detección de la trayectoria y su pre procesamiento, cada uno de los procesos se detallan en los siguientes apartados.



**Figura 24.** Detección de trayectoria

### 4.1.1 Conversión a espacio de color HSV

La detección de colores en un espacio de color *BGR* no se puede realizar de forma distintiva, es decir, se pierde la intuición del ser humano al describir los colores. Esta es la ventaja que ofrece el espacio de color *HSV* que a diferencia de *RGB* o *BGR*, la tonalidad está representada en un solo canal (*H*) y de estas tonalidades se pueden filtrar fácilmente los colores más saturados o con mayor presencia de brillo, brindando así la capacidad de distinguir una misma tonalidad con diferentes niveles de iluminación.

En el caso de imágenes de 8-bit o 16-bit, los valores de B, G y R son escalados a un valor de punto flotante en el rango de 0 a 1 (OpenCV, 2019). Entonces, la conversión de *BGR* a *HSV* se realiza de la siguiente forma:

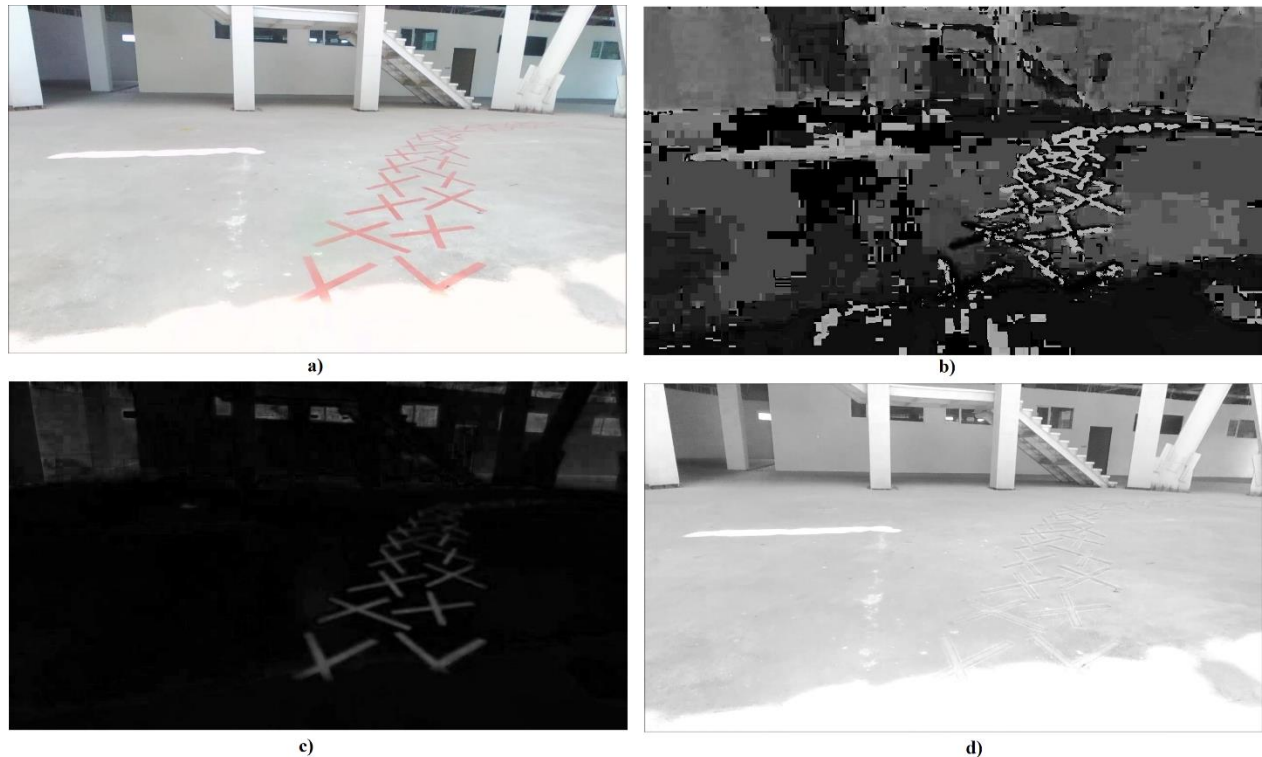
$$V \leftarrow \max(R, G, B) \quad (24)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R, G, B)}{V}, & \text{si } V \neq 0 \\ 0, & \text{cualquier otro caso} \end{cases} \quad (25)$$

$$H \leftarrow \begin{cases} \frac{60(G - B)}{V - \min(R, G, B)}, & \text{si } V = R \\ \frac{120 + 60(B - R)}{V - \min(R, G, B)}, & \text{si } V = G \\ \frac{240 + 60(R - G)}{V - \min(R, G, B)}, & \text{si } V = B \end{cases} \quad (26)$$

Si  $H < 0$  entonces,  $H \leftarrow H + 360$ . Los valores de salida que se obtienen son  $0 \leq V \leq 1, 0 \leq S \leq 1, 0 \leq H \leq 360$ .

Seguido de esto los valores se convierte al tipo de dato de destino, al tratarse de una imagen de 8-bit tenemos:  $V \leftarrow 255 * V$ ,  $S \leftarrow 255 * S$ ,  $H \leftarrow H/2$  para ajustarse de 0 a 179. En la Figura 25, se puede visualizar los tres canales separados de la imagen en *HSV*.

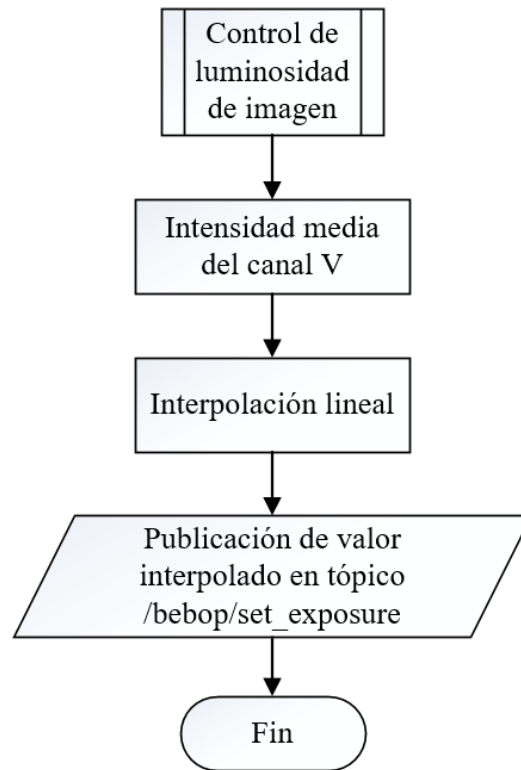


**Figura 25.** a) Imagen en BGR, b) Canal H, c) Canal S, d) Canal V

#### 4.1.2 Control de iluminación presente en las imágenes de la cámara

El cambio de iluminación es una interferencia que se introduce en las imágenes del *UAV*, este fenómeno produce el encandilamiento del lente objetivo del *UAV* y se producen detecciones erróneas.

Para solucionar el problema de encandilamiento ante cambios bruscos de iluminación se realiza un control proporcional para el cambio de iluminación (Ver Figura 26).



**Figura 26.** Diagrama de flujo de subproceso de control de luminosidad de imagen

En primer lugar, se realiza una transformación de color de *BGR* a el espacio de color *HSV*. Como se muestran en la Figura 25 el canal que indica la cantidad de iluminación presente en la imagen es el canal *V*, por lo tanto se obtiene la media de las intensidades de los pixeles como se muestra a continuación:

$$\mu_s = \frac{\sum_{i=0}^N V(u)}{N} \quad (27)$$

Donde,

- $\mu_s$  corresponde a la media de las intensidades de los pixeles del canal *V*
- $V(u)$  corresponde a la imagen del canal *V*
- $N$  corresponde al número total de pixeles



Luego, el porcentaje de iluminación corresponde a la relación entre la intensidad media de los píxeles del canal V y el valor máximo de intensidad de un píxel en una imagen de 8-bit, que corresponde a 255. Teniendo que un valor de  $\mu_s$  cercano a 0 corresponde a una imagen con una iluminación muy baja y un valor de  $\mu_s$  cercano a 255 corresponde a una imagen con una iluminación muy alta.

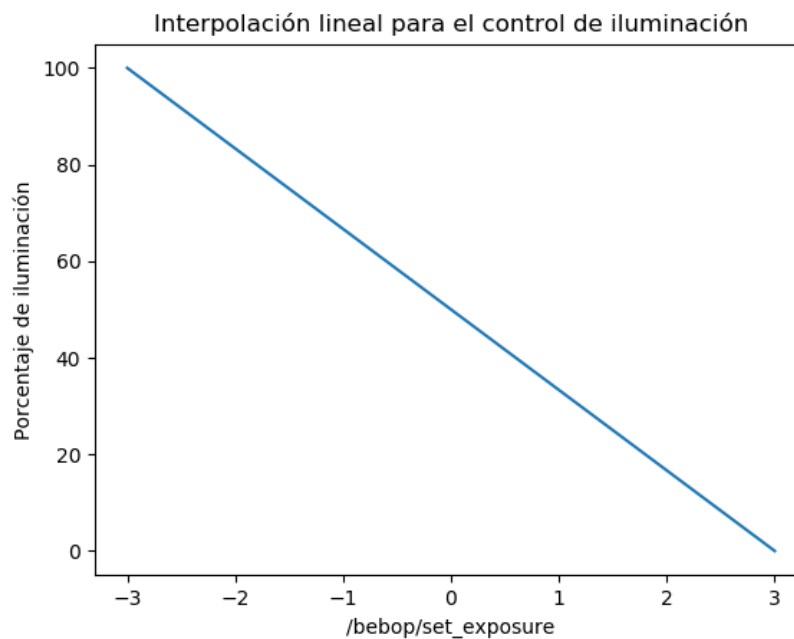
$$\%iluminación = \frac{\mu_s}{2^8 - 1} \times 100\% \quad (28)$$

Para controlar el nivel de iluminación en la imagen se debe publicar un mensaje de tipo *float32* en el tópic */bebop/set\_exposure* según la Tabla 12, además también se indicó que el rango de valores a ser publicados debe estar entre [-3.0 ... 3.0] siendo -3.0 una exposición mínima a la luz y 3.0 una exposición máxima a la luz. Tomando en cuenta estas premisas, se realiza una interpolación lineal de los valores de [0 ... 100] a [3.0 ... -3.0], obteniendo la ecuación de la recta

$$y = -\frac{3}{50}x + 3 \quad (29)$$

En donde,

- $y$  corresponde al valor a ser publicado en el tópic */bebop/set\_exposure* en el rango de [-3.0 ... 3.0]
- $x$  corresponde al porcentaje de iluminación



**Figura 27.** Interpolación lineal de la señal de control y el porcentaje de iluminación

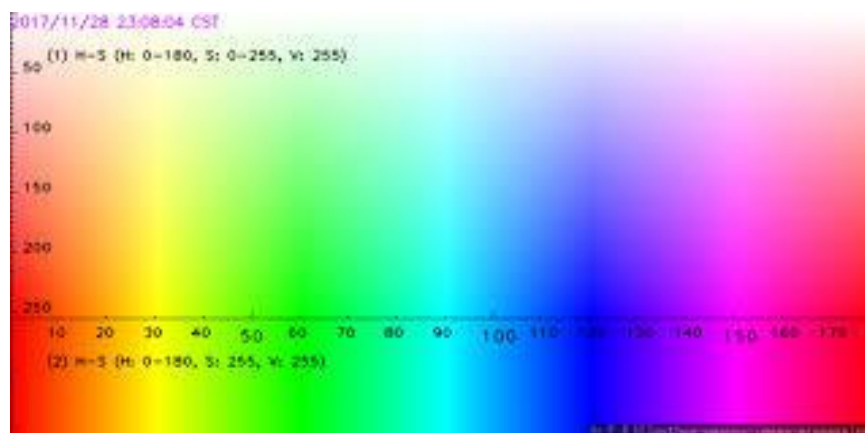
El porcentaje de iluminación será la referencia del controlador de luminosidad en la imagen. El nivel de iluminación (*Set Point*) que se va a mantener es de 75% para evitar el encandilamiento del lente del UAV. En la Figura 28, se evidencia la diferencia entre un frame de una secuencia de video con y sin el control de iluminación.



**Figura 28.** Frame con control de iluminación (izquierda) vs. frame sin control de iluminación (derecha)

### 4.1.3 Enmascaramiento de la trayectoria

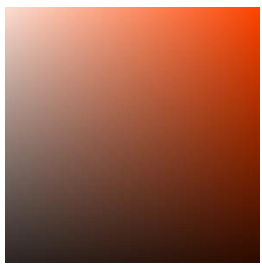
Una vez realizada la conversión al espacio de color *HSV* y el control de iluminación, se enmascara la trayectoria en la imagen, filtrando los valores que no correspondan a la misma. Es importante mencionar que para el enmascaramiento de la trayectoria fueron necesarios dos rangos, ya que el matiz de color rojo se encuentra en los extremos del canal H como se muestra en la Figura 29. En la Tabla 16, se muestran los rangos de colores seleccionados para enmascarar la trayectoria.



*Figura 29.* Matiz de colores del canal H

**Tabla 16**

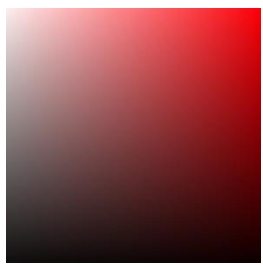
*Rango de valores para enmascaramiento de trayectoria*

Rango	Limite	Canal H	Canal S	Canal V	Color
1	Inferior	0	44	121	
	Superior	8	255	255	

*CONTINÚA* →

2

Inferior	159	15	0
Superior	179	255	255



Utilizando los rangos 1 y 2 de la Tabla 16, se obtiene las máscaras que se muestran en la Figura 30 y Figura 31 para diferentes frames.

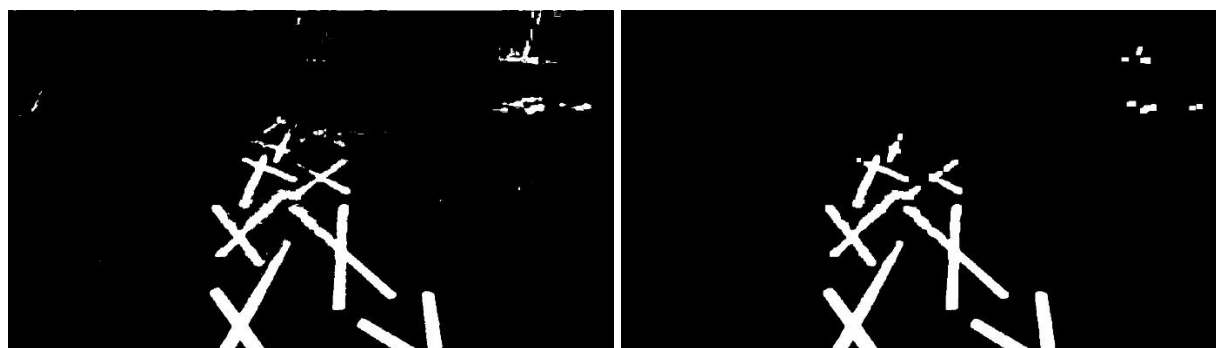


**Figura 30.** Máscaras para el frame 1, rango 1 (izquierda), rango 2 (derecha)



**Figura 31.** Máscaras para el frame 2, rango 1 (izquierda), rango 2 (derecha)

La máscara que resulta de filtrar los rangos de valores corresponde a una imagen binaria, la cual permite realizar transformaciones morfológicas y operaciones lógicas. La operación morfológica *OPENING* permite realizar la operación *ERODE* y *DILATE* conjuntamente, esta es útil para disminuir la presencia de ruido en la imagen binaria, en la Figura 32 se muestra la diferencia de la imagen binaria con y sin la transformación morfológica.



*Figura 32.* Máscara sin y con transformación morfológica

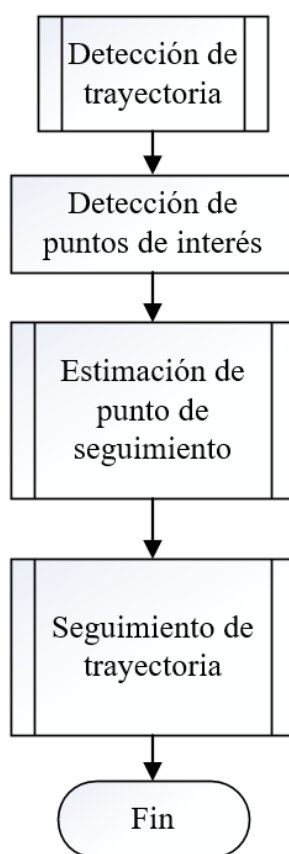
Seguido de esto, para enmascarar la imagen original se realiza una operación lógica *AND* con la máscara, los *frames* resultantes de esta operación se muestran en la Figura 33.



*Figura 33.* Frames 1 y 2 enmascarados

## 4.2 Detección de trayectoria

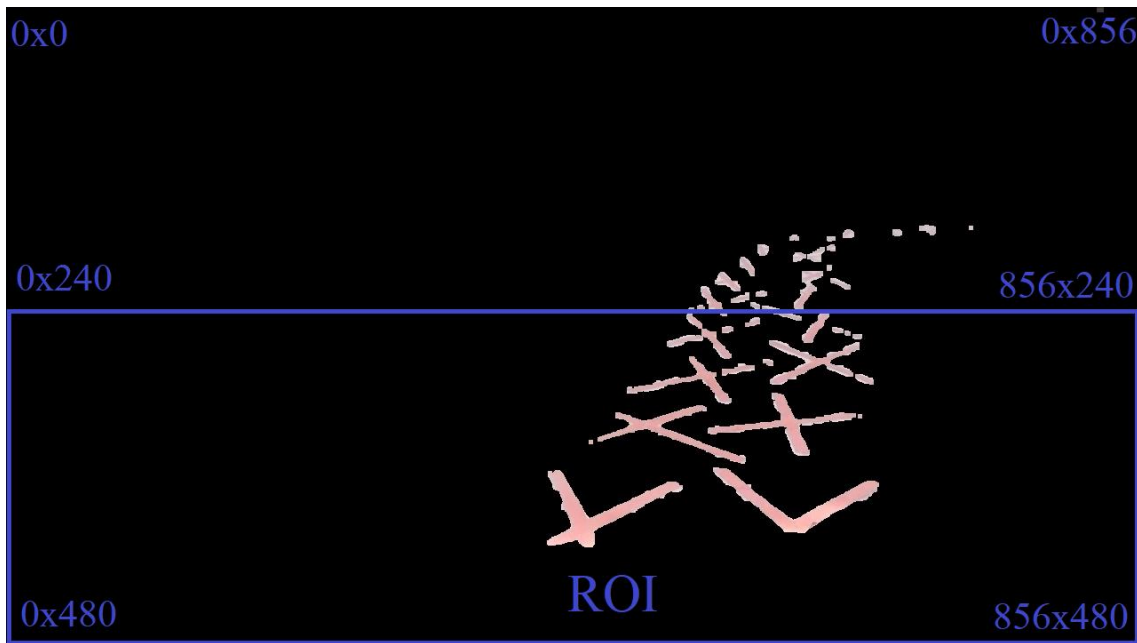
El proceso de detección comprende desde el proceso de detección de puntos de interés hasta el seguimiento de la trayectoria, pasando por la estimación del punto de seguimiento de la misma, en la Figura 34, se resume el proceso de detección de trayectoria.



**Figura 34.** Detección de trayectoria

### 4.2.1 Detección de puntos de interés

A partir de la trayectoria enmascarada de la Figura 33 se realiza la detección de puntos de interés, para este propósito se toma en cuenta una región de interés, por sus siglas en inglés *ROI* (*Region of Interest*) correspondiente al 50% de los píxeles más cercanos al UAV. (Ver Figura 35)



**Figura 35.** Región de interés para la detección de puntos de interés

La región de interés tiene como rango visible dependiente de las siguientes variables:

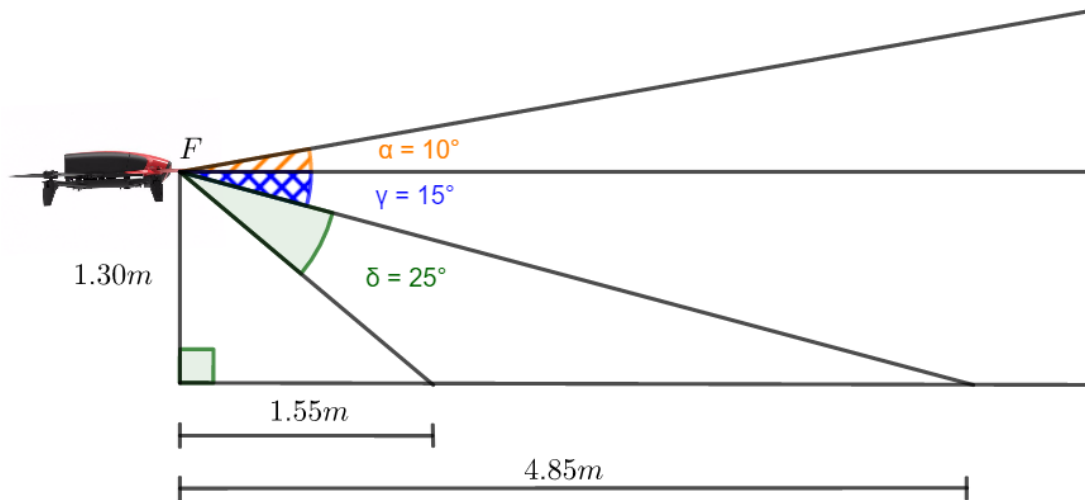
- Altura del *UAV* en metros
- Posición de la cámara virtual

La distancia mínima de detección está dada por la relación  $x_{min} = H \times \tan(90 - (25 + \gamma))$  y la distancia máxima de detección  $x_{max} = H \times \tan(90 - \gamma)$  en donde  $\gamma$  corresponde a la posición de la cámara virtual en tilt.

Las condiciones bajo las que se realizaron las pruebas de detección de puntos de interés en la trayectoria son (Ver Figura 36):

- Condición de iluminación: 2000 a 5000 lux
- Altura de detección: De 1 a 1.4 metros
- Ángulo de inclinación (*tilt*):  $-15^\circ$

- Distancia mínima de detección: 1.55 m
- Distancia máxima de detección: 4.85 m
- Algoritmos de detección: *SIFT*, *SURF*, *FAST*, *ORB*



**Figura 36.** Rango visible para la región de interés

La Tabla 17 y Tabla 18 resumen los tiempos de detección de puntos característicos y el número de puntos detectados por los algoritmos seleccionados. El tiempo de detección corresponde al tiempo medio de detección para cada *frame* de la video secuencia, de igual forma, el número de puntos detectados corresponde a la media de puntos detectados en cada *frame*. En adición en la Figura 37 y Figura 38 se muestra el tiempo medio de detección y la media de puntos característicos detectados para las seis secuencias de video que se han tomado para las pruebas de detección de trayectoria.



**Tabla 17**

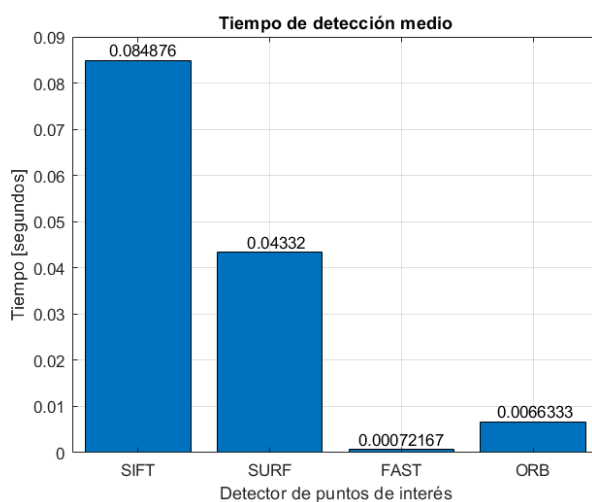
*Tiempos de detección y número de puntos característicos para aproximadamente 2000 lux*

	Secuencia de video 1		Secuencia de video 2		Secuencia de video 3	
	Tiempo de detección [s]	Número de puntos	Tiempo de detección [s]	Número de puntos	Tiempo de detección [s]	Número de puntos
<b>SIFT</b>	0.08562	241	0.08383	222	0.08374	291
<b>SURF</b>	0.04114	580	0.0427	730	0.04396	892
<b>FAST</b>	0.0007	354	0.00068	329	0.00072	458
<b>ORB</b>	0.00608	1460	0.00563	1514	0.00685	2059

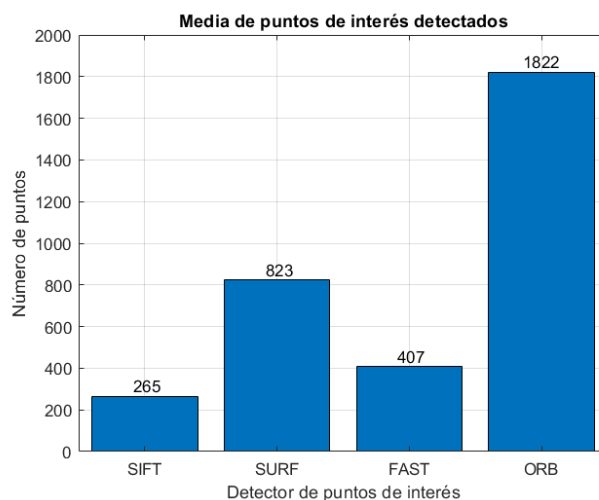
**Tabla 18**

*Tiempos de detección y número de puntos característicos para aproximadamente 5000 lux*

	Secuencia de video 4		Secuencia de video 5		Secuencia de video 6	
	Tiempo de detección [s]	Número de puntos	Tiempo de detección [s]	Número de puntos	Tiempo de detección [s]	Número de puntos
<b>SIFT</b>	0.08987	306	0.082105	238	0.08409	294
<b>SURF</b>	0.04341	978	0.0426	661	0.04611	1095
<b>FAST</b>	0.00076	498	0.00071	351	0.00076	454
<b>ORB</b>	0.00681	2220	0.00595	1508	0.00848	2173



**Figura 37.** Tiempo de detección medio de puntos característicos de la trayectoria



**Figura 38.** Media de número de puntos característicos detectados en la trayectoria

De las figuras y tablas que se muestran en la sección superior, se puede concluir que para diferentes condiciones de iluminación el algoritmo que muestran menor tiempo de detección es el algoritmo *FAST*, pero presenta una desventaja al tener una media de 407 puntos de interés. Mientras que el algoritmo *ORB* tiene una media de 1822 puntos de interés y un tiempo de detección 6 ms, lo cual significaría que sería capaz de detectar puntos de interés a una tasa aproximada de 150 *FPS*, por otra parte, los algoritmos *SIFT* y *SURF* cuentan con tiempo medio de detección de 80 ms y 40 ms, lo que significa que pueden procesar 13 y 25 *FPS*, respectivamente. Se concluye que los algoritmos *FAST* y *ORB* son la mejor opción para detectar puntos de interés en la trayectoria. Sin embargo, en los siguientes apartados se tomarán en cuenta los algoritmos *SIFT* y *SURF* a propósito de comparar el error medio cuadrático que presentan en la estimación del punto medio de la trayectoria.

#### 4.2.2 Estimación del punto medio de la trayectoria

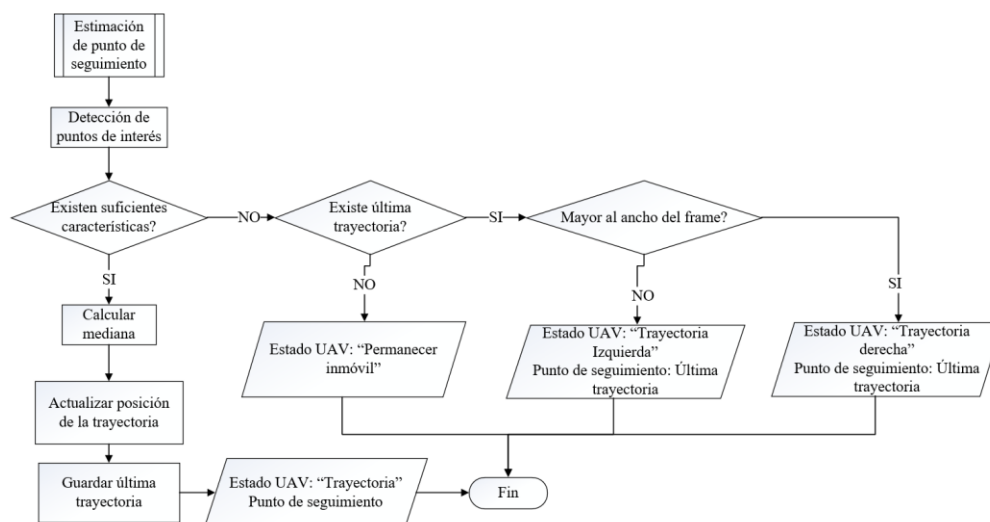
La estimación del punto medio de la trayectoria constituye el resultado final del proceso de detección de trayectoria, este punto corresponde a la referencia (*Set Point*) que deberá seguir el

dron (Grijalva & Aguilar, 2019). Para estimar el punto de seguimiento se han utilizado tres métodos:

- 1) Mediana
- 2) Media
- 3) Media con Filtro de *Kalman*

En los siguientes apartados, se resume el proceso de obtención del punto de seguimiento y el error medio cuadrático que presentan en comparación con el punto medio marcado a mano en 500 *frames* consecutivos de las video secuencias analizadas en secciones anteriores.

#### 4.2.2.1 Estimación con mediana de los puntos de interés



**Figura 39.** Diagrama de flujo de proceso de estimación de punto medio con mediana

En la Figura 39 se muestra el proceso de estimación del punto medio de la trayectoria cuando se obtiene la mediana de los puntos de interés detectados en la máscara, se ha seleccionado un umbral de puntos de interés como la segunda parte del mínimo número de características obtenidos

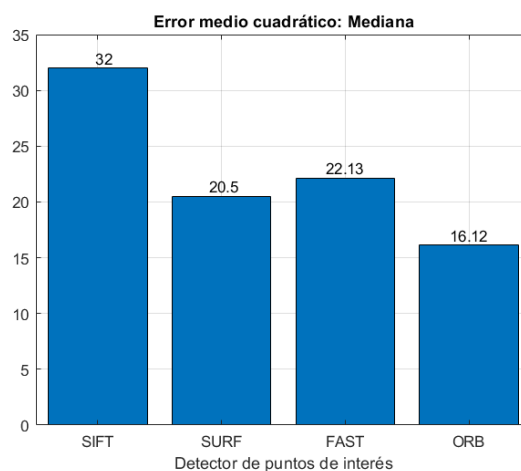
en la Figura 38, este valor corresponde a 111 puntos mínimos que deben ser detectados para considerar que en un frame existe una trayectoria.

En la Tabla 19 se muestra el error medio cuadrático obtenido para las 6 secuencias de video y en la Figura 40 se evidencia que la detección de puntos de interés con *ORB* presenta menos error en comparación con los otros detectores, se obtiene un *RMSE* de 16.12% en promedio para las 6 secuencias de video y una diferencia de 4.68% en comparación con el detector *SURF*.

**Tabla 19**

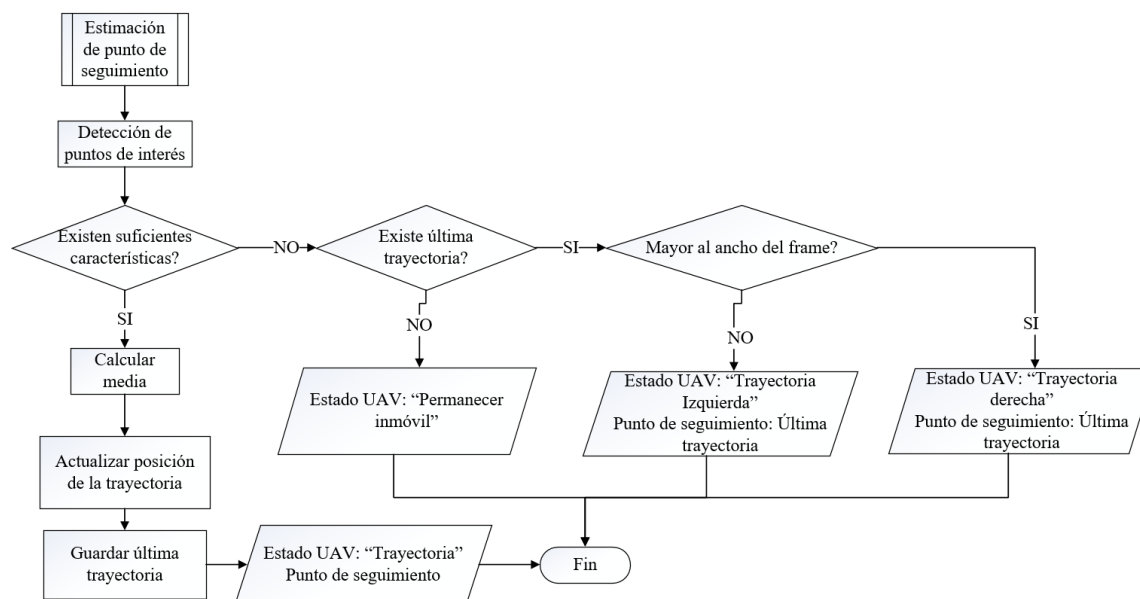
*Error medio cuadrático entre mediana y punto marcado*

N°	SIFT	SUFT	FAST	ORB
<b>Video 1</b>	33.81	20.29	19.89	12.63
<b>Video 2</b>	32.65	21.56	21.18	11.32
<b>Video 3</b>	31.33	20.78	20.61	19.4
<b>Video 4</b>	46.12	23.62	32.25	23.29
<b>Video 5</b>	16.76	17.27	17.83	12.55
<b>Video 6</b>	31.34	19.46	21.02	17.54



**Figura 40.** Error medio cuadrático de la mediana de la trayectoria

### 4.2.2.2 Estimación con media de los puntos de interés



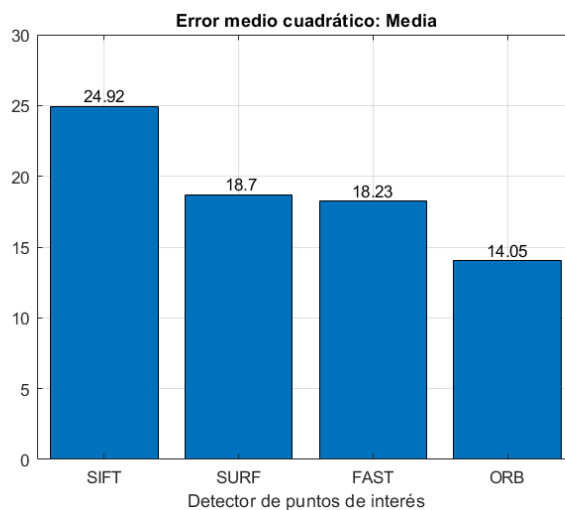
**Figura 41.** Diagrama de flujo de proceso de estimación de punto medio con media

En la Figura 41 se muestra el proceso de estimación del punto medio de la trayectoria cuando se obtiene la media de los puntos de interés detectados en la máscara, de igual manera que para la detección con la mediana, se ha seleccionado un umbral de puntos de interés como la segunda parte del mínimo número de características obtenidos en la Figura 38, correspondiente a 111.

En la Tabla 20 se muestra el error medio cuadrático obtenido para las 6 secuencias de video y en la Figura 42 se evidencia que la detección de puntos de interés con *ORB* presenta menos error en comparación con los otros detectores, se obtiene un *RMSE* de 14.05% en promedio para las 6 secuencias de video y una diferencia de 4.18% en comparación con el detector *FAST*.

**Tabla 20***Error medio cuadrático entre media y punto marcado*

N°	SIFT	SURF	FAST	ORB
<b>Video 1</b>	25.83	19.02	17.57	13.49
<b>Video 2</b>	23.65	20.57	15.65	10.75
<b>Video 3</b>	27.4	21.4	20.5	16.77
<b>Video 4</b>	32.54	21.22	25.32	18.38
<b>Video 5</b>	12.4	11.15	11.41	9.88
<b>Video 6</b>	27.67	18.83	19.26	15

**Figura 42.** Error medio cuadrático de la media de la trayectoria

#### 4.2.2.3 Estimación con Filtro de Kalman y media de los puntos de interés

La estimación del punto de seguimiento con el filtro de Kalman comprende un proceso más complejo que los dos anteriores, en primer lugar, se deben obtener los parámetros de incertidumbre en la posición y movimiento del dron, esto se debe a que el dron en el estado de “Permanecer inmóvil” realmente no lo hace y existen movimientos aleatorios que introducen error en las mediciones.

Para obtener el parámetro de error en la medición de posición se toma una secuencia de video con las siguientes condiciones:

- Condición de iluminación: ~5000 lux
- Altura de detección: 1 metro
- Ángulo de inclinación (*tilt*): -15°
- Estado del dron: Permanece inmóvil

De esta secuencia de video se obtiene la transformación afín de *frame a frame* y se obtiene la media de la suma acumulada de las traslaciones que se produjeron en el eje x, esto se representa en la ecuación:

$$\sigma_{tx}^2 = \frac{\sum_{i=0}^N tx_i}{N} \quad (30)$$

Para obtener el error en la medición del movimiento, de la misma secuencia de video se toma la suma acumulada de la diferencia absoluta de las traslaciones entre frames consecutivos representado por la ecuación:

$$\sigma_m^2 = \frac{\sum_{i=0}^{N-1} |tx_{i+1} - tx_i|}{N - 1} \quad (31)$$

De esta manera se obtiene que los valores de  $\sigma_{tx}$  y  $\sigma_m$ , son de 0.15957 y 0.13216 pixeles respectivamente. Con los valores obtenidos para el modelo es posible actualizar y predecir la posición del punto de seguimiento. En primer lugar, se deben obtener las mediciones de la posición de la trayectoria, para esto se forma un arreglo de mediciones con la media de los puntos de interés obtenidos en cada frame, seguido de esto se toma el valor promedio de las mediciones, esto se representa con la ecuación:

$$x = \frac{\sum_{i=1}^N M_t}{N} \quad (32)$$

En donde,

- $X$  corresponde a la posición media de la trayectoria
- $M_t$  corresponde a un arreglo de mediciones de la posición de la trayectoria
- $N$  corresponde al número de mediciones de la posición trayectoria

Seguido de esto se calcula el movimiento o la velocidad con la que se han producido cambios en la posición de la trayectoria, que corresponde a la derivada numérica de la posición de la trayectoria, representada en la ecuación:

$$\dot{x} = \frac{\sum_{i=1}^{N-1} (x_{i+1} - x_i)}{N - 1} \quad (33)$$

En donde,

- $\dot{x}$  corresponde a la velocidad de cambio media de la posición de la trayectoria
- $x_{i+1}$  corresponde a la posición de la trayectoria en el *frame*  $i + 1$
- $x_i$  corresponde a la posición de la trayectoria en el *frame*  $i$
- $N$  corresponde al número de mediciones de la posición de la trayectoria

Con las mediciones de posición y velocidad de cambio, se actualiza la posición de la trayectoria utilizando una suma ponderada de las medias anteriores, normalizada por sus factores de ponderación (Nishad, 2019; Teow, 2018), esto se representa en la ecuación:

$$\mu' = \frac{\sigma_{tx}^2 \mu + \sigma_m^2 v}{\sigma_{tx}^2 + \sigma_m^2} \quad (34)$$

En donde,



- $\mu'$  corresponde a la posición de la trayectoria actualizada
- $\sigma_{tx}^2$  corresponde a la incertidumbre o varianza en las mediciones del modelo
- $\mu$  corresponde a la posición a priori de la trayectoria
- $\sigma_m^2$  corresponde a la incertidumbre o varianza de la posición a priori de la trayectoria
- $v$  corresponde a la posición media de la nueva medición ( $x$ )

De manera similar se actualiza el valor de la varianza, representada por la ecuación:

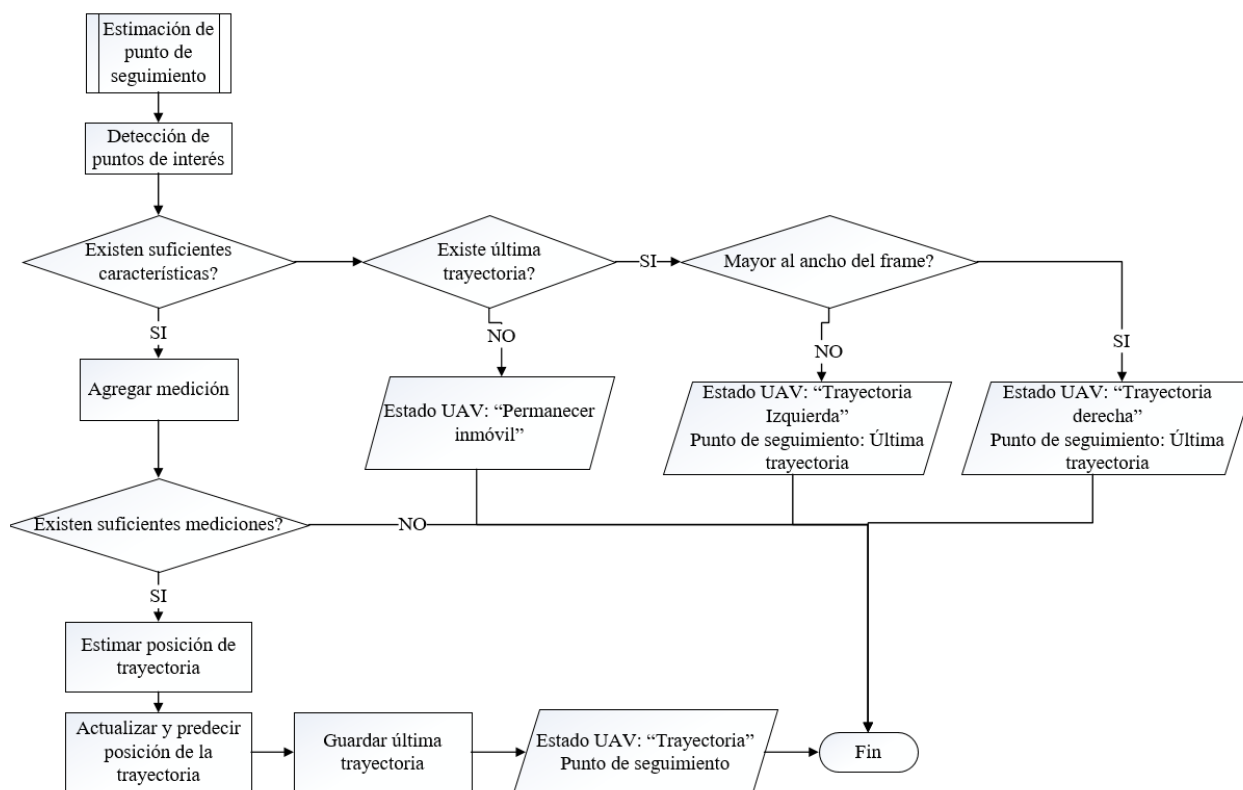
$$\sigma^{2'} = \frac{1}{\frac{1}{\sigma_{tx}^2} + \frac{1}{\sigma_m^2}} \quad (35)$$

Finalmente, con los valores actualizados se realiza la predicción de la posición de la trayectoria que resulta ser una suma del valor actualizado más el valor del movimiento actualizado, esto se representa en las ecuaciones:

$$\mu' \leftarrow \mu + v \quad (36)$$

$$\sigma^{2'} \leftarrow \sigma_m^2 + \sigma_{tx}^2 \quad (37)$$

Los valores de  $\mu'$  y  $\sigma^{2'}$  corresponderán a la posición de la trayectoria y la incertidumbre en la medición de la misma, respectivamente. En la Figura 43 se resume el proceso de estimación del punto de seguimiento con el filtro de Kalman.



**Figura 43.** Diagrama de flujo de proceso de estimación de punto medio con Filtro de Kalman

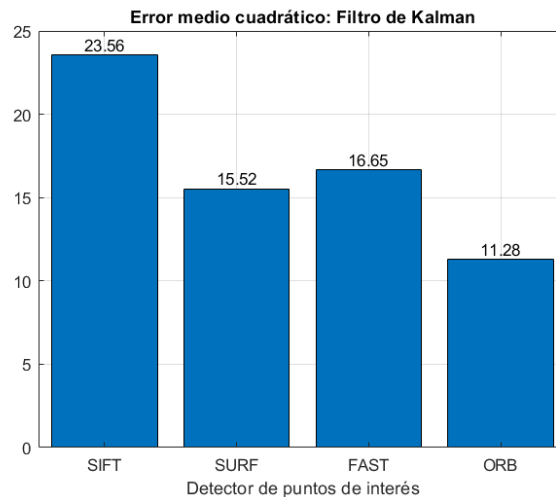
**Tabla 21**

*Error medio cuadrático entre estimación con Filtro de Kalman y punto marcado*

Nº	SIFT	SURF	FAST	ORB
<b>Video 1</b>	24.227	18.47	16.51	12.03
<b>Video 2</b>	25.78	11.45	15.77	10.59
<b>Video 3</b>	23.4	16.7	16.89	10.46
<b>Video 4</b>	30.32	20.29	23.16	15.69
<b>Video 5</b>	11.51	10.34	10.62	8.82
<b>Video 6</b>	26.12	15.88	16.97	10.1

En la Tabla 21 se muestra el error medio cuadrático obtenido para las 6 secuencias de video y en la Figura 44 se evidencia que la detección de puntos de interés con *ORB* presenta menos error

en comparación con los otros detectores, se obtiene un  $RMSE$  de 11.28% en promedio para las 6 secuencias de video y una diferencia de 4.24% en comparación con el detector *SURF*.



**Figura 44.** Error medio cuadrático de la Estimación con Filtro de Kalman de la trayectoria

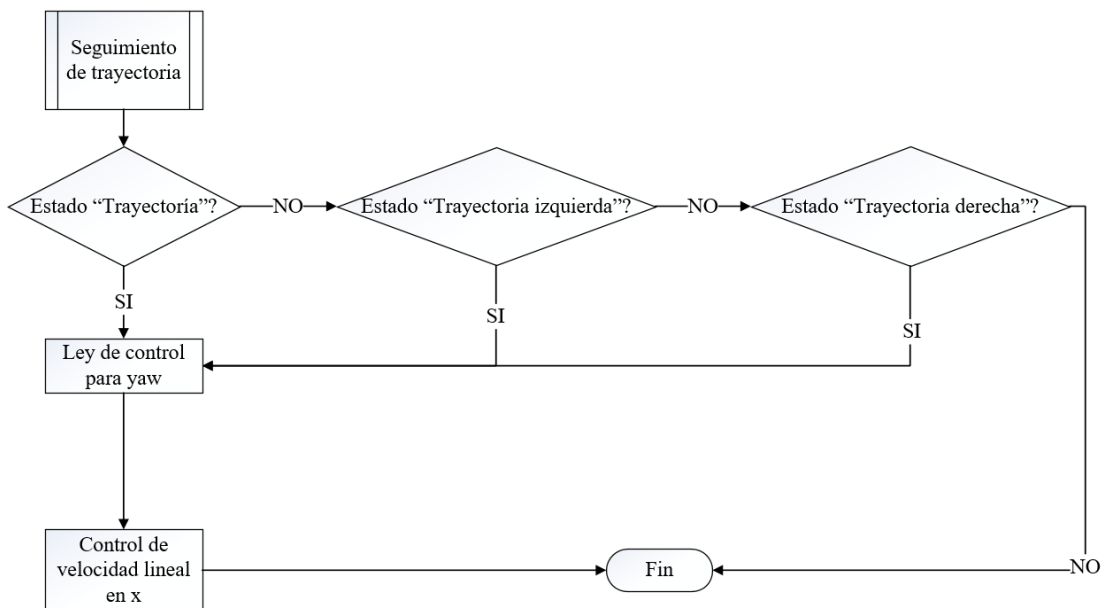
### 4.2.3 Seguimiento de trayectoria

El seguimiento de trayectoria es el proceso en el cual se sigue al punto medio obtenido de la estimación realizada en el apartado anterior dependiendo del estado en el que se encuentre el *UAV*. Los estados que se han considerado para el seguimiento de la trayectoria se definen a continuación:

- Trayectoria: Corresponde al estado en el cual el *UAV* ha detectado una trayectoria y esta se encuentra en el campo de visión del mismo.
- Trayectoria izquierda: Corresponde al estado en el cual el *UAV* ha perdido la trayectoria habiendo previamente detectado la misma y el último punto de seguimiento que se guardó corresponde a un punto menor a la segunda parte del ancho de la imagen del *UAV*.

- Trayectoria derecha: Corresponde al estado en el cual el *UAV* ha perdido la trayectoria habiendo previamente detectado la misma y el último punto de seguimiento que se guardó corresponde a un punto mayor a la segunda parte del ancho de la imagen del *UAV*.
- Permanecer inmóvil: Corresponde al estado en el cual el *UAV* ha despegado y no ha encontrado una trayectoria.

Una vez que se ha comprobado el estado del *UAV*, como se muestra en la Figura 45 y haciendo referencia a la Tabla 1, se toma una acción de control para *Yaw*, es decir, se generan movimientos en el eje del dron, simulando el movimiento de un vehículo terrestre de configuración diferencial.



**Figura 45.** Diagrama de flujo de seguimiento de trayectoria

#### 4.2.3.1 Diseño del controlador

Para poder tomar la acción de control sobre el *UAV* se requiere un modelo matemático que describa la dinámica del *UAV*, en la literatura (Hyunchul Shim, Hyoun Jin, & Sastry, 2000;

Dorobantu, y otros, 2011; Hoffer, Coopmans, Jensen, & Chen, 2014; Kendoul, 2012) es común encontrar la identificación de sistemas aéreos no tripulados mediante la utilización de sensores inerciales (Aguilar, Casaliglla, & Pólit, Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles, 2017), desafortunadamente como se mencionó en el capítulo correspondiente a la descripción del sistema el *driver bebop\_autonomy* limita la frecuencia con la cual se envían los datos inerciales, esta frecuencia corresponde a 5 Hz, misma que no es suficiente para modelar el sistema y tomar acciones en tiempo real a la hora de realizar el seguimiento de la trayectoria, por otro lado los datos de la cámara son enviados con una frecuencia de 30 Hz que son convenientes para obtener un modelo servo visual del *UAV* (Salcedo, 2018; Aguilar, Morales, Ruiz, & Abad, RRT\* GL Based Optimal Path Planning for Real-Time Navigation of UAVs, 2017).

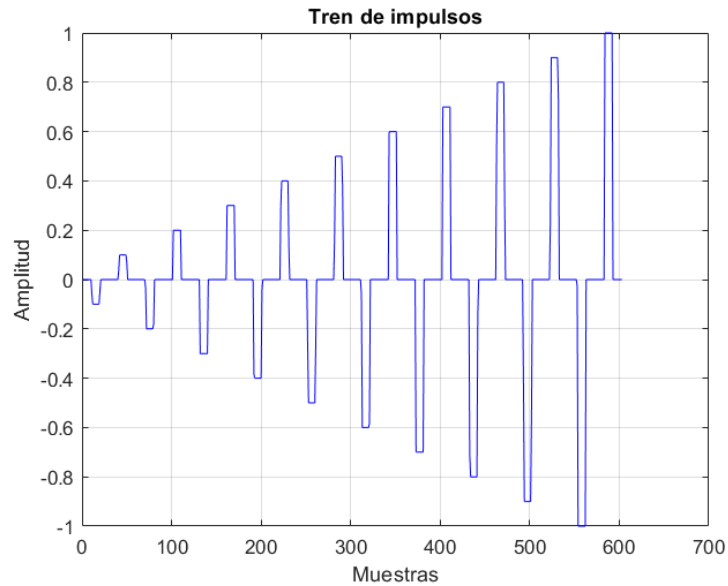
Según (Salcedo, 2018) se puede aproximar cada uno de los cuatro movimientos independientes que posee el *UAV* (sobre el eje x, y, z y en su propio eje) como un modelo *SISO*.

En el caso del seguimiento de trayectoria se necesita modelar el movimiento sobre el propio eje del dron (*Yaw*), para llevar a cabo esto, se publica en el tópico */bebop/cmd\_vel* (mientras el *UAV* se encuentra volando) un tren de impulsos como el que se muestra en la Figura 46, específicamente en el parámetro *angular.z* que hace referencia al movimiento angular alrededor del eje z, además se graba la secuencia de video producida por el tren de impulsos para su posterior procesamiento.

Las condiciones en las cuales se grabó la secuencia de video son las siguientes:

- Condición de iluminación: ~2000 lux
- Altura de detección: 1 metro

- Ángulo de inclinación (*tilt*):  $-15^\circ$
- Estado del dron: Gira sobre su propio eje



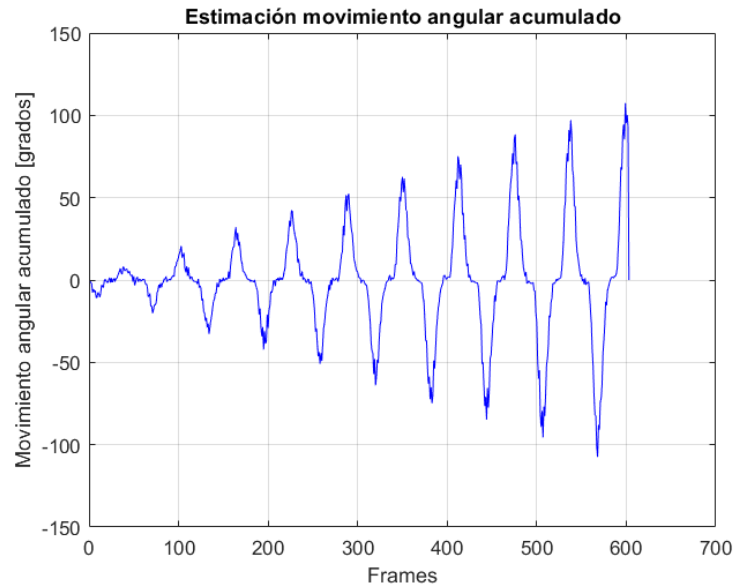
**Figura 46.** Tren de impulsos para la obtención del modelo servo visual

Una vez obtenida la grabación de video, se lleva a cabo el proceso de estimación de movimiento a través de una transformación *Affine2D* (Transformación de similitud). Para lograr esto se obtienen los puntos de interés del *frame*  $I_{t-1}$  y se comparan con los puntos de interés del *frame*  $I_t$  y a través de la función *cv2.estimateAffinePartial2D* de *OpenCV* se obtiene la matriz de transformación de la forma:

$$M = \begin{pmatrix} \cos \theta \cdot s & -\sin \theta \cdot s & t_x \\ \sin \theta \cdot s & \cos \theta \cdot s & t_y \end{pmatrix} \quad (38)$$

De la matriz de la ecuación (38) se obtiene el parámetro  $\theta$  que corresponde a la rotación que existe de *frame* a *frame*, y se realiza la estimación de la rotación en función de la acumulación del

movimiento. En la Figura 47, se muestra el resultado de la acumulación del parámetro  $\theta$  para el tren de impulsos de la Figura 46.

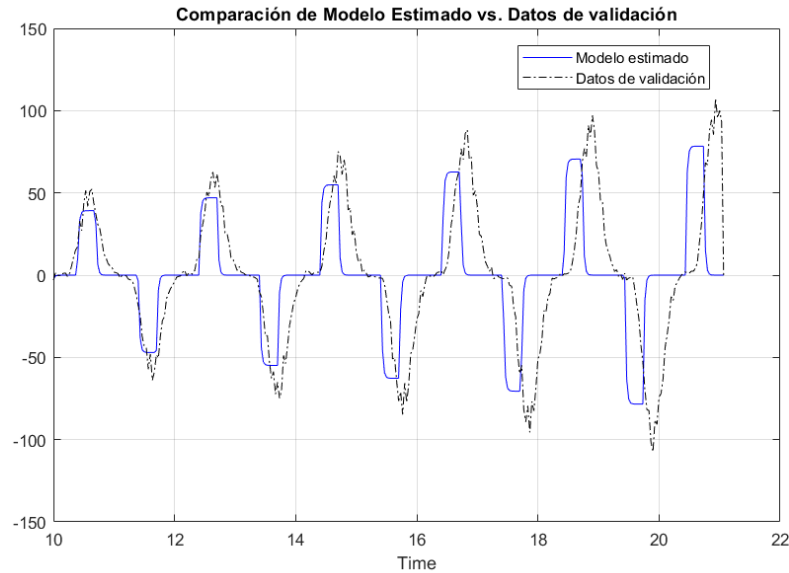


**Figura 47.** Estimación de movimiento angular acumulado

Con los datos obtenidos de la estimación de movimiento angular, se realiza la identificación del modelo de la planta, para esto, se utiliza el *Toolkit* de *Matlab systemIdentification* y se realiza la estimación de una planta de proceso de primer orden correspondiente a la función de transferencia de la ecuación ( 39):

$$G(s) = \frac{K_p}{1 + T_p \cdot s} \quad (39)$$

Con los parámetros correspondientes a  $K_p = 83.3539$  y  $T_p = 0.034752$ , en la Figura 48, se observa el modelo estimado vs los datos de validación correspondientes a la estimación de movimiento acumulado de la Figura 47.



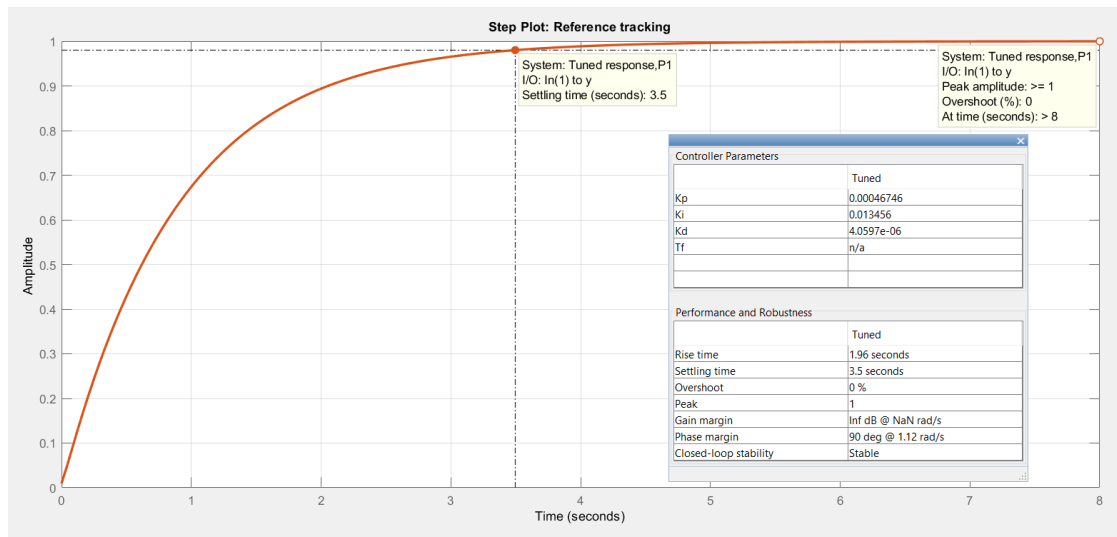
**Figura 48.** Comparación de modelo estimado para Yaw vs. Datos de validación del movimiento angular acumulado

A continuación, utilizando el *Toolkit* de Matlab *pidTuner*, se sintoniza un controlador *PID*, con las siguientes especificaciones de *performance*.

$$M_p = 0\% , \quad t_s = 3.5 \quad (40)$$

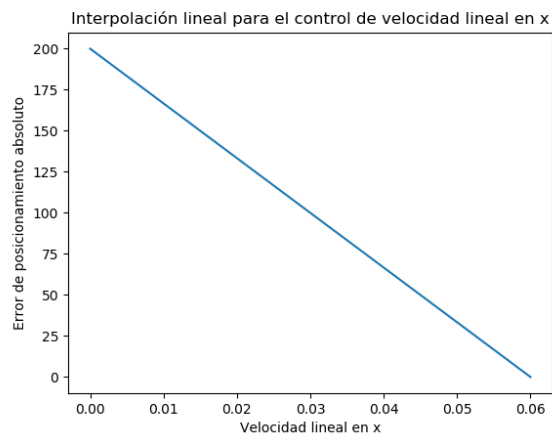
Mediante la sintonización del controlador *PID* en el *Toolkit pidTuner*, se obtienen las ganancias correspondientes a  $K_p = 0.00046746$ ,  $K_i = 0.013456$  y  $K_d = 4.0597e-06$ . La respuesta en estado estacionario del conjunto Planta-Controlador, se muestra en la Figura 49.





**Figura 49.** Respuesta al escalón unitario del conjunto Planta-Controlador para el movimiento angular alrededor del eje Z (Yaw)

En cuanto al controlador para generar el movimiento lineal a lo largo del eje X (*pitch*) que permitirá al dron desplazarse sobre la trayectoria, se decide controlar la velocidad lineal como una relación inversamente proporcional al error de posicionamiento absoluto en la trayectoria, es decir, si el error tiende a cero la velocidad lineal en X es máxima, y si el error tiende a ser mayor a la segunda parte del ancho del frame (428 píxeles), la velocidad es mínima. En la Figura 50, se muestra la interpolación correspondiente entre la relación del error de posicionamiento de la trayectoria y la velocidad lineal a lo largo del eje X.



**Figura 50.** Interpolación lineal para el control de velocidad lineal en x

La ecuación que describe la velocidad a lo largo del eje X, es la siguiente:

$$x = -\frac{6}{20000}y + \frac{6}{100}$$

En donde,

- $x$  corresponde a la velocidad lineal en el eje x
- $y$  corresponde al error de posicionamiento absoluto

## CAPÍTULO V

### 5. DETECCIÓN Y EVASIÓN DE OBSTÁCULOS

En este capítulo se trata la detección y evasión de los obstáculos a través de las imágenes frontales que se obtienen desde el *UAV* en la estación terrestre.

Se tiene como premisa que los colores de los obstáculos corresponderán a colores que tengan contraste con el fondo, es decir que estos se diferencien del ambiente percibido por el *UAV*.

Para la detección y evasión del obstáculo se han establecido dos etapas:

- 1) Pre procesamiento de las imágenes frontales
- 2) Detección de obstáculo
  - a. Detección de contornos
  - b. Estimación del centro de masa del obstáculo
  - c. Evasión del obstáculo

En los siguientes apartados, se detalla el proceso de estimación del centro de masa del obstáculo, se compara la precisión que se obtiene con el método convencional para determinar centros de masa en *OpenCV* versus el método propuesto para el proyecto de investigación que tiene como componente principal la estimación del centro de masa a través de un filtro de Kalman.

#### 5.1 Pre procesamiento de imágenes frontales

Este proceso se realiza de forma similar tanto para las imágenes frontales como para las imágenes terrestres, ya que los algoritmos de conversión de espacio de color y control de

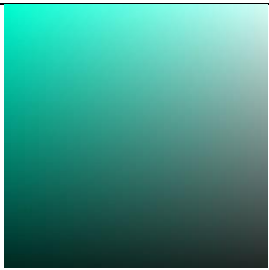
iluminación son independientes de la posición de la cámara, en consecuencia, se describe directamente el proceso de enmascaramiento de los obstáculos.

### 5.1.1 Enmascaramiento del obstáculo

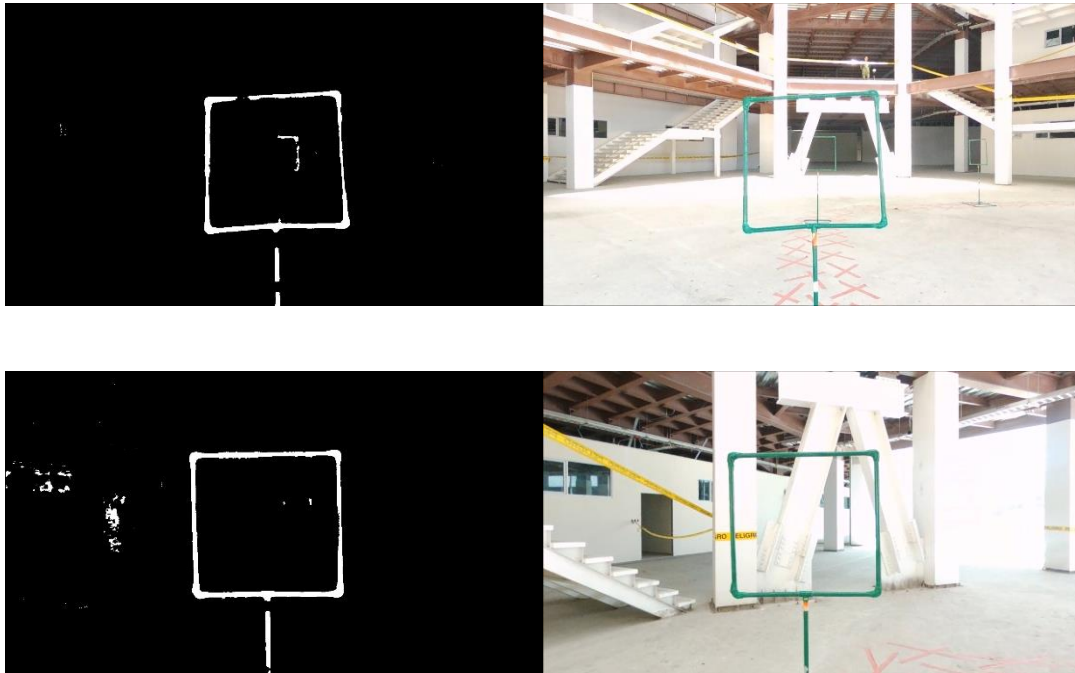
Una vez realizada la conversión al espacio de color *HSV* y el control de iluminación, se enmascara el obstáculo presente en la imagen, filtrando las intensidades de los píxeles que no correspondan a un obstáculo. A diferencia de la trayectoria, el matiz de los obstáculos se encuentra en un solo rango de colores, por lo que no es necesaria la adición de un rango adicional. En la Tabla 22 se muestra el rango del color necesario para enmascarar el obstáculo.

**Tabla 22**

*Rango de valores para enmascaramiento del obstáculo*

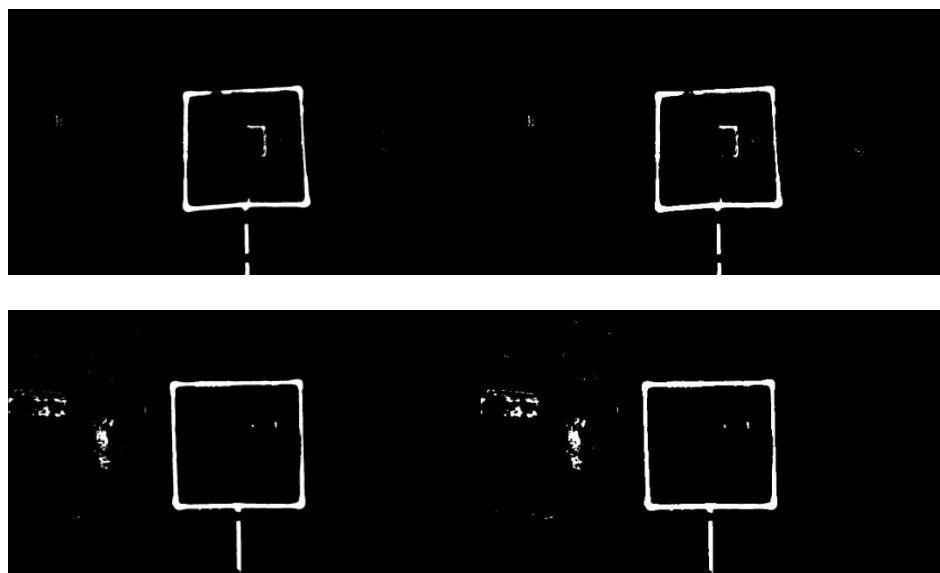
Limite	Canal H	Canal S	Canal V	Color
Inferior	25	50	100	
Superior	45	255	255	

Utilizando el rango de color de la Tabla 22, se obtiene la máscara que se muestra en la Figura 51 para diferentes frames.



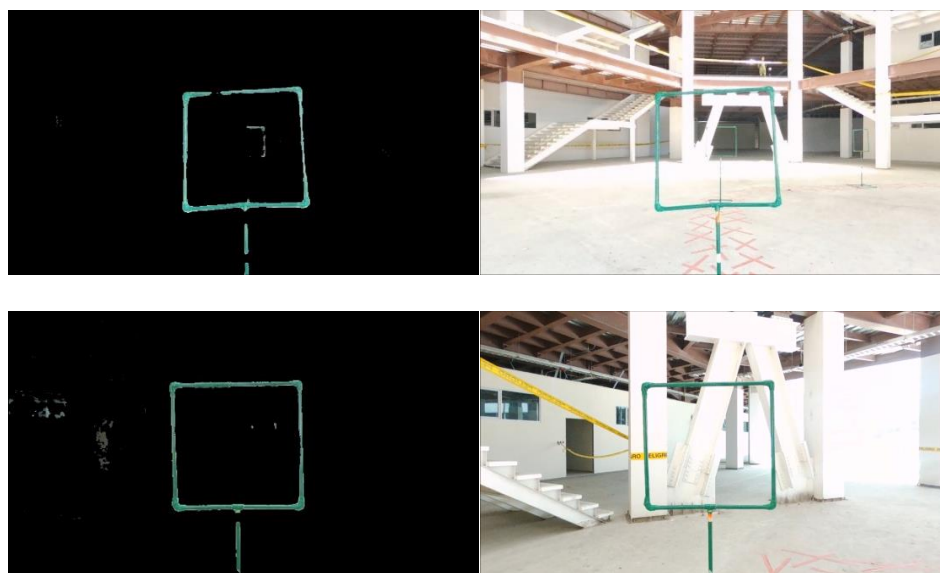
**Figura 51.** Máscaras para diferentes frames que contienen un obstáculo

La imagen de la Figura 51 corresponde a una imagen binaria, es decir tiene intensidades de 0 o 1, al tratarse de una máscara obtenida por un filtro de color se producen detecciones erróneas, estas pueden ser reducidas por operaciones morfológicas. En el caso del obstáculo, se utiliza la transformación morfológica DILATE para evitar que se produzcan cortes en la detección del obstáculo, la diferencia entre *frames* con y sin transformación morfológica DILATE se muestra en la Figura 52, a la izquierda se muestran las imágenes binarias originales y a la derecha se muestran las imágenes binarias después de realizar la transformación morfológica.



**Figura 52.** Comparación entre frames con y sin transformación morfológica

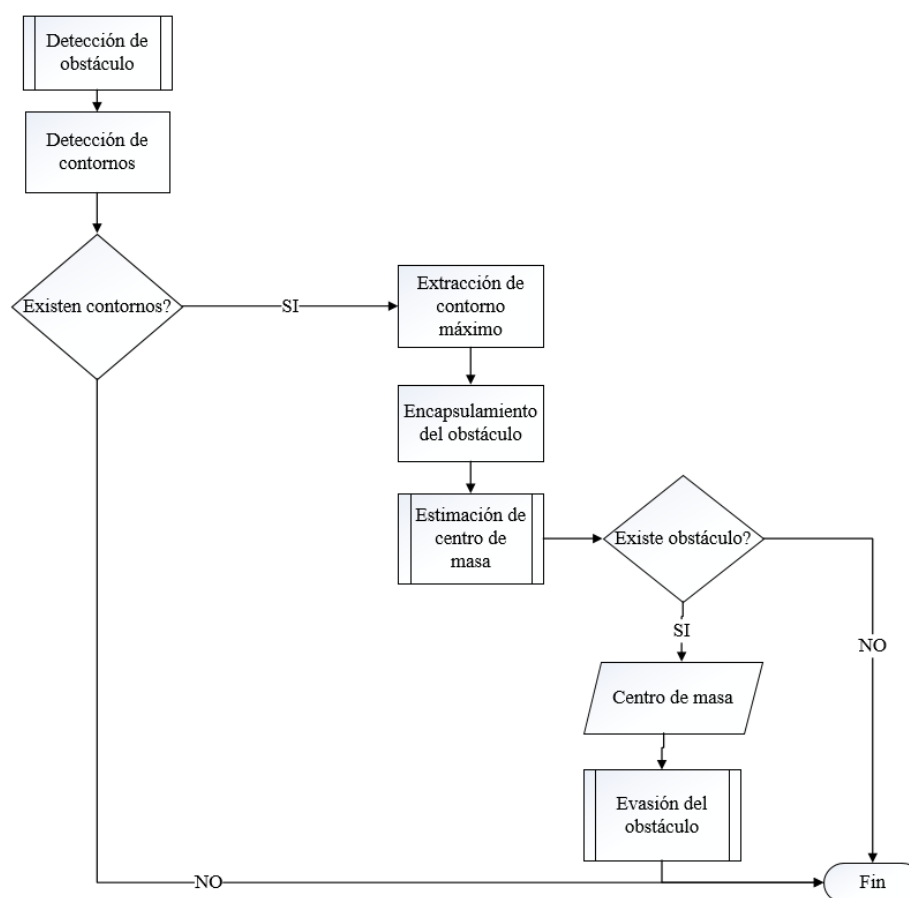
Finalmente, se realiza una operación lógica *AND* con la máscara para enmascarar la imagen original, los *frames* resultantes de esta operación se muestran en la Figura 53.



**Figura 53.** Frames 1 y 2 enmascarados

## 5.2 Detección de obstáculo

La detección de obstáculos supone un componente importante en el desarrollo de investigación, este proceso comprende la discriminación, localización y las acciones que se van a tomar en caso de detectar un obstáculo. En la Figura 54, se resume el proceso de detección de obstáculos.



*Figura 54.* Proceso de detección de obstáculos

### 5.2.1 Detección de contornos

Los contornos pueden ser entendidos como una curva que une puntos adyacentes alrededor de un límite que está definido por el color o intensidad de la región. Es una herramienta útil para el

análisis de formas, detección y reconocimiento de objetos. En este trabajo de investigación se propone una detección de obstáculos basada en contornos, estas funciones tienen una alta eficiencia y se pueden realizar las operaciones en tiempo real.

Para encontrar los contornos del frame de la Figura 53 se utiliza la función de *OpenCV* *cv2.findContours()*, la función acepta imágenes binarias o enmascaradas y realiza la extracción de los puntos que rodean a una región de la imagen. En la Figura 55 se muestran los frames con los contornos marcados de color verde.



**Figura 55.** Detección de contornos

Una vez detectados los contornos en un frame, se toma el contorno con la mayor área para discriminar obstáculos que se encuentren detrás del primer obstáculo o detecciones erróneas de objetos que se encuentren en el mismo rango de color del obstáculo. Tomando en cuenta al contorno de mayor área, se utiliza la función de *OpenCV* *cv2.boundingRect()*, que encuentra un cuadrilátero que rodea al contorno en sus puntos máximos, en la Figura 56 se muestran los frames con el encapsulamiento del obstáculo.





**Figura 56.** Encapsulamiento del obstáculo

## 5.2.2 Estimación del centro de masa del obstáculo

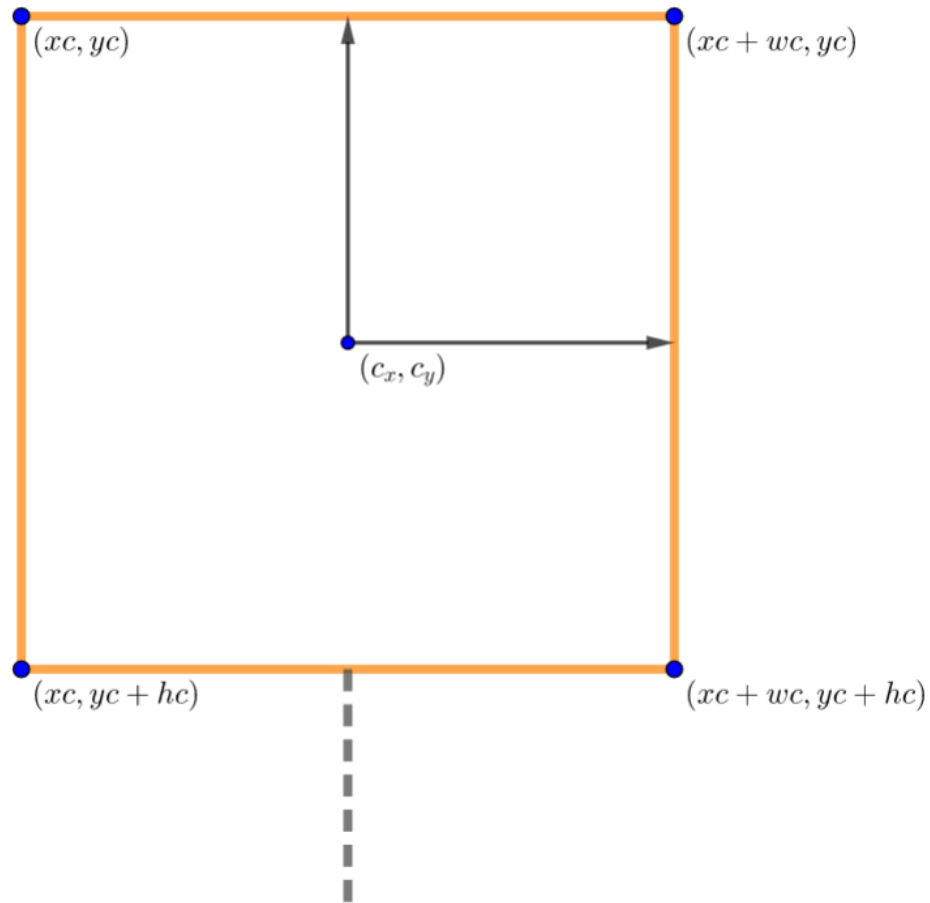
Para la estimación del centro de masa del obstáculo a través de las imágenes monoculares que transmite el *UAV*, se han tomado en cuenta dos métodos:

- Obtención del centro del recuadro de encapsulamiento del obstáculo
- Filtro de Kalman, teniendo como medida el centro del recuadro de encapsulamiento

### 5.2.2.1 Centro del recuadro de encapsulamiento del obstáculo

Una vez que se ha encapsulado el obstáculo, de la función *cv2.boundingRect()* se obtienen los siguientes parámetros:

- *xc* que corresponde a la coordenada en *x* del recuadro de encapsulamiento
- *yc* que corresponde a la coordenada en *y* del recuadro de encapsulamiento
- *wc* que corresponde a la anchura del recuadro de encapsulamiento
- *hc* que corresponde a la altura del recuadro de encapsulamiento



**Figura 57.** Centro del recuadro de encapsulamiento

Con los parámetros mencionados se obtiene el centro del recuadro para el obstáculo, en la ecuación se representa el valor del centro de masa para  $x$  e  $y$ :

$$c_x = x_c + \frac{w_c}{2} \quad (41)$$

$$c_y = y_c + \frac{h_c}{2} \quad (42)$$

### 5.2.2.2 Estimación del centro del recuadro a través de un Filtro de Kalman

Para la estimación del centro del recuadro utilizando un Filtro de Kalman se tiene como premisa que las mediciones vienen dadas como el centro del recuadro obtenido en el anterior apartado.

Al igual que en el apartado de estimación del punto de referencia para el seguimiento de la trayectoria a través de un Filtro de Kalman, se toma en cuenta un error en la medición de posición correspondiente a la acumulación promedio de error en el desplazamiento en el eje de las abscisas en una secuencia de video en la que el dron se encuentra en el estado de “permanecer inmóvil” y el error en la medición del movimiento como la diferencia absoluta frame a frame de la misma secuencia de video, estos valores corresponden a 0.15957 y 0.13216, respectivamente.

Las mediciones de la posición del obstáculo corresponden a un arreglo de  $N$  mediciones del centro del recuadro de encapsulamiento de este, siendo  $N$  el número mínimo de muestras que se consideran un movimiento, esto se representa en la ecuación ( 32), de igual manera para medir el movimiento se considera un arreglo de  $N-1$  muestras que corresponden a la derivada numérica del arreglo de mediciones de posición, esto se representa en la ecuación ( 33).

Seguido, se actualiza la posición del obstáculo y el error en la estimación usando las ecuaciones ( 34) y ( 35) respectivamente, finalmente, se predice la posición y el error empleando las ecuaciones ( 36) y ( 37).

La medición a través del filtro de Kalman se utiliza cuando existe una medición errónea, esto quiere decir, si el contorno se ha entrecortado o si el cambio de medición es abrupto.

Para comparar los resultados en la estimación de la posición del obstáculo, se tomaron seis secuencias de video en las siguientes condiciones:

- Condición de iluminación: ~5000 lux
- Altura de detección: 1.3 metros
- Distancia inicial con respecto al obstáculo: 2 metros

- Ángulo de inclinación (*tilt*): 0°
- Estado del dron: Se acerca al obstáculo

Luego, se marca a mano la posición real del obstáculo para ser tomado como valor referencial al momento de obtener el error medio cuadrático, resultados que se muestran en la Tabla 23.

**Tabla 23**

*Error medio cuadrático para la estimación de posición del obstáculo*

	Secuencias de video					
	1	2	3	4	5	6
<b>Centro de masa</b>	12.09	10.78	2.51	49.43	12.33	15.45
<b>Filtro de Kalman</b>	4.68	5.05	1.97	20.66	10.45	9.86

Además del error medio cuadrático, se considera como métrica a la varianza siendo esta un indicador de la dispersión de las mediciones del centro del obstáculo, los resultados se muestran en la Tabla 24.

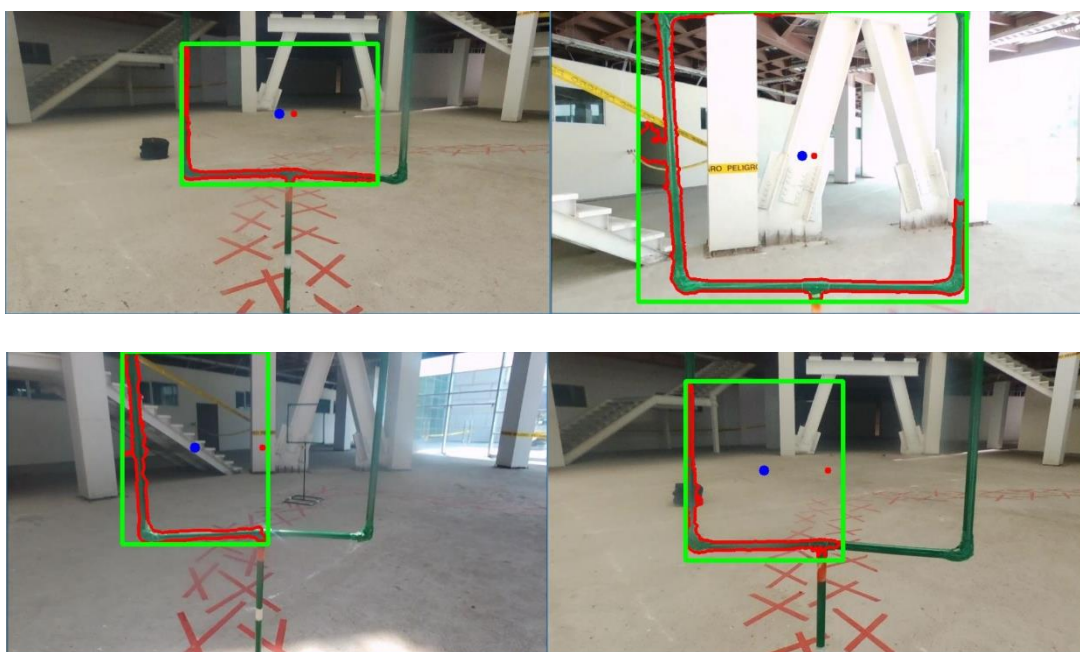
**Tabla 24**

*Varianza de las mediciones de la posición del obstáculo*

	Secuencias de video					
	1	2	3	4	5	6
<b>Centro de masa</b>	320.08	387.32	250.94	3653.22	273.17	3905.87
<b>Filtro de Kalman</b>	151.61	271.85	245.62	1568.19	203.55	3631.62

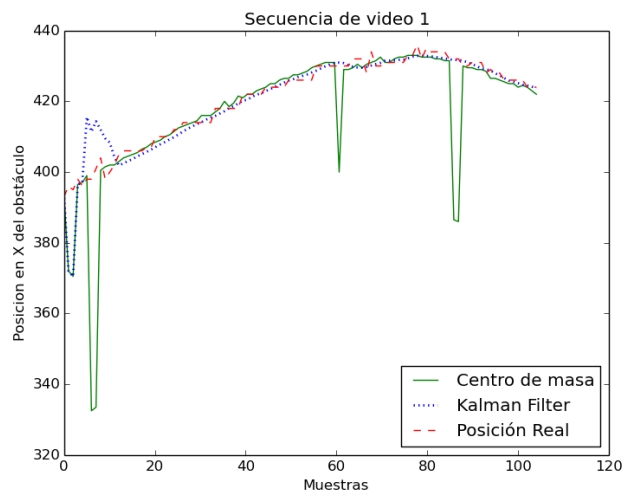
De la Tabla 23 se puede concluir que la implementación de un filtro de Kalman para estimar la posición del obstáculo mientras que el dron se está aproximando al mismo, reduce el error medio cuadrático hasta en 8.32 % que se traduce a aproximadamente 72 pixeles de reducción de error.

De la Tabla 24 se puede concluir que el filtro de Kalman añade la propiedad de suavizado en la estimación de la posición del obstáculo, ya que reduce la varianza de las mediciones en un 50%, lo que significaría que la aproximación del dron hacia el obstáculo se va a realizar de manera paulatina. Lo antes mencionado se puede evidenciar en *frames* específicos de las secuencias de video en la Figura 58, se muestran varios frames en los que se realiza una estimación errónea cuando no se utiliza un filtro de Kalman, el punto azul corresponde a detecciones erróneas y el punto rojo corresponde a la detección utilizando un filtro de Kalman.

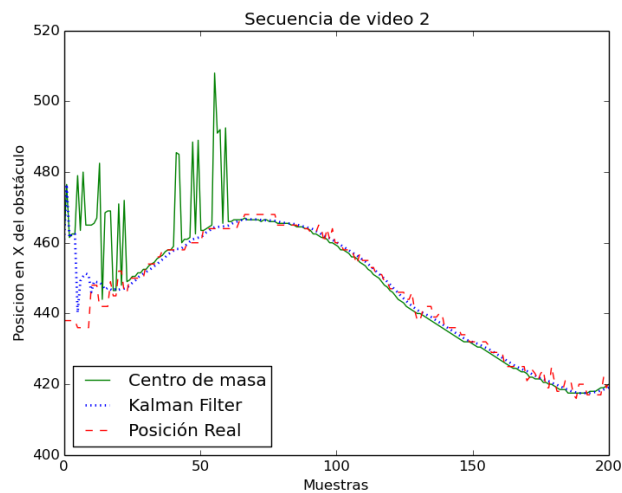


**Figura 58.** Detecciones erróneas del centro del obstáculo por cambios de iluminación

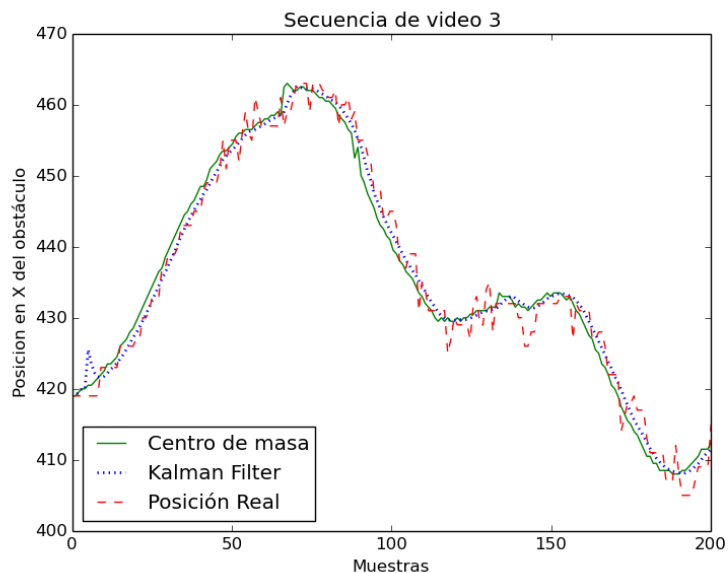
A su vez, dichas estimaciones se muestran a manera de picos en las gráficas de la Figura 59 a la Figura 64, que resultarían en movimientos abruptos teniendo la posibilidad de perder el obstáculo en la vista frontal debido a las limitaciones en el campo de visión del dron.



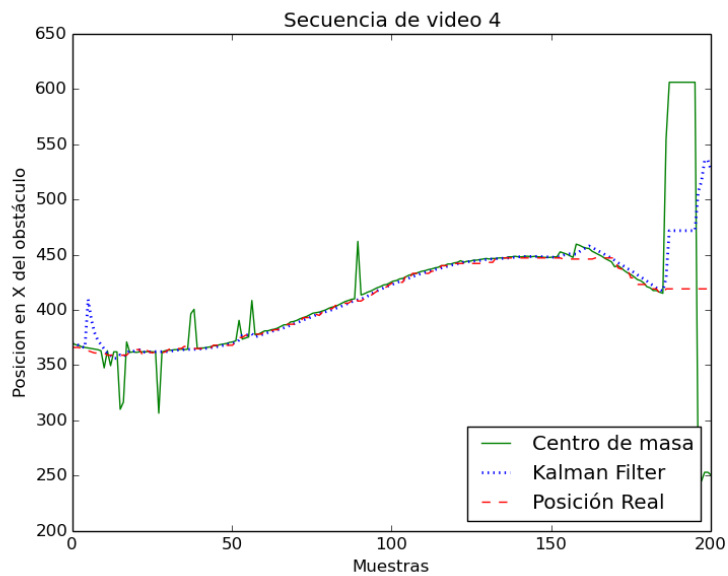
**Figura 59.** Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 1



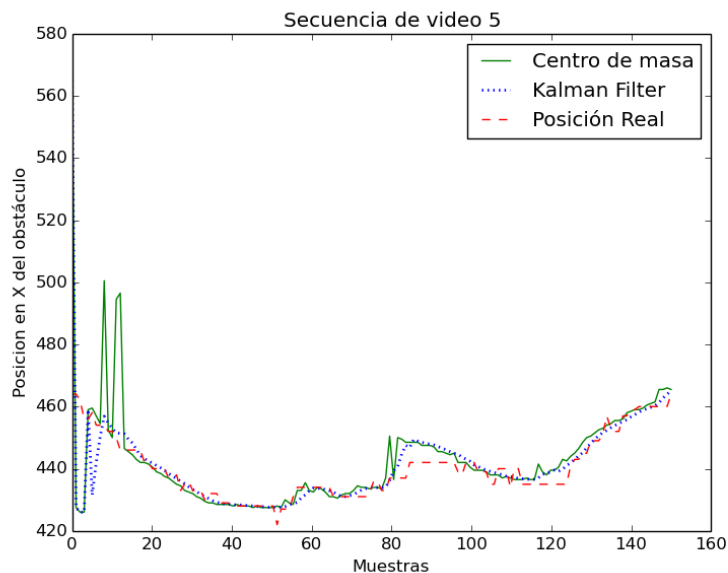
**Figura 60.** Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 2



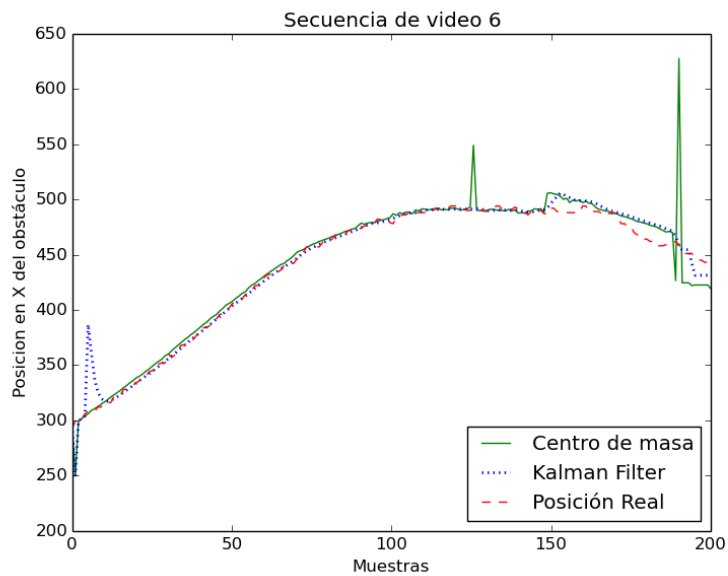
**Figura 61.** Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 3



**Figura 62.** Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 4



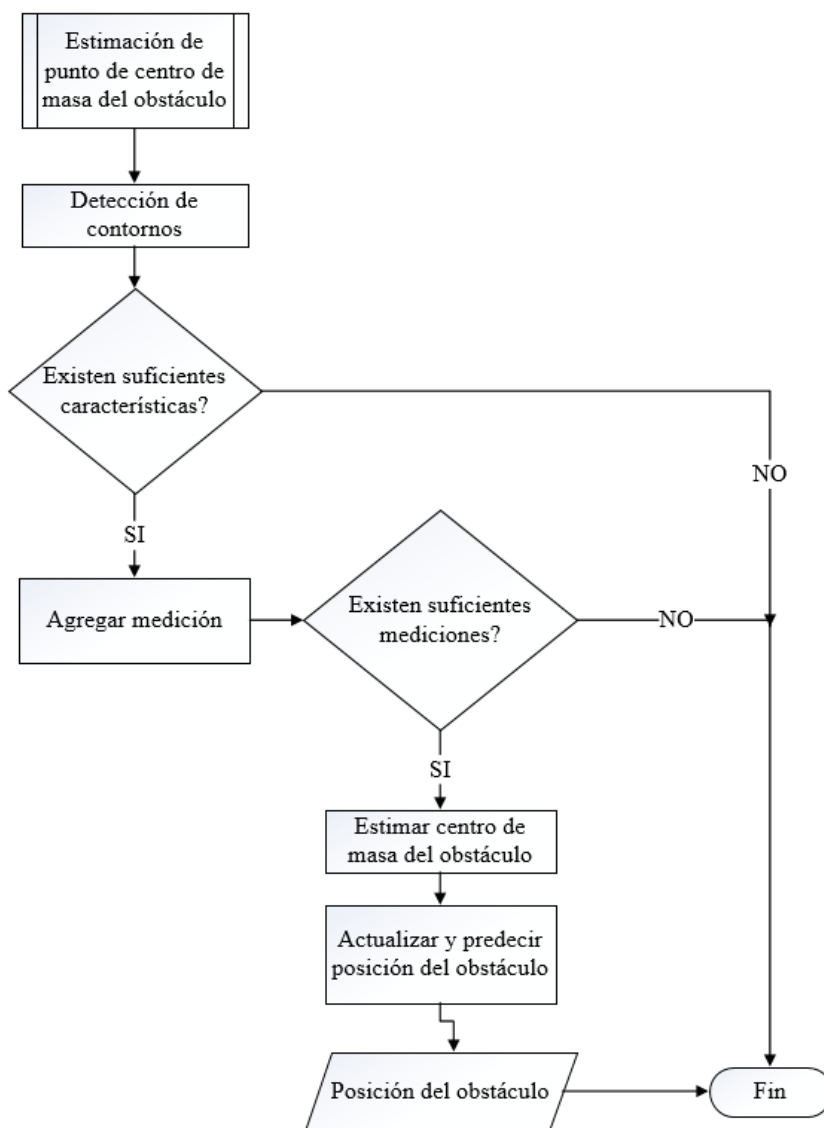
**Figura 63.** Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 5



**Figura 64.** Comparativa de seguimiento de obstáculo entre el centro del recuadro y el mismo a través de un Filtro de Kalman para la secuencia de video 6



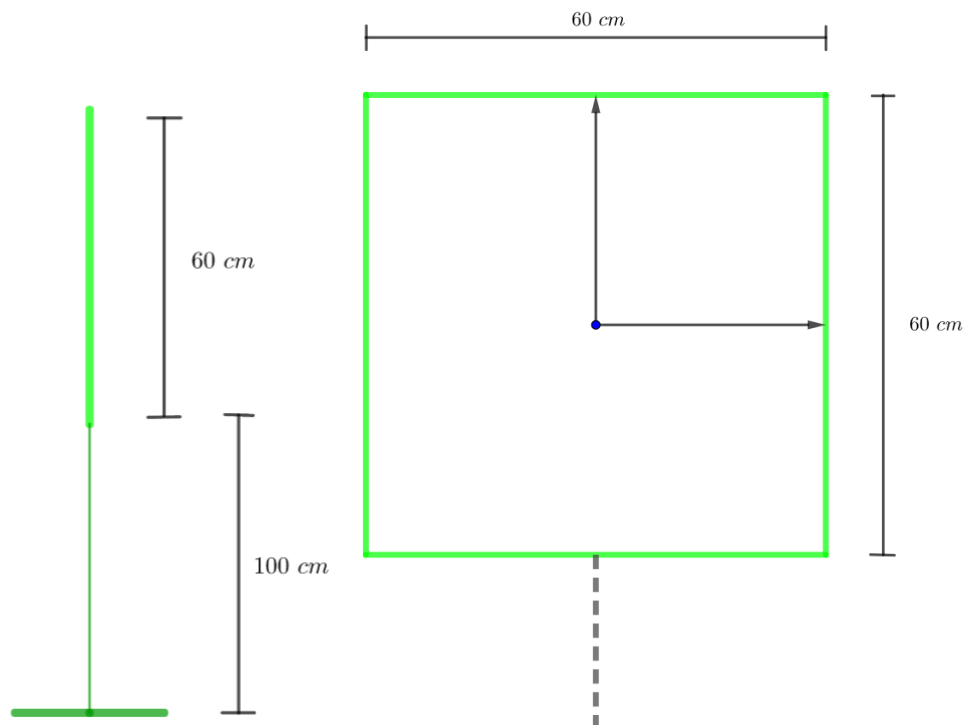
Por lo tanto, para la estimación del centro de masa del obstáculo se utilizan como mediciones el centro del recuadro *frame a frame* en conjunto con el filtro de Kalman, reduciendo así la posibilidad de colisión ante el obstáculo por una estimación errónea. En la Figura 65, se resume el proceso de estimación del centro de masa del obstáculo.



**Figura 65.** Proceso de estimación de centro de masa del obstáculo

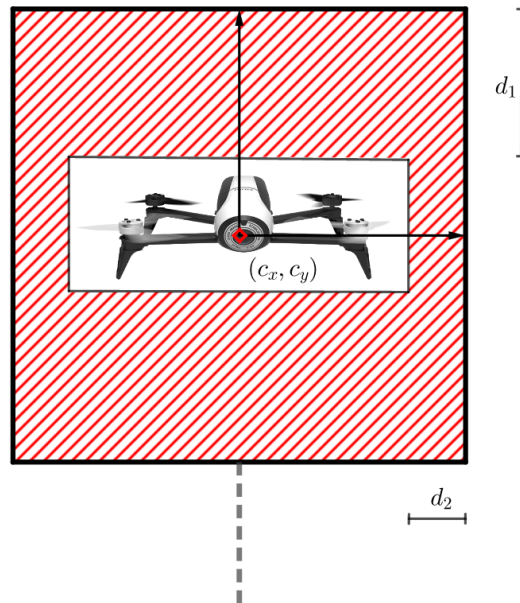
### 5.2.3 Evasión del obstáculo

La evasión del obstáculo va a estar definida como la acción que toma el dron de manera autónoma para atravesar el obstáculo, no se considera una evasión si el dron pasa por arriba, debajo o rodea el obstáculo. En la Figura 66, se muestran las dimensiones definidas para el obstáculo.



**Figura 66.** Dimensiones del obstáculo

Teniendo como premisa la definición de evasión de obstáculos para el proyecto de investigación, el dron deberá atravesar el obstáculo con un grado de precisión alto, los umbrales de decisión que tiene permitido el dron corresponden a  $d_1 = 25\text{cm}$  en el eje de las ordenada y a  $d_2 = 7.4\text{ cm}$  en el eje de las abscisas (Ver Figura 67).



**Figura 67.** Distancias umbral para atravesar el obstáculo

Si se considera que al momento de tomar la decisión de atravesar el obstáculo la imagen que recibe el dron corresponde a una imagen en la cual el obstáculo ocupa totalmente la altura de la imagen que corresponde a aproximadamente 480 píxeles y considerando que la medida real del obstáculo en altura es de 60 cm, podemos obtener los umbrales de decisión de evasión de obstáculo como se muestra en las ecuaciones ( 43) y ( 44):

$$umbral_{x\ px} = \frac{d_2 \times ancho\ obstáculo_{px}}{ancho\ obstáculo_{x\ real}} \quad (43)$$

$$umbral_{y\ px} = \frac{d_1 \times altura\ obstáculo_{px}}{altura\ obstáculo_{y\ real}} \quad (44)$$

Los valores correspondientes al  $umbral_{x\ px}$  y al  $umbral_{y\ px}$  son de 40 píxeles y 200 píxeles respectivamente.

Ahora que se ha encontrado el umbral para atravesar el obstáculo, es necesario definir el umbral de área, ya que como se mencionó en la anterior definición, el obstáculo debe ocupar la totalidad de la altura de la imagen en píxeles que corresponde a para que se cumplan las condiciones que permiten al dron que atraviese el obstáculo y considerando que el obstáculo corresponde a un cuadrilátero con medidas iguales en sus cuatro lados el área umbral viene dada por la ecuación (45):

$$\text{área}_{\text{umbral}} = (\text{altura}_{\text{obstáculo}_{px}})^2 \quad (45)$$

Por lo tanto para que se cumplan las condiciones de  $\text{umbral}_{x_{px}}$  de 40 píxeles y de  $\text{umbral}_{y_{px}}$  de 200 píxeles el área del obstáculo debe corresponder a 230400 píxeles cuadrados.

Cuando se han cumplido las condiciones de área umbral, umbral en x y umbral en y, el dron pasa el estado de atravesar el obstáculo, este permanecerá en este estado hasta que haya atravesado en su totalidad el obstáculo, para determinar el tiempo que al dron le toma para atravesar el obstáculo, se debe definir la distancia que existe entre este y el obstáculo.

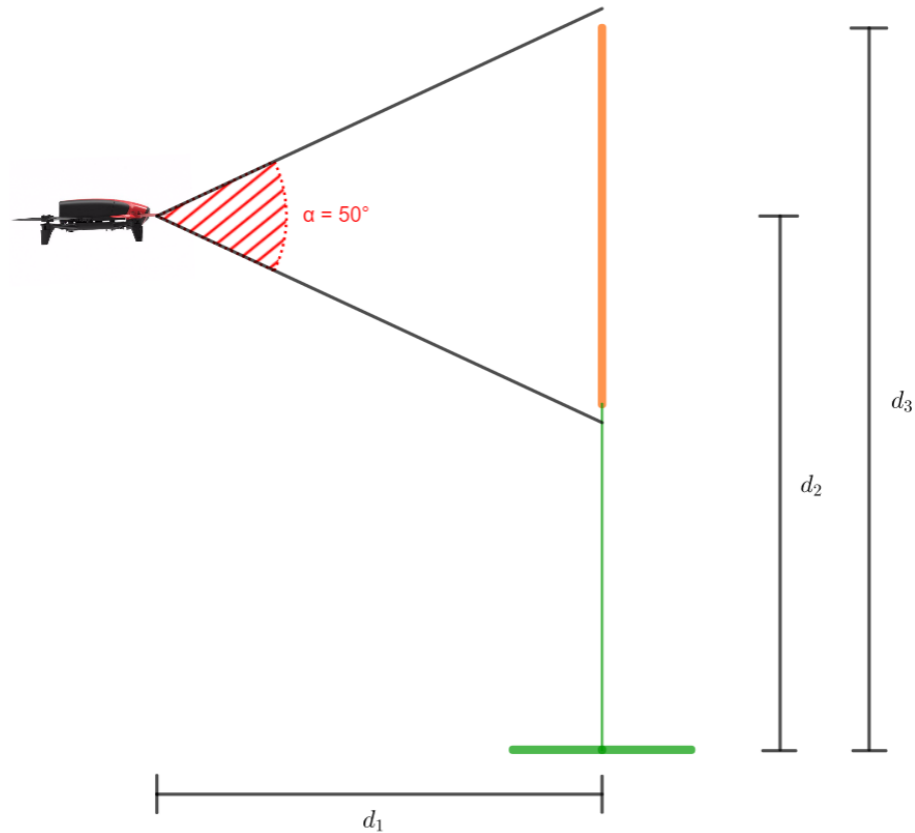
Considerando que el área umbral se ha cumplido y que la posición de la cámara virtual en tilt es de  $0^\circ$  como se muestra en la Figura 68, la distancia está definida por la ecuación (46):

$$d_1 = \frac{d_3 - d_2}{\tan\left(\frac{\alpha}{2}\right)} \quad (46)$$

En donde,

- $d_3$  corresponde a la altura total del obstáculo medida desde el piso (1.6 m)
- $d_2$  corresponde a la altura del obstáculo medida desde el piso hasta el centro (1.3 m)

- $\alpha$  corresponde al ángulo de visión total del dron en *tilt*



**Figura 68.** Distancia entre el dron y el obstáculo cuando se ha alcanzado el área umbral

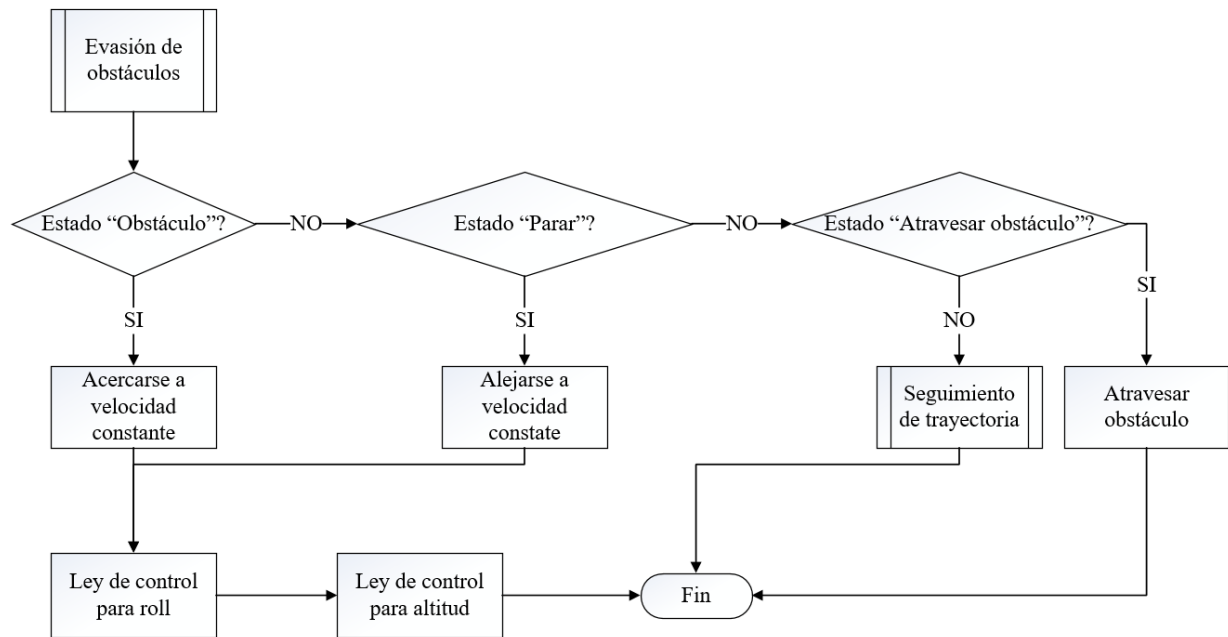
Por lo tanto, la distancia existente entre el dron y el obstáculo corresponde a una distancia de 64 cm aproximadamente.

Para el cálculo del tiempo que al dron le toma para atravesar el obstáculo se considera el doble de la distancia antes calculada ya que con esto se asegura que el dron atraviesa el obstáculo y se adelanta 64 cm, asegurando que este atraviese el obstáculo totalmente, en consecuencia, considerando que la velocidad a la que atraviesa el obstáculo corresponde a 1 m/s, se obtiene que el dron necesita permanecer en el estado de *atravesar obstáculo* durante 3.125 segundos.

### 5.2.3.1 Estados de evasión de obstáculos

El centro de masa del obstáculo puede perderse cuando el dron se encuentre a la distancia umbral para tomar la decisión de atravesar el obstáculo, no obstante, se establecen estados en los cuales el dron pueda realizar ajustes antes de tomar la decisión de atravesar el obstáculo, estos se encuentran listados a continuación:

- **Obstáculo:** Estado en el cual el dron ha detectado el obstáculo, pero aún no se encuentra en la distancia umbral. En este estado el dron se aproxima a una velocidad constante hacia el obstáculo.
- **Parar:** Estado en el cual el dron ha detectado el obstáculo y este se encuentra en la distancia umbral, sin embargo, el error de posicionamiento del dron en el eje x es mayor al error umbral para atravesar el obstáculo. En este estado el dron se aleja del obstáculo a la ve que corrige el error.
- **Atravesar obstáculo:** Estado en el cual el dron ha detectado el obstáculo y este cumple dos condiciones: 1) se encuentra en la distancia umbral y 2) el error de posicionamiento en eje x es menor al error umbral.



**Figura 69.** Algoritmo de evasión de obstáculos

### 5.2.3.2 Diseño del controlador

En la Figura 69 se puede apreciar que para que el dron se posicione dentro del error umbral necesario para atravesar el obstáculo, se necesita un controlador que posicione al dron en dicha posición, de forma similar al diseño del controlador para el seguimiento de la trayectoria se ha seleccionado un controlador *PID* para el control del posicionamiento siguiendo como referencia el centro de masa.

A diferencia del seguimiento de la trayectoria en la cual se imita a un vehículo de configuración diferencial con el movimiento angular alrededor de eje Z lo que representa un grado de libertad, para posicionarse frente al centro de masa del obstáculo se necesita realizar un movimiento a lo largo del eje Z y eje Y, lo que significa que hacen falta dos controladores uno para controlar la altitud (eje Z) y otro para generar movimientos de izquierda a derecha (*Roll*).

Al igual que en el seguimiento de trayectoria, se necesita el modelo matemático que describa el movimiento del dron cuando se generan movimientos en altitud y a lo largo del eje Y. En el trabajo de (Salcedo, 2018), se modelan los movimientos del dron cuando este tiene una vista cenital (imagen del plano sobre el cual se encuentra sobrevolando el dron), sin embargo, es posible obtener el modelo matemático del movimiento si la cámara del dron tiene una vista frontal (ángulo de inclinación *tilt* de 0 grados), tomando en cuenta que:

- Los movimientos producidos en altitud en el eje Z, corresponderán a movimientos generados en la imagen a lo largo del eje X.
- Los movimientos producidos en el eje Y, corresponderán a movimientos generados en la imagen a lo largo del eje Y.

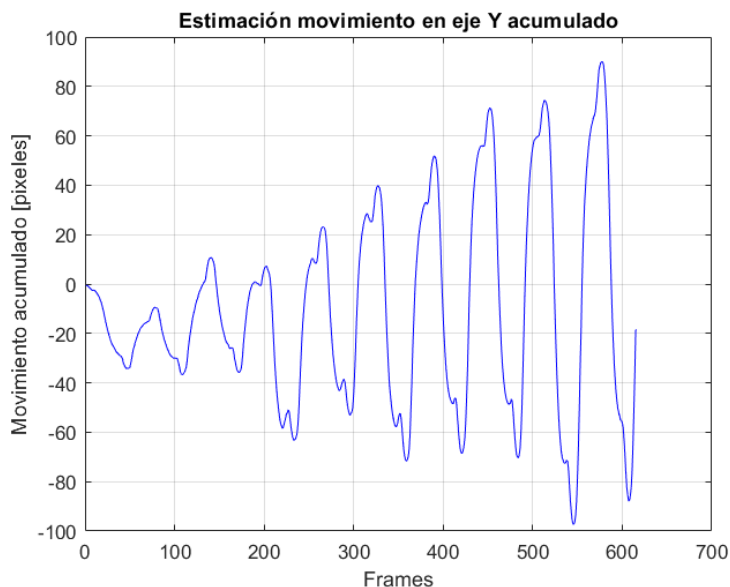
Para llevar a cabo esto, se publica en el tópico `/bebop/cmd_vel` (mientras el UAV se encuentra volando) un tren de impulsos como el que se muestra en la Figura 46, específicamente en el parámetro *linear.z* y *linear.y* que hacen referencia al movimiento lineal a lo largo del eje z y a lo largo del eje y, respectivamente. Además, se graban las secuencias de video producidas por el tren de impulsos para su posterior procesamiento.

Las condiciones en las cuales se grabaron las secuencias de video son las siguientes:

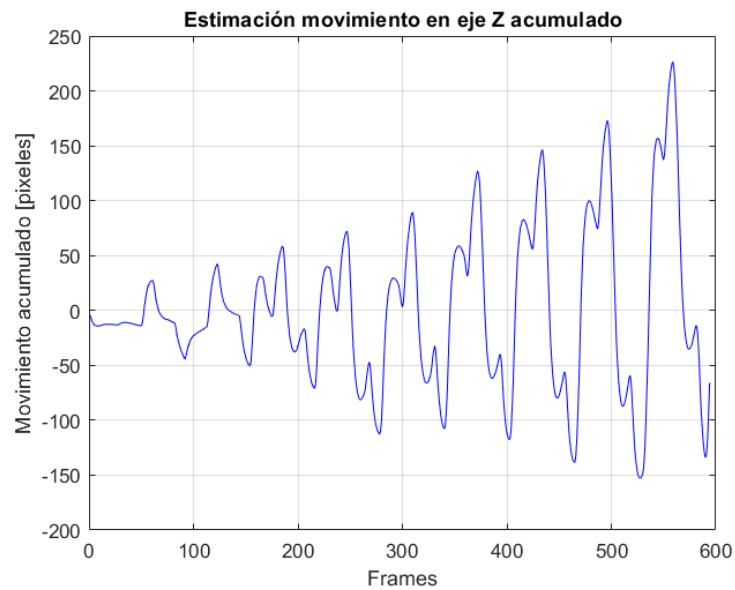
- Condición de iluminación: ~2000 lux
- Altura de detección: 1 metro
- Ángulo de inclinación (*tilt*): 0°
- Estados del dron: 1) se mueve a lo largo del eje Z, 2) se mueve a lo largo del eje Y



Una vez obtenida la grabación de video, se lleva a cabo el proceso de estimación de movimiento a través de una transformación *Affine2D* (Similitud no reflectiva). Para lograr esto se obtienen los puntos de interés del *frame*  $I_{t-1}$  y se comparan con los puntos de interés del *frame*  $I_t$  y a través de la función *cv2.estimateAffinePartial2D* de *OpenCV* se obtiene la matriz de transformación de la ecuación ( 38), de la matriz de transformación se obtienen los parámetros  $t_x$  y  $t_y$  que corresponden al movimiento a lo largo del eje Z y a lo largo del eje Y, respectivamente. Luego, se realiza la estimación de la traslación del dron en función de la acumulación del movimiento. En la Figura 70 y Figura 71, se muestran los resultado de la acumulación del parámetro  $t_y$  y  $t_x$  para el tren de impulsos de la Figura 46.



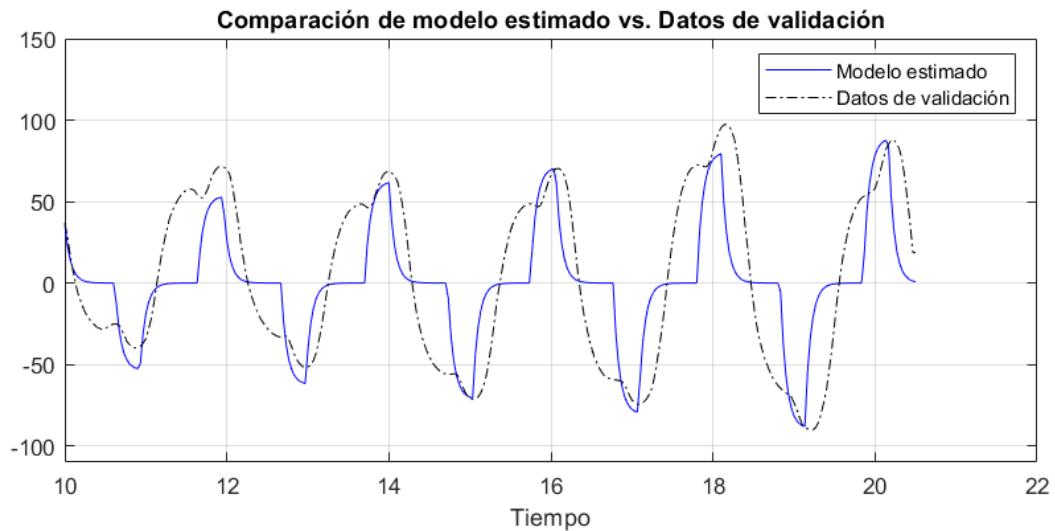
**Figura 70.** Estimación de movimiento acumulado en el eje Y



**Figura 71.** Estimación de movimiento acumulado en el eje Z

Con los datos obtenidos de la estimación de movimiento a lo largo del eje Y, se realiza la identificación del modelo de la planta, para esto, se utiliza el *Toolkit* de *Matlab systemIdentification* y se realiza la estimación de una planta de proceso de primer orden correspondiente a la función de transferencia de la ecuación ( 39).

Con los parámetros correspondientes a  $K_p = 89.346$  y  $T_p = 0.07034$ , en la Figura 72, se observa el modelo estimado vs los datos de validación correspondientes a la estimación de movimiento acumulado de la Figura 70.

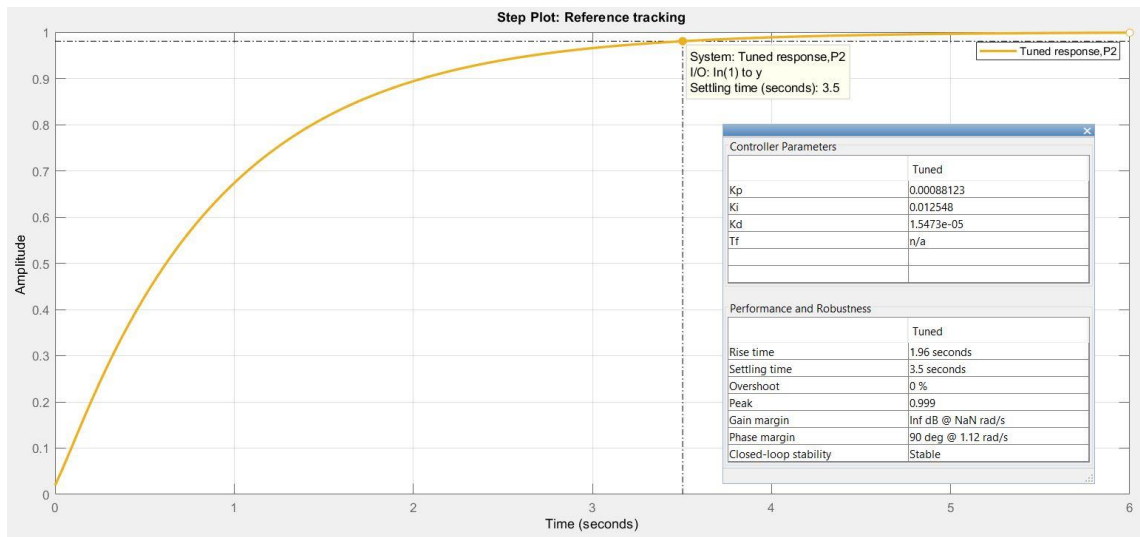


**Figura 72.** Comparación de modelo estimado para Roll vs. Datos de validación del movimiento a lo largo del eje Y acumulado

A continuación, utilizando el *Toolkit* de *Matlab pidTuner*, se sintoniza un controlador *PID*, con las siguientes especificaciones de *performance*.

$$M_p = 0\%, \quad t_s = 3.5 \quad (47)$$

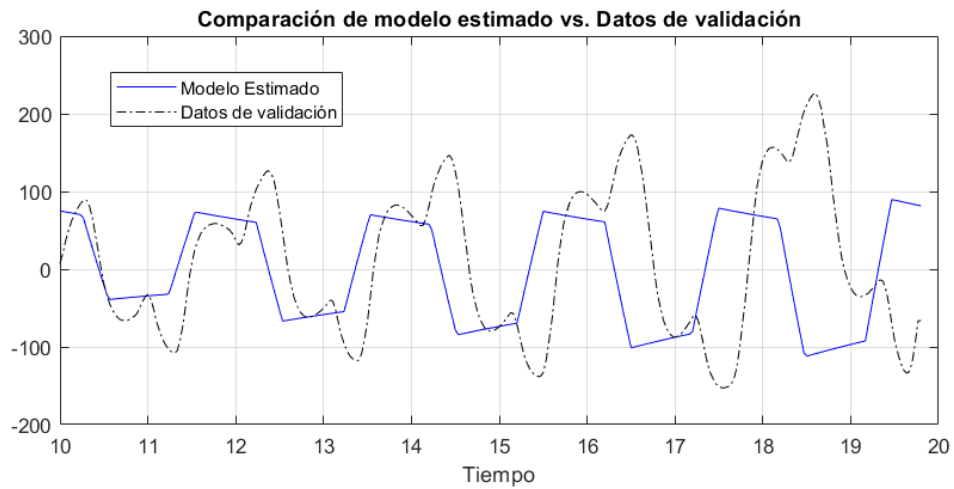
Mediante la sintonización del controlador *PID* en el *Toolkit pidTuner*, se obtienen las ganancias correspondientes a  $K_p = 0.00088123$ ,  $K_i = 0.012548$  y  $K_d = 1.5473e-05$ . La respuesta en estado estacionario del conjunto Planta-Controlador, se muestra en la Figura 73.



**Figura 73.** Respuesta al escalón unitario del conjunto Planta-Controlador para el movimiento lineal a lo largo del eje Y (Roll)

De la misma manera que para el movimiento lineal a lo largo del eje y, se realiza la estimación de una planta de proceso de primer orden correspondiente a la función de transferencia de la ecuación ( 39) para el movimiento lineal a lo largo del eje Z correspondiente a la traslación en x para la imagen.

Con los parámetros correspondientes a  $K_p = 2107$  y  $T_p = 3.4718$ , en la Figura 75, se observa el modelo estimado vs los datos de validación correspondientes a la estimación de movimiento acumulado de la Figura 71.

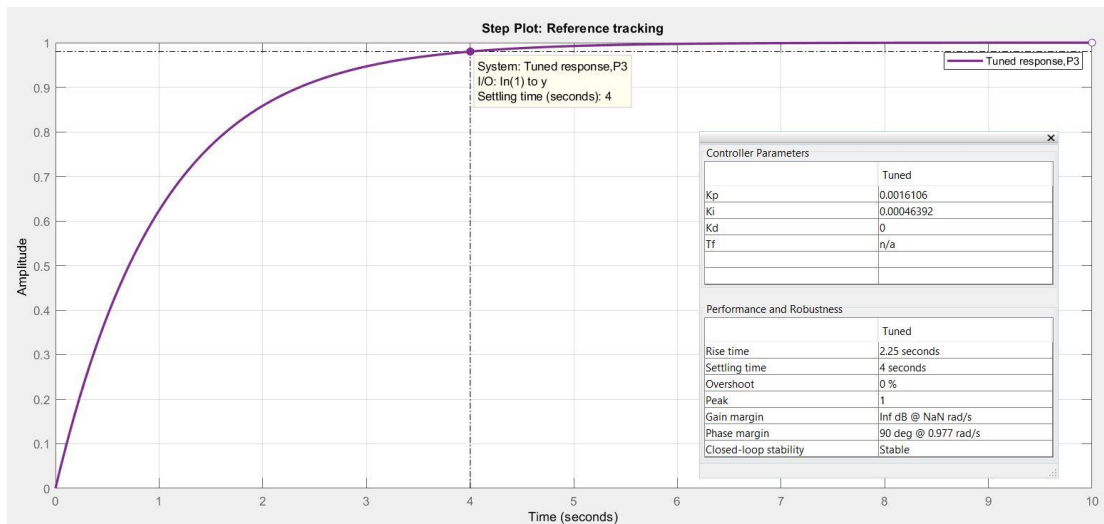


**Figura 74.** Comparación de modelo estimado para altitud vs. Datos de validación del movimiento a lo largo del eje Z acumulado

A continuación, utilizando el *Toolkit* de *Matlab pidTuner*, se sintoniza un controlador *PID*, con las siguientes especificaciones de *performance*.

$$M_p = 0\%, \quad t_s = 4 \quad (48)$$

Mediante la sintonización del controlador *PID* en el *Toolkit pidTuner*, se obtienen las ganancias correspondientes a  $K_p = 0.0016106$ ,  $K_i = 0.00046392$  y  $K_d = 0$ . La respuesta en estado estacionario del conjunto Planta-Controlador, se muestra en la Figura 75.



**Figura 75.** Respuesta al escalón unitario del conjunto Planta-Controlador para el movimiento lineal a lo largo del eje Z (Altitud)

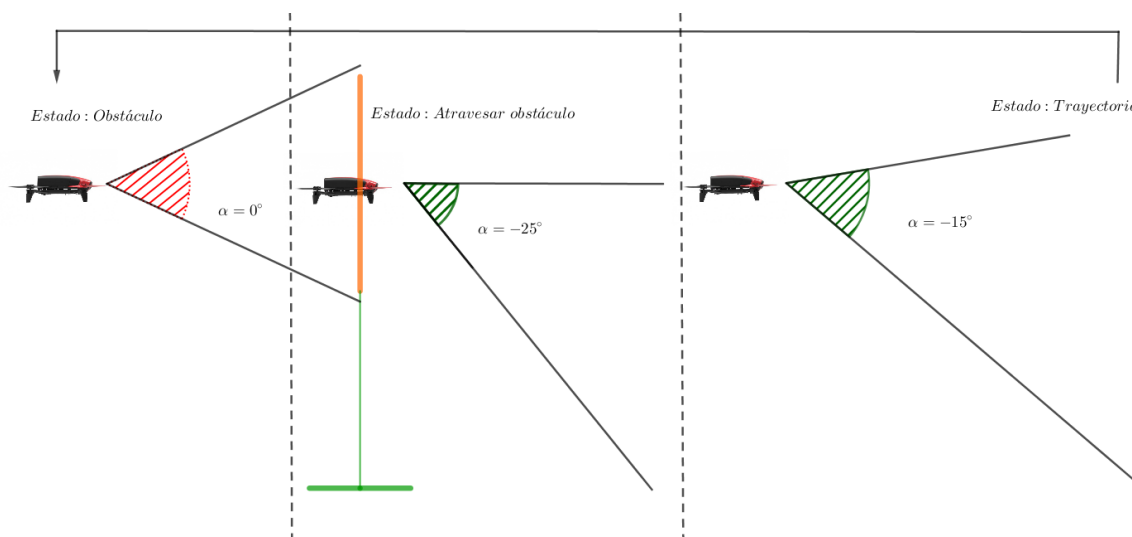
### 5.2.3.3 Evasión de obstáculos y seguimiento de trayectoria

Por último, cuando ya se ha detectado la trayectoria y el obstáculo en un mismo frame, se establece la prioridad de pasar el obstáculo, no obstante, el cálculo del error del punto de seguimiento de trayectoria se sigue realizando sin ejercer ninguna acción de control para seguirla. Además, es importante recalcar que si se pierde de vista la trayectoria se seguirá guardando el último punto de seguimiento de la misma para evitar que el dron pierda la trayectoria.

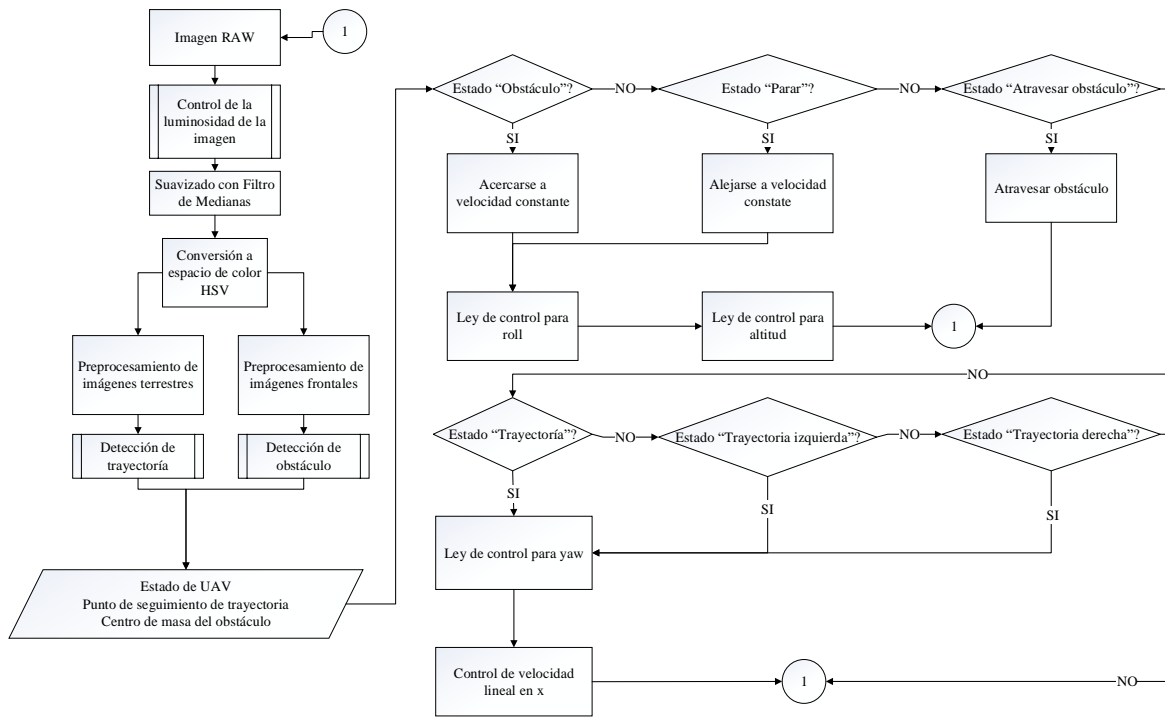
Para evitar el uso excesivo de los estados en el cual el dron pierde la trayectoria, se realiza un artificio con el movimiento de la cámara virtual. El artificio se puede apreciar en la Figura 76, en la cual se muestra la transición correspondiente entre el obstáculo y la trayectoria.

En primera instancia si el dron detecta un obstáculo, se realizan las acciones de control del diagrama de flujo de la Figura 69 y cuando se pasa del estado *Obstáculo* al estado *Atravesar obstáculo* se realiza un movimiento de la cámara virtual que corresponde a  $-25^\circ$  de inclinación (*tilt*), permitiendo así al dron tener a la trayectoria en constante observación y una vez que el dron sale

del estado *Atravesar obstáculo* se realiza un movimiento de la cámara virtual correspondiente a  $-15^\circ$  con el fin de observar tanto a la trayectoria como a los posibles obstáculos que se encuentren al frente del dron. El algoritmo que comprende los estados de detección de trayectoria y detección de obstáculos se muestra en la Figura 77.



**Figura 76.** Transición Obstáculo-Trayectoria



**Figura 77.** Algoritmo de navegación autónoma para detección y seguimiento de trayectoria, y detección y evasión de obstáculos



## CAPÍTULO VI

### 6. EXPERIMENTOS Y ANÁLISIS DE RESULTADOS

Para la evaluación de desempeño del sistema de detección y seguimiento de trayectoria, y detección y evasión de obstáculos, se propone una contrastación de las acciones de control entre un tele-operador y el algoritmo de navegación autónoma.

Se tomarán en cuenta dos escenarios, en primer lugar, se realizará el seguimiento de la trayectoria sin presencia de obstáculos, en este escenario se evaluarán los siguientes parámetros:

- Tiempo de seguimiento de trayectoria
- Error medio cuadrático de seguimiento de trayectoria

En segundo lugar, se realizarán las pruebas experimentales en el circuito construido, que consiste de una trayectoria establecida por marcas superficiales que se pueden distinguir del terreno en presencia de obstáculos que se encuentran sobre la trayectoria. En este escenario se evaluarán los siguientes parámetros:

- Seguimiento de trayectoria en presencia de obstáculos
- Tiempo de ejecución del circuito
- Tiempo de evasión de obstáculos
- Observaciones en tercera persona (exclusivo para tele-operadores) (Aguilar, Cobeña, Rodríguez, Salcedo, & Collaguazo, 2018)

Todas las pruebas y escenarios antes descritos se realizarán en *hall* central del CENTRO DE INVESTIGACIÓN de la UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE, en esta ubicación se deben en cuenta las perturbaciones externas, definidas como:

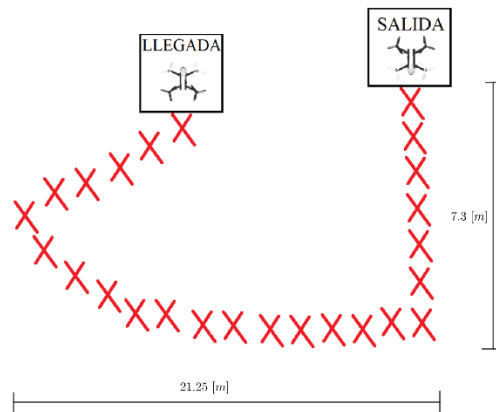
- Ráfagas de viento: De moderadas a altas
- Cambio de iluminación: Alto, aproximadamente de 1000 lux a 5000 lux

Además, se toman en cuenta tres grados de experticia para las pruebas experimentales con los tele-operadores, considerados como:

- Experto: Tele-operador con experiencia en competencias de nivel internacional de tele operación de drones de destreza. Dos o más años de experticia.
- Medio: Tele-operador con experiencia en control y operación de drones por afición. Seis meses a dos años de experticia.
- Novato: Tele-operador con menos de seis meses de experticia.

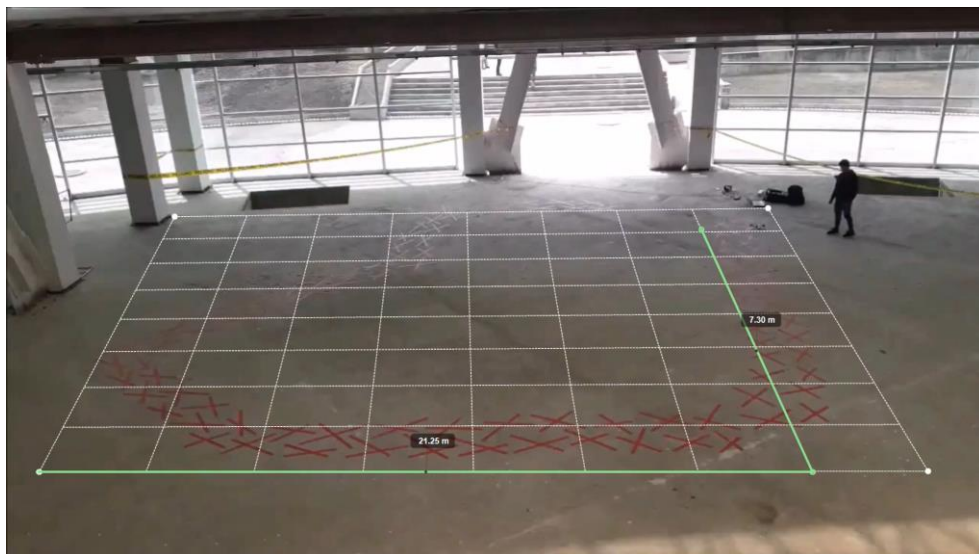
### **6.1 Pruebas experimentales de seguimiento de trayectoria**

La trayectoria que se ha definido para el seguimiento se evidencia en la Figura 78, la cual tiene como dimensiones 7.3 metros de ancho y 21.25 metros de largo, la misma que ha sido construida con marcas de superficie de los colores especificados en la Tabla 16, de manera que tenga contraste con el color del piso de la construcción.



**Figura 78.** Trayectoria de pista de carreras

En adición, en la Figura 79 se muestra una vista cenital de la trayectoria, desde esta perspectiva se realizarán los análisis de seguimiento de trayectoria. Es necesario recalcar que para el análisis de trayectoria se necesita tener una vista cenital de manera que se pueda realizar el seguimiento de dron. Para llevar a cabo esto, se utilizará el software libre *Kinovea* el cual es útil para realizar análisis cinemáticos, este además permite marcar una malla de perspectiva para facilitar el análisis de la trayectoria.



**Figura 79.** Perspectiva para análisis de seguimiento de trayectoria

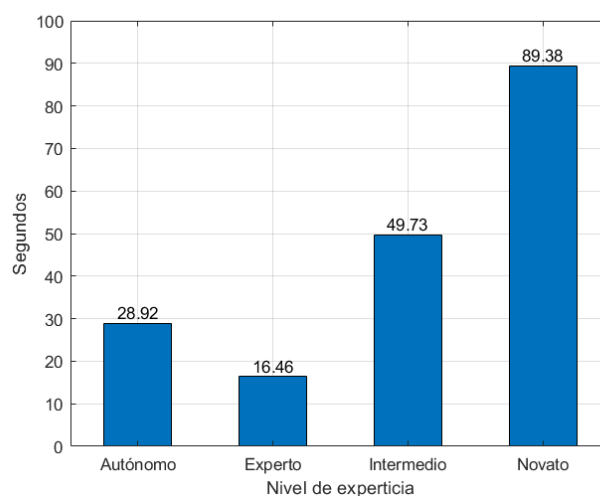
### 6.1.1 Tiempo de seguimiento de trayectoria

El tiempo de seguimiento de trayectoria se considera desde el momento en el cual despegamos el dron del punto de partida, hasta que llega al punto final (sin considerar el aterrizaje). Para evitar un análisis sesgado por un solo intento, se da la oportunidad a cada tele-operador y al algoritmo de navegación autónoma de realizar el seguimiento de la trayectoria por tres oportunidades. En la Tabla 25 y la Figura 80 se evidencia que el tele-operador experto supera a los otros tele-operadores y el algoritmo de navegación autónoma por más de 20 segundos, pero como se mostrará en el siguiente apartado, no cumple con el mejor seguimiento de trayectoria.

**Tabla 25**

*Tiempo de seguimiento de trayectoria para tres intentos*

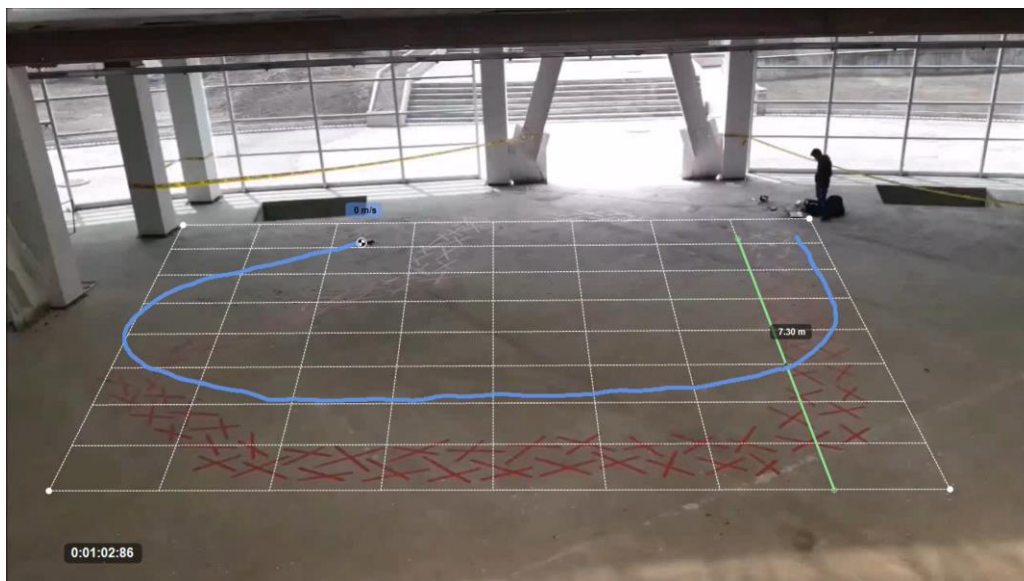
Nivel de Experticia	Autónomo	Experto	Intermedio	Novato
<b>1</b>	00:27:59	00:15:71	00:48:58	2:12:41
<b>2</b>	00:32:27	00:17:45	00:43:86	00:59:76
<b>3</b>	00:26:89	00:16:21	00:56:74	01:15:98
<b>Promedio</b>	00:28:92	00:16:46	00:49:73	01:29:38



**Figura 80.** Tiempo de seguimiento promedio para diferentes niveles de experticia

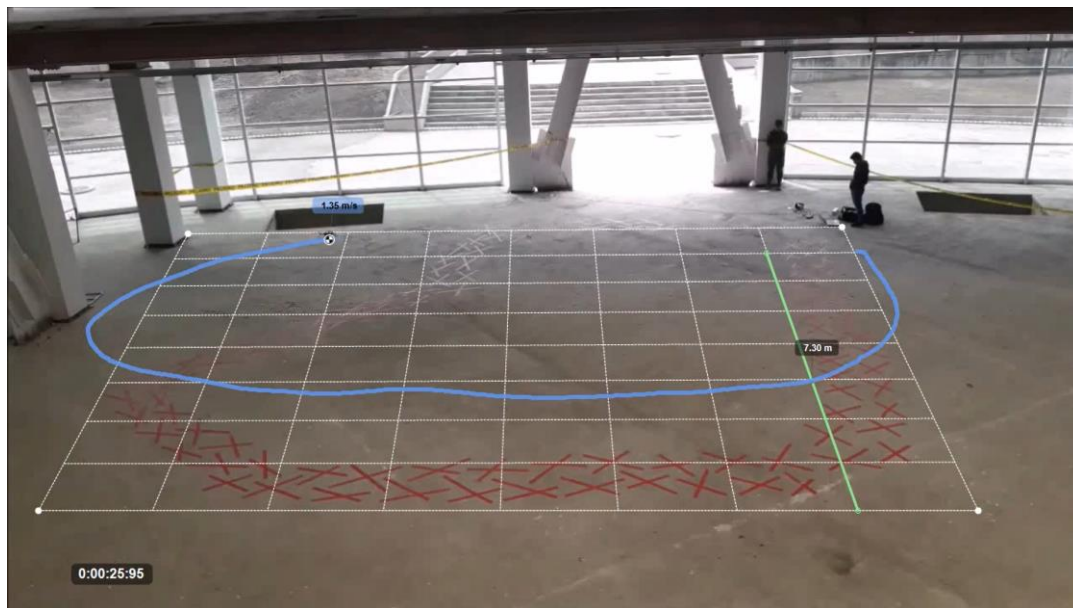
### 6.1.2 Error de seguimiento de la trayectoria

Para el cálculo del error de seguimiento de trayectoria se utilizará la raíz del error medio cuadrático, ya que este posee las mismas unidades de la unidad que se mide para este caso el error estará representado en metros. En la Figura 81 se muestra la trayectoria ideal que servirá de referencia como valor verdadera para comparar las trayectorias realizadas por los tele-operadores y el algoritmo de navegación autónoma.

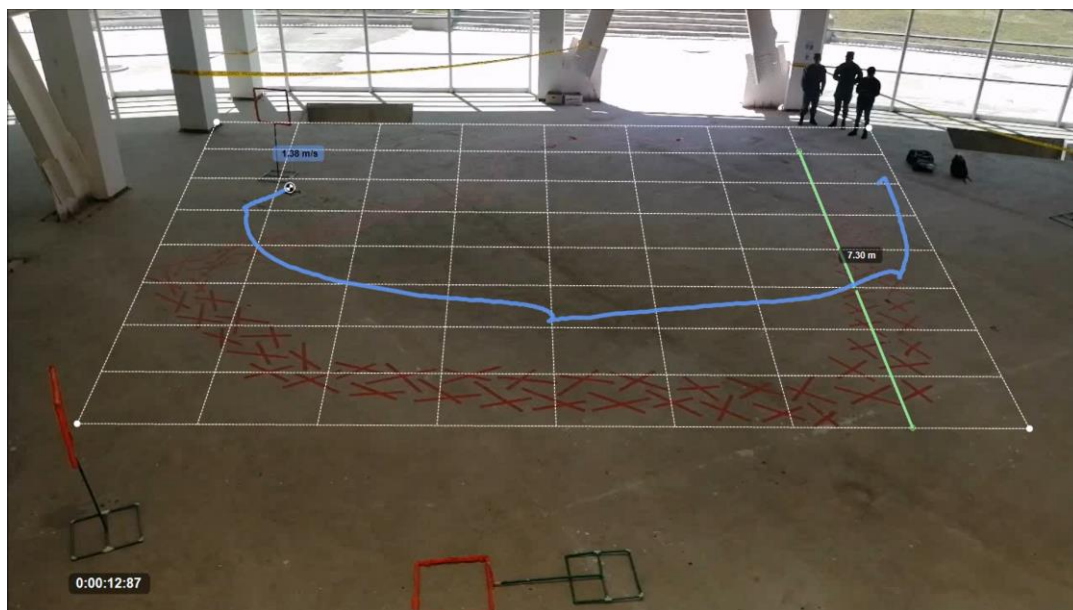


**Figura 81.** Trayectoria de referencia

Desde la Figura 82 hasta la Figura 85 se muestran las trayectorias trazadas por los tele-operadores y el algoritmo de navegación autónoma, y se evidencia que los tele-operadores no son capaces de trazar una trayectoria fina, es decir, esta se muestra errática y con curvas sinuosas, a diferencia del algoritmo de navegación autónoma que marca una trayectoria suave con respecto a la trayectoria de referencia.

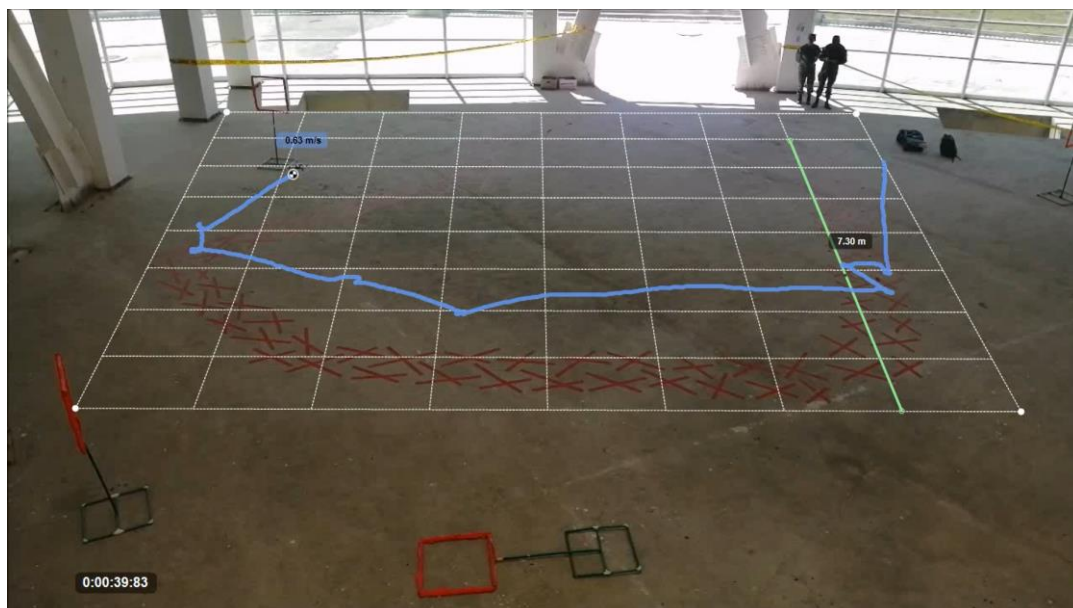


**Figura 82.** Trayectoria recorrida por el dron con el algoritmo de seguimiento de trayectoria

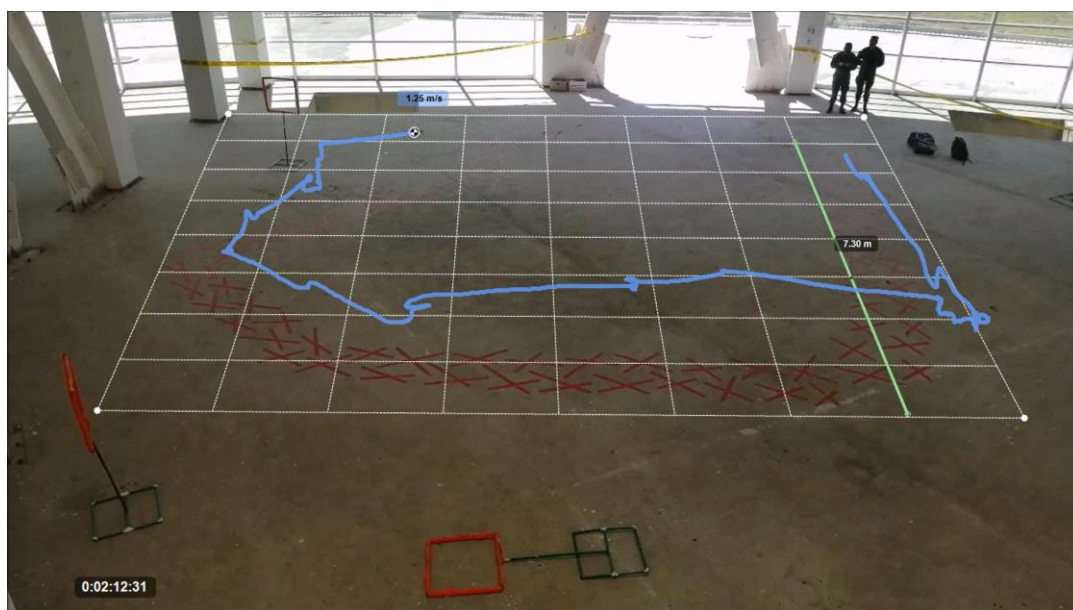


**Figura 83.** Trayectoria recorrida por el dron bajo el mando del tele-operador experto





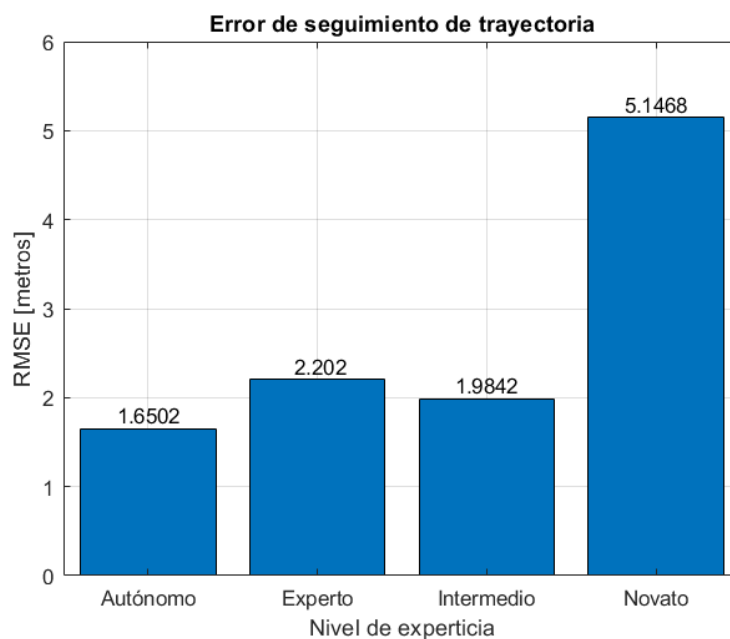
**Figura 84.** Trayectoria recorrida por el dron bajo el mando del tele-operador intermedio



**Figura 85.** Trayectoria recorrida por el dron bajo el mando del tele-operador novato

En la Figura 86 se muestra el error de seguimiento de trayectoria y se evidencia que el tele-operador novato no realiza un seguimiento correcto a la trayectoria obteniendo un error de 5.15 metros, a diferencia de los otros niveles de experticia que obtienen un error que bordea los 2 metros.

El algoritmo de navegación autónoma destaca, obteniendo un error de 1.65 metros en comparación con la trayectoria de referencia.



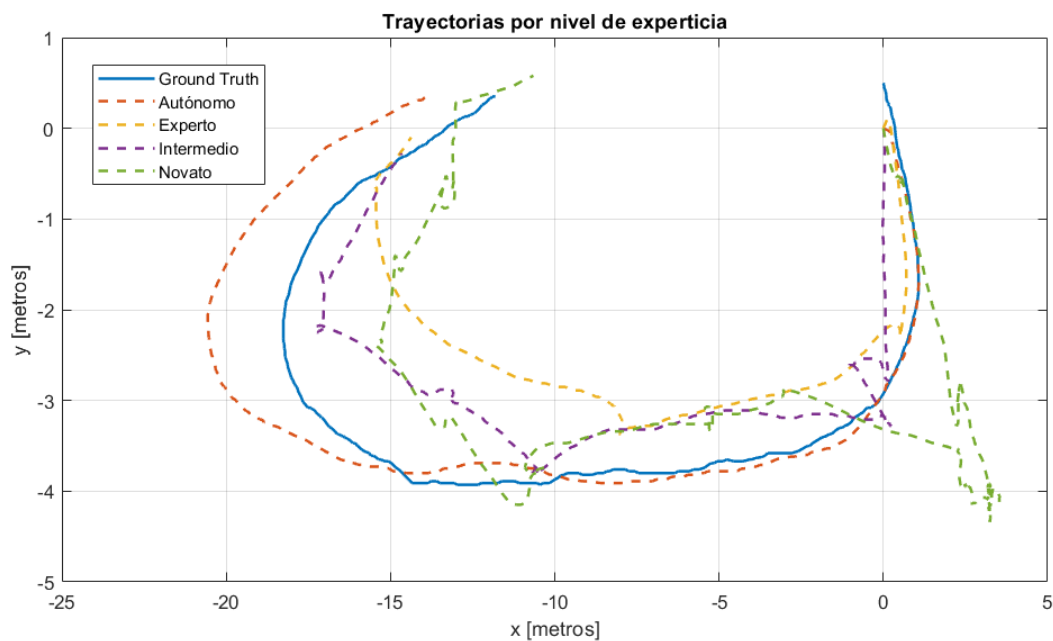
**Figura 86.** Error medio cuadrático de seguimiento de trayectoria por niveles de experticia

En la Figura 87 se muestra la trayectoria realizada por los tele-operadores y el algoritmo de navegación autónoma con corrección de perspectiva con los datos que se obtienen del software libre *Kinovea*, se puede notar claramente la diferencia que existe entre el seguimiento fino que realiza el dron de manera autónoma contra el seguimiento errático de los tele-operadores.

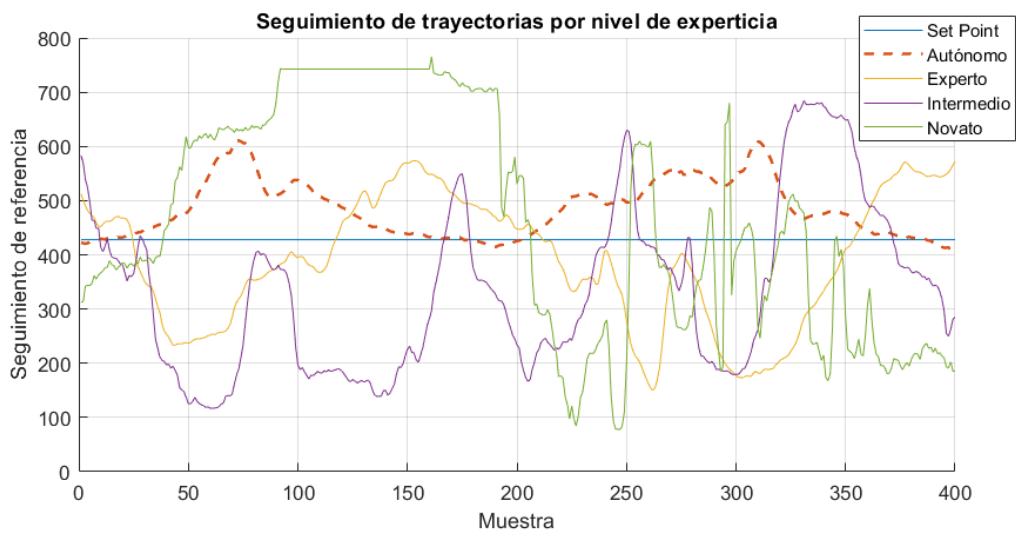
Otro punto importante a analizar es el seguimiento del punto medio de la trayectoria, ya que de esta manera se puede poner en evidencia las acciones de control que toma el tele-operador y el algoritmo de navegación autónoma cuando existe un cambio de referencia. En la Figura 88 se visualiza el seguimiento de la referencia y es notable que los movimientos que realizan los tele-



operadores son erráticos y agresivos en comparación al seguimiento del algoritmo de navegación autónoma.



**Figura 87.** Trayectoria realizada por los diferentes niveles de experticia



**Figura 88.** Seguimiento de Set Point por nivel de experticia

## 6.2 Pruebas experimentales en trayectoria con presencia de obstáculos

Para realizar las pruebas de seguimiento con presencia de obstáculos, se utilizará el circuito limitado por las marcas de superficie de la Figura 78. Se presentarán tres diferentes escenarios para validar el funcionamiento del algoritmo de navegación autónoma:

- Circuito 1: Circuito con marcas de superficie con presencia de dos obstáculos. Se considera como referencia para el cálculo del error medio cuadrático la trayectoria desde el obstáculo 1 al 2.



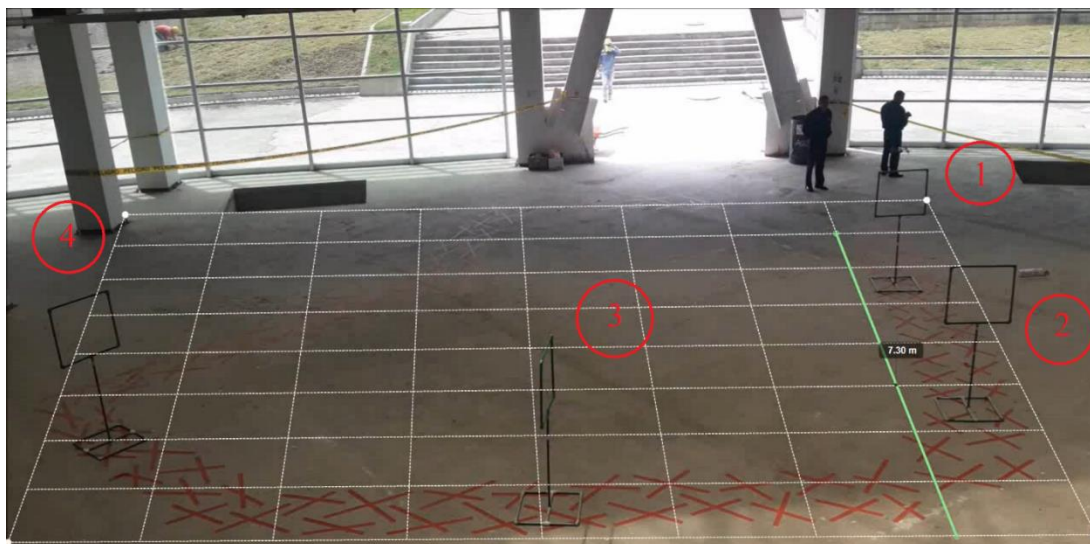
**Figura 89.** Trayectoria con presencia de 2 obstáculos

- Circuito 2: Circuito con marcas de superficie con presencia de tres obstáculos. Se considera como referencia para el cálculo del error medio cuadrático la trayectoria desde el obstáculo 2 al 3.



**Figura 90.** Trayectoria con presencia de 3 obstáculos

- Circuito 3: Circuito con marcas de superficie con presencia de cuatro obstáculos. Se considera como referencia para el cálculo del error medio cuadrático la trayectoria desde el obstáculo 2 al 3.



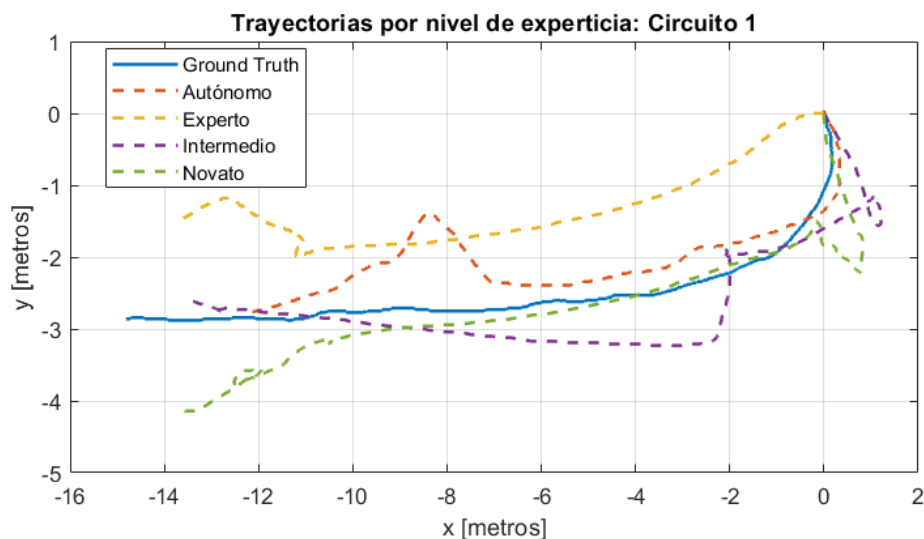
**Figura 91.** Trayectoria con presencia de 4 obstáculos

### 6.2.1 Seguimiento de trayectoria en presencia de obstáculos

En este experimento se pone a prueba la destreza del tele-operador para evadir obstáculos y conjuntamente seguir la trayectoria delimitada por las marcas de superficie. Se realizarán comparaciones del seguimiento de trayectoria únicamente entre dos obstáculos, esto se debe a que *Kinovea* tiene limitaciones en su uso y al atravesar el dron los obstáculos este pierde la trayectoria y deja de trazar la misma. En los siguientes apartados se realiza el análisis de seguimiento de trayectoria para el circuito 1, 2 y 3.

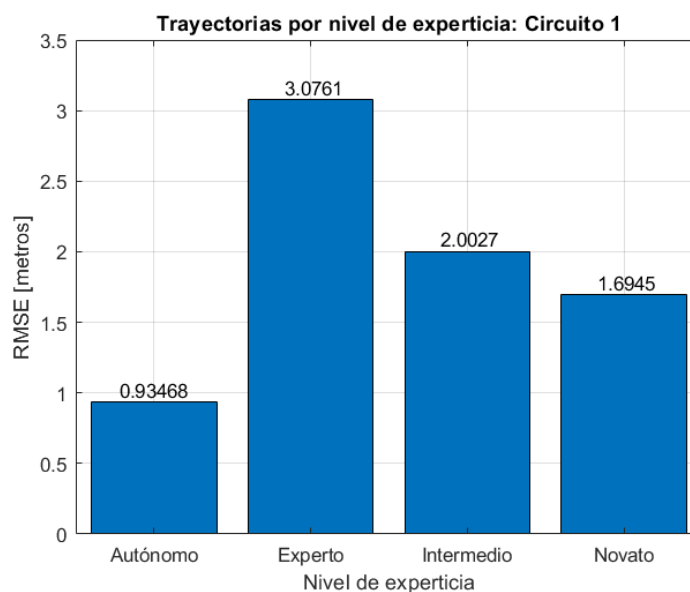
#### 6.2.1.1 Circuito 1

En la Figura 92 se visualiza la trayectoria trazada por los diferentes niveles de experticia desde el obstáculo 1 al 2, se puede notar que la trayectoria trazada por el algoritmo de navegación autónoma tiene un pico que significa un desvío en la trayectoria, esto se debe a que en ese instante el dron observa al obstáculo y deja de seguir la trayectoria, ya que atravesar el obstáculo tiene prioridad sobre el seguimiento de trayectoria.



**Figura 92.** Trayectorias trazadas por diferentes niveles de experticia en el Circuito 1

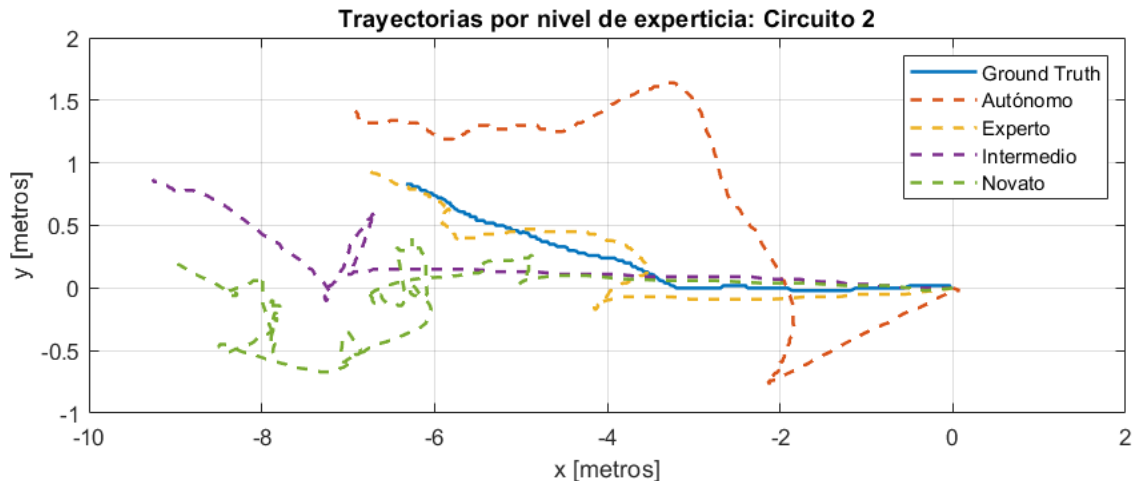
A pesar del desvío que presenta el seguimiento trazado por el algoritmo de navegación autónoma, este presenta el menor error de seguimiento con respecto a la referencia, siendo este de 0.93 m y presentando una diferencia mayor a 2 m con respecto al tele-operador experto, esto se evidencia en la Figura 93.



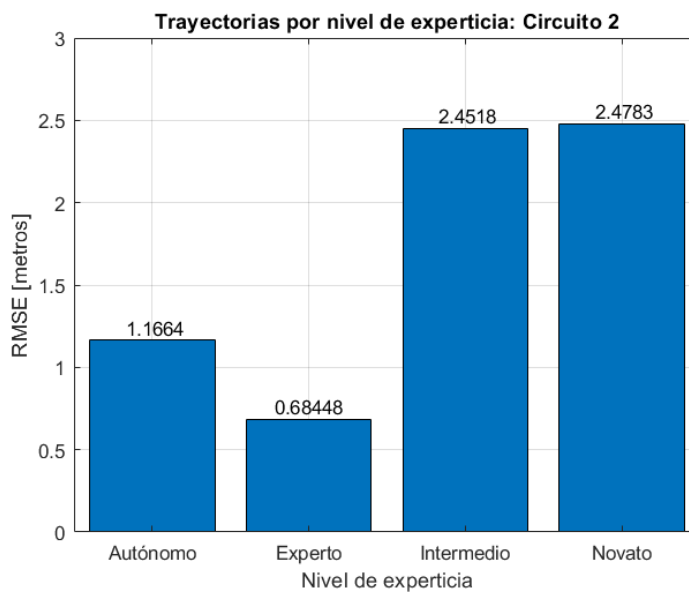
**Figura 93.** Error de seguimiento de trayectoria desde el obstáculo 1 al 2 en el Circuito 1

### 6.2.1.2 Circuito 2

En la Figura 94 se visualiza la trayectoria trazada por los diferentes niveles de experticia desde el obstáculo 2 al 3. Al igual que en el Circuito 1, el algoritmo de navegación autónoma presenta picos en su trayectoria debido a la detección de obstáculos en medio del seguimiento de la trayectoria, a pesar de esto obtuvo resultados superiores a los tele-operadores intermedio y novato. En este circuito el menor error respecto de la referencia lo obtuvo el tele-operador experto con un error de 0.68 m, esto se visualiza en la Figura 95.



**Figura 94.** Trayectorias trazadas por diferentes niveles de experticia en el Circuito 2

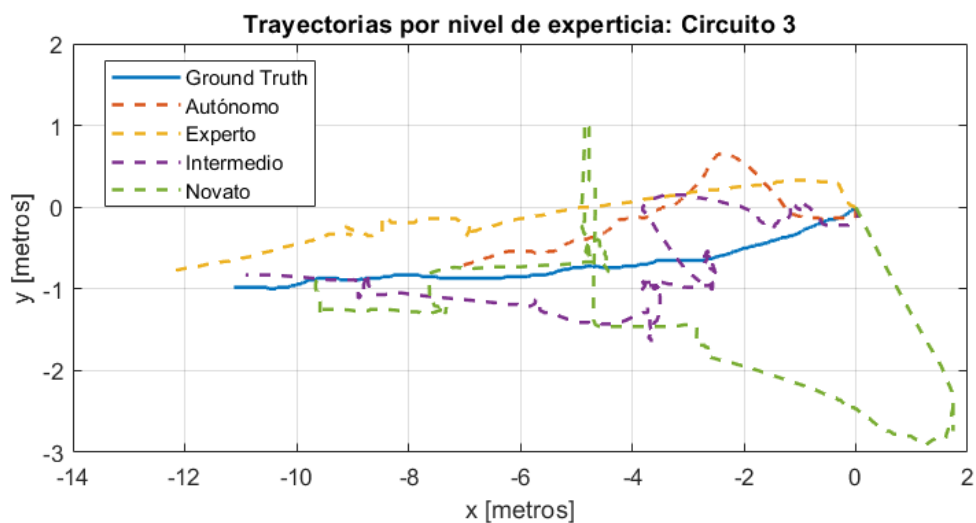


**Figura 95.** Error de seguimiento de trayectoria desde el obstáculo 2 al 3 en el Circuito 2

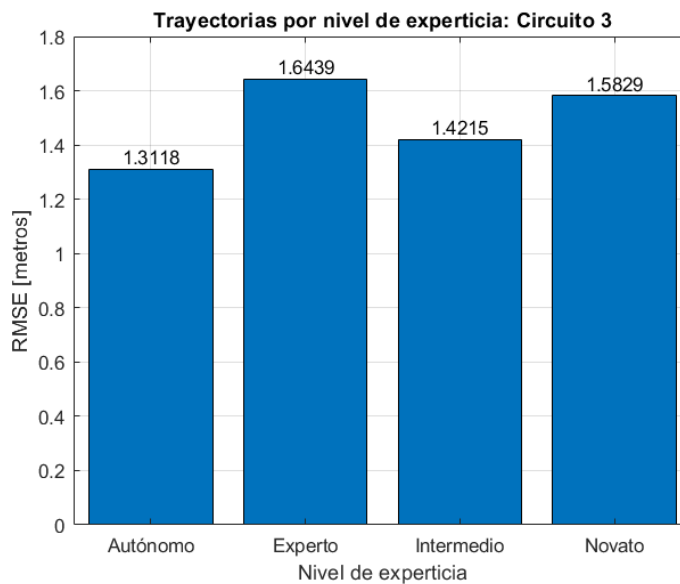
### 6.2.1.3 Circuito 3

En la Figura 96 se visualiza la trayectoria trazada por los diferentes niveles de experticia desde el obstáculo 2 al 3. Al igual que en el Circuito 1 y 2, el algoritmo de navegación autónoma presenta desvíos en su trayectoria en forma de picos debido a la detección de obstáculos en medio del

seguimiento de la trayectoria. En este circuito el menor error respecto de la referencia lo obtuvo el algoritmo de navegación autónoma con un error de 1.17 m, esto se visualiza en la Figura 97.



**Figura 96.** Trayectorias trazadas por diferentes niveles de experticia en el Circuito 3



**Figura 97.** Error de seguimiento de trayectoria desde el obstáculo 2 al 3 en el Circuito 3

En la Tabla 26 se resumen los errores de seguimiento de trayectoria para los tres circuitos, y se puede concluir que el nivel de experticia que mejor realiza la tarea de evasión de obstáculos y

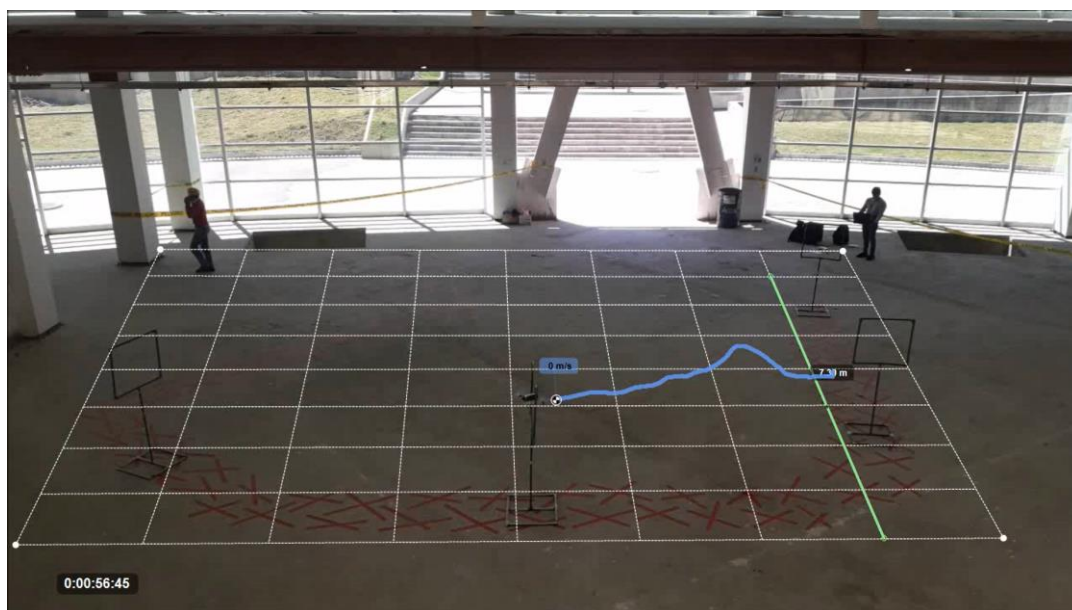


seguimiento de trayectoria es el algoritmo de navegación autónoma, teniendo como diferencia hasta 0.78 m con respecto a los tele-operadores.

**Tabla 26**

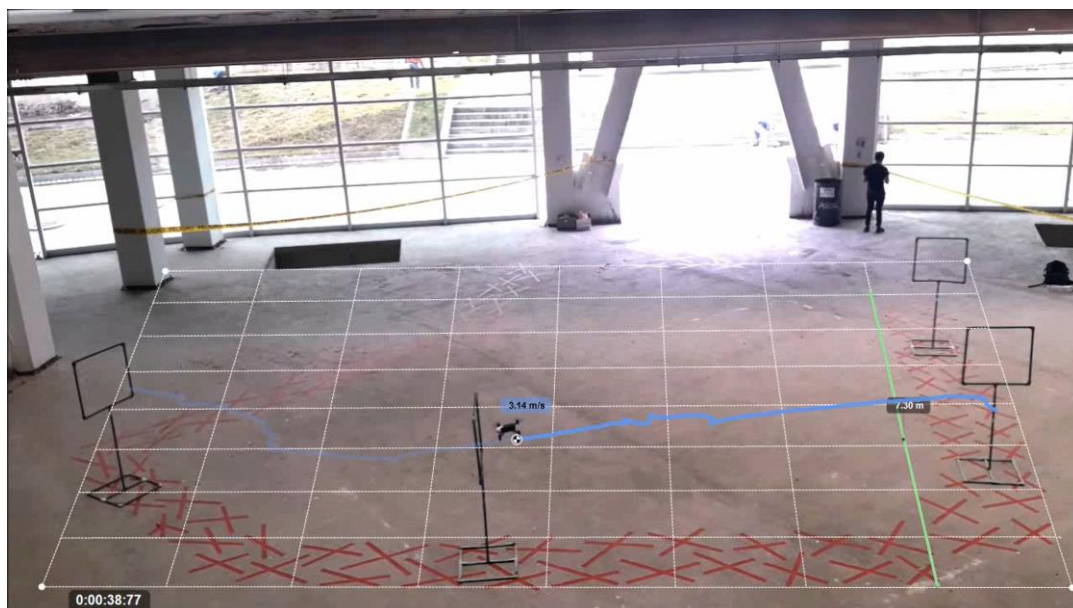
*Error de seguimiento promedio para los circuitos 1, 2 y 3*

Nivel de Experticia	Autónomo	Experto	Intermedio	Novato
<b>Circuito 1</b>	0.93468	3.0761	2.0027	1.6945
<b>Circuito 2</b>	1.1664	0.684480	2.4518	2.4783
<b>Circuito 3</b>	1.3118	1.6439	1.4215	1.5829
<b>Promedio</b>	1.14	1.80	1.96	1.92



**Figura 98.** Trayectoria trazada por el algoritmo de navegación autónoma en el circuito 3

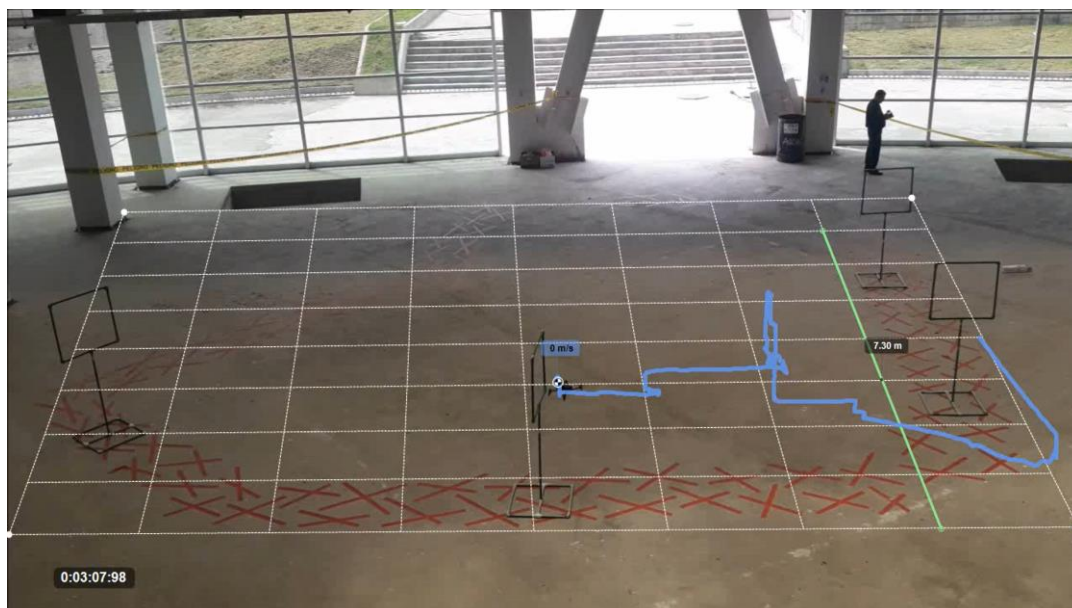




**Figura 99.** Trayectoria trazada por el tele-operador experto en el circuito 3



**Figura 100.** Trayectoria trazada por el tele-operador intermedio en el circuito 3



**Figura 101.** Trayectoria trazada por el tele-operador novato en el circuito 3

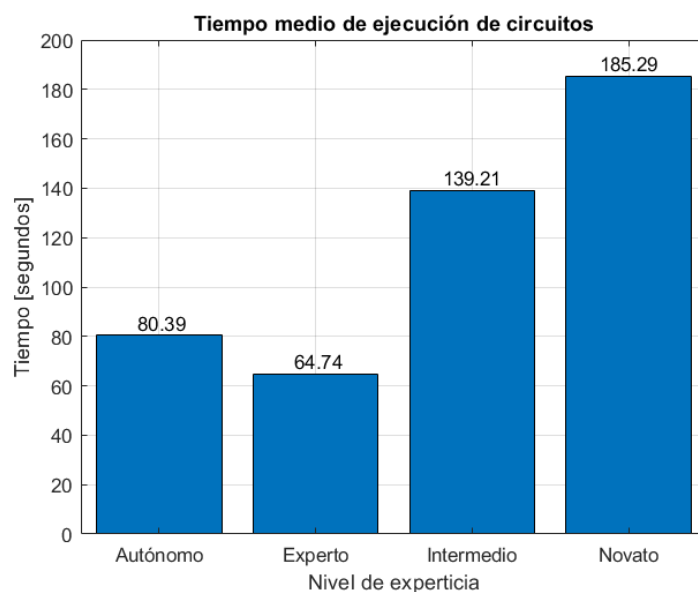
### 6.2.2 Tiempo de ejecución del circuito

El tiempo de ejecución del circuito se considera como el tiempo que transcurre desde el momento en el cual dron despegó del punto de partida, hasta que llega al punto final (sin tomar en cuenta el aterrizaje) según la secuencia de video tomada con vista cenital. En la Tabla 27 y la Figura 102, se puede evidenciar que el tele-operador experto ejecuta los circuitos con una diferencia de 15.35 segundos respecto del algoritmo de navegación autónoma, adicionalmente, se evidencia que el algoritmo de navegación autónoma supera a los tele-operadores intermedio y novato con una diferencia de más de 35 segundos.

**Tabla 27**

*Tiempo de ejecución de circuito en presencia de obstáculos*

Nivel de Experticia	Autónomo	Experto	Intermedio	Novato
<b>Circuito 1</b>	01:07:00	01:01:64	01:36:49	01:37:13
<b>Circuito 2</b>	01:21:38	01:05:63	02:26:51	03:52:63
<b>Circuito 3</b>	01:32:79	01:06:94	02:54:64	03:46:12
<b>Promedio</b>	01:20:39	01:04:74	02:19:21	03:05:29



**Figura 102.** Tiempo medio de ejecución de los circuitos 1, 2 y 3 para diferentes niveles de experticia

### 6.2.3 Tiempo de evasión de obstáculos

En la Tabla 28, Tabla 29 y Tabla 30, se muestra el tiempo promedio que los tele-operadores lograron en los Circuitos 1, 2 y 3 respectivamente. Además, en la Figura 103, se muestran los tiempos promedio de los tres circuitos en conjunto y se concluye que el algoritmo de navegación autónomo logra atravesar los obstáculos con una diferencia de 3.3 segundos en comparación con el tele-operador experto, en adición, el algoritmo de navegación autónomo atraviesa los obstáculos con una diferencia de más de 25 segundos tomando como referencia los tiempos logrados por los tele-operadores de experticia intermedia y novato.

**Tabla 28***Tiempos de evasión de obstáculos para el Circuito 1*

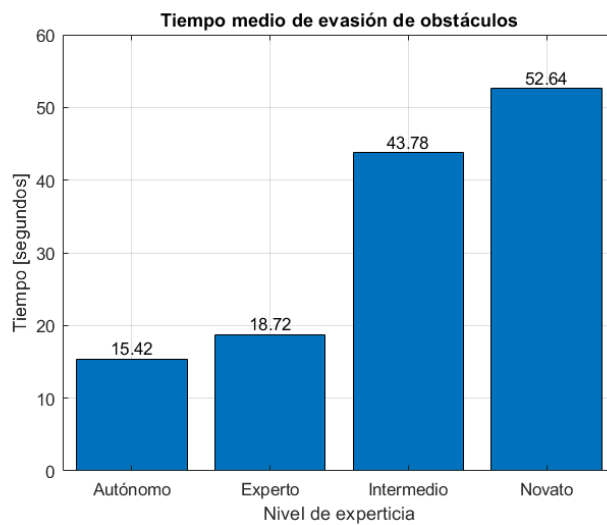
Tiempo de evasión	Autónomo	Experto	Intermedio	Novato
<b>N° Obstáculo</b>	<b>Circuito 1</b>			
1	12	21	51	41
2	20	28	40	34
<b>Promedio</b>	16.00	24.50	45.50	37.50

**Tabla 29***Tiempos de evasión de obstáculos para el Circuito 2*

Tiempo de evasión	Autónomo	Experto	Intermedio	Novato
<b>N° Obstáculo</b>	<b>Circuito 2</b>			
1	12	11	92	52
2	15	29	31	77
3	15	13	10	77
<b>Promedio</b>	14.00	17.67	44.33	68.67

**Tabla 30***Tiempos de evasión de obstáculos para el Circuito 3*

Tiempo de evasión	Autónomo	Experto	Intermedio	Novato
<b>N° Obstáculo</b>	<b>Circuito 3</b>			
1	16	10	47	79
2	17	12	44	41
3	17	15	55	62
4	15	19	20	25
<b>Promedio</b>	16.25	14	41.5	51.75



**Figura 103.** Tiempo de evasión medio de obstáculos de los circuitos 1, 2 y 3 para diferentes niveles de experticia

#### 6.2.4 Observaciones en tercera persona

Las observaciones en tercera persona solo se consideran para los tele-operadores, ya que el sistema de navegación autónomo no requiere de la intervención de un humano. Además, como el circuito debe ser realizado en primera persona, el número de visualizaciones en tercera persona será penalizado. En la Tabla 31, se muestra el número de visualizaciones promedio que realizaron los tele-operadores en el circuito 1, 2 y 3, y se puede destacar que el tele-operador novato para lograr atravesar los obstáculos necesita de una previa visualización en tercera persona, esto se permitió para evitar daños por colisiones en el dron.

**Tabla 31**

*Visualizaciones en tercera persona para tele-operadores*

Nivel de Experticia	Experto	Intermedio	Novato
<b>Circuito 1</b>	0	0	5
<b>Circuito 2</b>	0	1	7
<b>Circuito 3</b>	0	0	7
<b>Promedio</b>	0	0.33	6.33

## CAPÍTULO VII

### 7. CONCLUSIONES Y RECOMENDACIONES

#### 7.1 Conclusiones

Se implementó un sistema de navegación autónomo basado en visión para pistas de carreras de drones capaz de realizar el seguimiento de trayectorias, detección y evasión de obstáculos simultáneamente, teniendo como referencia de su entorno únicamente imágenes monoculares.

Se diseñaron tres controladores, tanto para los movimientos lineales a lo largo del eje X y Z y para el movimiento angular alrededor del eje Z, a partir de modelos matemáticos basados en visión.

El Filtro de Kalman se ha utilizado como un *suavizador* de la trayectoria que ha de seguir el dron, superando a alternativas clásicas de estimación de estados, en este caso la media y la mediana necesitan de muchos puntos para converger en un valor preciso, a diferencia del Filtro de Kalman que necesita de pocas mediciones para obtener un valor óptimo.

El seguimiento de trayectoria en presencia de obstáculos realizado por el algoritmo de navegación autónoma, supera a los tele-operadores, obteniendo un error promedio en los tres circuitos de 1.16 m, comparado con los niveles de experticia: experto, intermedio y novato, que obtuvieron errores de 1.80, 1.96 y 1.92 m, respectivamente.

La detección de obstáculos utilizando filtros de color y detección de contornos es susceptible a mediciones de la posición del obstáculo erróneas, esto se logra corregir utilizando la medición resultante con un Filtro de Kalman en lugar de la medición del centro de masa del contorno como se observa de la Figura 59 a la Figura 64.

El tiempo de evasión de obstáculos medio logrado por el dron de manera autónoma supera a los tele-operadores experto, intermedio y novato por 3.3, 28.36 y 37.22 segundos respectivamente, esto se debe al factor de indecisión e inseguridad que los tele-operadores adoptan al momento de atravesar el obstáculo, ya que si este realiza una maniobra errada podría comprometer la integridad del vehículo, por otro lado, el dron de manera autónoma al tener información fiable no adopta los factores de indecisión y atraviesa el obstáculo.

El dron con el algoritmo de navegación autónoma atraviesa los obstáculos de manera eficiente, no obstante, no logra superar en la totalidad del circuito al tele-operador experto, siendo la velocidad constante que tiene el dron entre obstáculos el factor limitante al tratar de superar al tele-operador experto.

Pese a que se realizó un control de iluminación, el sistema es susceptible a cambios de iluminación agresivos, esto se debe a las características de hardware de la cámara, ya que el parámetro de exposición a la luz presenta un retardo al cambiar su valor.

## **7.2 Recomendaciones**

Por razones de seguridad, se recomienda utilizar un ambiente abierto y con poca afluencia de personas para evitar accidentes por la pérdida de control del UAV, además, para los tele-operadores se recomienda utilizar gafas de protección y guantes para evitar lesiones por el giro de las hélices.

Antes de realizar cada vuelo de prueba se recomienda realizar una inspección breve del estado del UAV, en especial de las hélices y motores. Si las hélices no se encuentran alineadas es probable que el UAV no despegue y el giro de las hélices dañen al mismo, en el caso de los motores, si estos

no se encuentran ajustados debidamente se producen colisiones al instante de realizar maniobras de giro.

Utilizar el sistema operativo *ROS* para realizar el control del UAV, ya que este permite obtener los datos de estado del UAV a través de suscriptores que actúan como hilos, es decir, se obtiene información paralela de los estados del dron.

Se recomienda que las observaciones del obstáculo que realiza el UAV no se encuentren a contraluz para evitar colisiones, debido a que bajo estas condiciones de iluminación el dron no es capaz de detectar el obstáculo aun cuando se realice el control de iluminación.

Se recomienda que al obtener el modelo matemático del UAV basado en video, se posicione la cámara virtual en la posición en la cual se va a utilizar el modelo, además, posicionar al UAV a una distancia entre 1 y 2 metros ya que el movimiento de objetos que se encuentran cercanos al UAV es de fácil reconocimiento.

Se recomienda realizar un seguimiento del obstáculo basado en puntos de interés, teniendo como puntos consigna a los puntos que conforman el área delimitada por el contorno marcado por el *bounding box* obtenido en el proceso de detección de obstáculo.



## 8. REFERENCIAS

- Aguilar, W. G., & Angulo, C. (2014). Real-time video stabilization without phantom movements for micro aerial vehicles. *EURASIP Journal on Image and Video Processing*, 1, 1-13.
- Aguilar, W. G., & Angulo, C. (2014). Robust video stabilization based on motion intention for low-cost micro aerial vehicles. *11th International Multi-Conference on Systems, Signals & Devices (SSD)*. Barcelona, Spain.
- Aguilar, W. G., & Angulo, C. (2015). Real-Time Model-Based Video Stabilization for Microaerial Vehicles. *Neural Process Lett.*
- Aguilar, W. G., & Morales, S. (2016). 3D Environment Mapping Using the Kinect V2 and Path Planning Based on RRT Algorithms. *Electronics*, 5(4), 70.
- Aguilar, W. G., Álvarez, L., Grijalva, S., & Rojas, I. (2019). Monocular Vision-Based Dynamic Moving Obstacles Detection and Avoidance. *Intelligent Robotics and Applications. ICIRA 2019. Lecture Notes in Computer Science*, 11744, 386-398.
- Aguilar, W. G., Angulo, C., & Costa-Castello, R. (2017). Autonomous Navigation Control for Quadrotors in Trajectories Tracking. En *Lecture Notes in Computer Science* (págs. 287-297).
- Aguilar, W. G., Angulo, C., & Pardo, J. A. (2017). Motion intention optimization for multirotor robust video stabilization. *Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. Pucón, Chile.

- Aguilar, W. G., Casaliglla, V., & Pólit, J. (2017). Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles. *Electronics*, 6(1), 10.
- Aguilar, W. G., Casaliglla, V., & Pólit, J. (2017). Obstacle Avoidance for Low-Cost UAVs. *IEEE 11th International Conference on Semantic Computing (ICSC)*. San Diego.
- Aguilar, W. G., Casaliglla, V., Pólit, J., Abad, V., & Ruiz, H. (2017). Obstacle Avoidance for Flight Safety on Unmanned Aerial Vehicles. En *Lecture Notes in Computer Science* (págs. 575-584).
- Aguilar, W. G., Cobeña, B., Rodriguez, G., Salcedo, V. S., & Collaguazo, B. (2018). SVM and RGB-D Sensor Based Gesture Recognition for UAV Control. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 713-719). Springer.
- 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. *IEEE 11th International Conference on Semantic Computing (ICSC)*. San Diego.
- 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. En *Lecture Notes in Computer Science* (págs. 563-574).
- 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. En *Lecture Notes in Computer Science* (págs. 501-510).

- Aguilar, W. G., Luna, M. A., Moya, J. F., Luna, M. P., Abad, V., Ruiz, H., & Parra, H. (2017). Real-Time Detection and Simulation of Abnormal Crowd Behavior. En *Lecture Notes in Computer Science* (págs. 420-428).
- Aguilar, W. G., Luna, M. A., Ruiz, H., Moya, J. F., Luna, M. P., Abad, V., & Parra, H. (2017). Statistical Abnormal Crowd Behavior Detection and Simulation for Real-Time Applications. En *Lecture Notes in Computer Science* (págs. 671-682).
- 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. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 547-561). Springer.
- Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). RRT\* GL Based Optimal Path Planning for Real-Time Navigation of UAVs. En *Lecture Notes in Computer Science* (págs. 585-595).
- Aguilar, W. G., Morales, S., Ruiz, H., & Abad, V. (2017). RRT\* GL Based Path Planning for Virtual Aerial Navigation. En *Lecture Notes in Computer Science* (págs. 176-184).
- 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. En *Lecture Notes in Computer Science* (págs. 298-308).

- 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. En *Lecture Notes in Computer Science* (págs. 410-419).
- 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. En *Lecture Notes in Computer Science* (págs. 596-606).
- Aguilar, W. G., Salcedo, V. S., Sandoval, D. S., & Cobeña, B. (2017). Developing of a Video-Based Model for UAV Autonomous Navigation. En *Communications in Computer and Information Science* (págs. 94-105).
- Alonso, M. (Agosto de 2009). Espacios de Color RGB, HSI y sus Generalizaciones a n-Dimensiones. Puebla, Tonantzintla, México: INAOE.
- Amaguaña, F., Collaguazo, B., Tituaña, J., & Aguilar, W. G. (2018). Simulation System Based on Augmented Reality for Optimization of Training Tactics on Military Operations. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 394-403). Springer.
- 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. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 171-182). Springer.

- Basantes, J., Godoy, L., Carvajal, T., Castro, R., Toulkeridis, T., Fuertes, W., . . . Addison, A. (2018). Capture and processing of geospatial data with laser scanner system for 3D modeling and virtual reality of Amazonian Caves. *IEEE Ecuador Technical Chapters Meeting (ETCM)*. Samborondón, Ecuador.
- Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded Up Robust Features. *European Conference on Computer Vision. Lecture Notes in Computer Science*, 404-417.
- Becker, A. (2018). *About Kalman Filter*. Obtenido de KalmanFilter.NET: <https://www.kalmanfilter.net/default.aspx>
- Bedros, S. J. (2017). Lecture 2: Image Formation and Cameras. University of Minnesota.
- Dalamagkidis, K. (2014). Classifications of UAVs. *Handbook of Unmanned Aerial Vehicles*, 83-91.
- Dorobantu, A., Ozdemir, A. A., F. P., Murch, A., Mettler, B., & Balas, G. (2011). Frequency Domain System Identification for a Small, Low-Cost, Fixed-Wing UAV. *AIAA Guidance, Navigation, and Control Conference*.
- Gageik, N., & Benz, P. (2015). Obstacle Detection and Collision Avoidance for a UAV With Complementary Low-Cost Sensors. *Digital Object Finder*, 3, 599-609.
- Galarza, J., Pérez, E., Serrano, E., Tapia, A., & Aguilar, W. G. (2018). Pose Estimation Based on Monocular Visual Odometry and Lane Detection for Intelligent Vehicles. *International Conference on Augmented Reality, Virtual Reality and Computer Graphics* (págs. 562-566). Springer.

- Grijalva, S., & Aguilar, W. G. (2019). Landmark-Based Virtual Path Estimation for Assisted UAV FPV Tele-Operation with Augmented Reality. (S. Cham, Ed.) *Intelligent Robotics and Applications. ICIRA 2019. Lecture Notes in Computer Science, 11745*, 688-700.
- Harris, C., & Stephens, M. (1988). A combined corner and edge detector. *Alvey Vision Conference*, 147-151.
- Hartley, R., & Zisserman, A. (2003). *Multiple View Geometry in computer vision*. Cambridge: Cambridge University Press.
- Hassaballah, M., Abdelmgeid, A. A., & Alshazly, H. A. (2016). Image Features Detection, Description and Matching. *Studies in Computational Intelligence*, 11-45.
- Hoffer, N. V., Coopmans, C., Jensen, A. M., & Chen, Y. (2014). A Survey and Categorization of Small Low-Cost Unmanned Aerial Vehicle System Identification. *Journal of Intelligent & Robotic Systems*, 129-145.
- Hrabar, S., & Sukhatme, G. (2005). Combined Optic-Flow and Stereo-Based Navigation of Urban Canyons for a UAV. *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Edmonton.
- Huang, A. S., & Bachrach, A. (2016). Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. *Springer Tracts in Advanced Robotics, 100*, 235-252.
- Hyunchul Shim, D., Hyoun Jin, K., & Sastry, S. (2000). Control system design for rotorcraft-based unmanned aerial vehicles using time-domain system identification. Proceedings of the

2000. IEEE International Conference on Control Applications. Conference . *Proceedings of the 2000. IEEE International Conference on Control Applications*.

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. En *Smart Innovation, Systems and Technologies* (págs. 372-380).

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

Kalman, R. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 35-45.

Karpenko, S., & Konovalenko, I. (s.f.). Uav Control on the Basis of 3D Landmark Bearing-Only Observations.

Kaufmann, E., Gehrig, M., Foehn, P., Ranftl, R., Dosovitskiy, A., Koltun, V., & Scaramuzza, D. (2019). Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing. *IEEE International Conference on Robotics and Automation (ICRA)*.

Kaufmann, E., Loquercio, A., Ranftl, R., Dosovitskiy, A., Koltun, V., & Scaramuzza, D. (2018). Deep Drone Racing: Learning Agile Flight in Dynamic Environment. *2nd Conference on Robot Learning*.

- Kendoul, F. (2012). Survey of Advances in Guidance, Navigation, and Control. *Journal of Field Robotics*, 29(2), 315-378.
- Lippiello, V., & Mebarki, R. (2013). Closed-Form Solution for Absolute Scale Velocity Estimation Using Visual and Inertial Data with a Sliding Least-Squares Estimation. *Mediterranean Conference on Control & Automation (MED)*, 21, 1261-1266.
- Liu, P., & Chen Y., A. (2014). A review of rotorcraft Unmanned Aerial Vehicle (UAV) developments and applications in Civil Engineering. *Smart Structures and Systems*, 13(6), 1065-10694.
- Lowe, D. G. (November de 2004). Distinctive Image Features from Scale-Invariant Keypoints. (Springer, Ed.) *International Journal of Computer Vision*, 60(2), 91-110.  
doi:<https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- Luukkonen, T. (2011). Modelling and control of quadcopter. *Espoo*.
- Mori, T., & Scherer, S. (2013). First Results in Detection and Avoiding Front Obstacles from Monocular Camera for Micro Unmanned Aerial Vehicles. *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe.
- Nex, F., & Fabio, R. (2013). UAV for 3D mapping applications: a review. *Applied Geomatics*, 6(1), 1-15.
- Nishad, G. (8 de Marzo de 2019). *Kalman Filters : A step by step implementation guide in python*.  
Obtenido de Towards DataScience: <https://towardsdatascience.com/kalman-filters-a-step-by-step-implementation-guide-in-python-91e7e123b968>



- Nonami, K., Kendoul, F., Suzuki, S., Wang, W., & Nakazawa, D. (2010). *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*. Tokyo Dordrecht Heidelberg London New York: Springer.
- OpenCV. (06 de Agosto de 2019). *Color conversions*. Obtenido de OpenCV Documentation: [https://docs.opencv.org/3.1.0/de/d25/imgproc\\_color\\_conversions.html](https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html)
- 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. En *Lecture Notes in Computer Science* (págs. 199-211).
- Orbea, D., Moposita, J., Aguilar, W. G., Paredes, M., Reyes, R. P., & Montoya, L. (2017). Vertical take off and landing with fixed rotor. *Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. Pucón, Chile.
- Pardo, J. A., Aguilar, W. G., & Toulkeridis, T. (2017). Wireless communication system for the transmission of thermal images from a UAV. *Chilean Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. Pucón, Chile.
- Penguin Engine. (7 de Diciembre de 2017). *Drone Classification and Types*. Obtenido de Technology/Drones: <http://penguinengine.com/drone-classification-types/>
- Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. *9th European Conference on Computer Vision*, 430-443.
- Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision*.

- Russ, J. C. (2011). *The Image Processing Handbook*. New York: CRC Press Taylor & Francis Group.
- Sa, I., Hrabar, S., & Corke, P. (2015). Outdoor Flight Testing of a Pole Inspection UAV Incorporating High-speed Vision. (C. Springer, Ed.) *Field and Service Robotics. Springer Tracts in Advanced Robotics*, 105.
- Salcedo, V. (2018). Aterrizaje automático de un vehículo aéreo no tripulado basado en seguimiento de puntos de interés para superficies móviles. Sangolquí.
- Scaramuzza, D., & Fraundorfer, F. (2011). Visual Odometry Part I: The first 30 Years and Fundamentals. *IEEE Robotics & Automation Magazine*, 80-92.
- Skrede, O.-J. (08 de Marzo de 2017). *University of Oslo*. Obtenido de Department of Informatics: The Faculty of Mathematics and Natural Sciences: [https://www.uio.no/studier/emner/matnat/ifi/INF2310/v17/undervisningsmateriale/slides\\_inf2310\\_s17\\_week08.pdf](https://www.uio.no/studier/emner/matnat/ifi/INF2310/v17/undervisningsmateriale/slides_inf2310_s17_week08.pdf)
- Soloviev, A., & Rutkowski, A. J. (2009). Fusion of inertial, optical flow, and airspeed measurements for UAV navigation in GPS-denied environments. *Unmanned Systems Technology XI*. doi:doi: 10.1117/12.820177
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- Teow, J. (3 de Mayo de 2018). *Understanding Kalman Filters with Python*. Obtenido de Medium: <https://medium.com/@jaems33/understanding-kalman-filters-with-python-2310e87b8f48>

Yi, J., & Junjie, Z. (2007). IMU-based localization and slip estimation for skid-steered mobile robots. *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego.

Zhang, C., & Kovacs, J. M. (2012). The application of small unmanned aerial systems for precision agriculture: a review. *PrecisionAgriculture*, *13*(6), 693-712.