Máster Universitario en Ingeniería de Sistemas y Control

Aplicación de técnicas de Tracking para el control de un robot autónomo aéreo

Autor: Jesús Mª Pérez Colmenar

Directores:

Dictino Chaos García

Luis de la Torre Cubillo

Curso 2014-2015

Convocatoria de Septiembre

Máster Universitario en Ingeniería de Sistemas y Control

Aplicación de técnicas de Tracking para el control de un robot autónomo aéreo

Proyecto tipo B: Proyecto específico propuesto por el alumno

Autor: Jesús Mª Pérez Colmenar

Directores:

Dictino Chaos García

Luis de la Torre Cubillo

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: JM Pérez Colmenar

A Teresa por acompañarme todos los días que el drone no volaba.

Resumen del proyecto

En los últimos años gracias a los avances en la robótica y la electrónica de consumo hemos podido ver cómo cada día surgen nuevas aplicaciones de los llamados drones. Lo que comenzó siendo un juguete teledirigido tiene innumerables aplicaciones añadiéndole una cámara y distintos sensores. Desde aplicaciones de seguridad ya existentes, hasta labores de mensajería, estos aparatos tienen un gran potencial del que seremos testigos en los próximos años.

Dentro del mercado audiovisual podemos encontrar un gran número de películas, series y anuncios que han hecho uso de drones para capturar contenido aéreo a un precio mucho más reducido que con el uso de un helicóptero y con una calidad de imagen semejante.

El uso de drones en eventos televisados de bajo presupuesto, puede enriquecer enormemente el atractivo de esos eventos para los espectadores, por ejemplo, deportes de aventura en montaña. Para el productor, un drone será una cámara con movimiento libre con la que poder capturar escenas nunca vistas antes en retrasmisiones de bajo presupuesto.

La tendencia creciente de los usuarios a generar contenido audiovisual y el éxito de las cámaras deportivas indican que este tipo de usuarios estarían encantados de grabarse realizando alguna actividad, abstrayéndose del control de la cámara y consiguiendo un contenido con un atractivo insuperable.

Este proyecto tiene el objetivo de controlar un drone comercial no profesional usando tecnologías de visión artificial y utilizando únicamente la imagen que obtenemos del drone como sensor para controlarlo.

Como se demostrará en esta memoria, con la tecnología en visión artificial existente en la actualidad, es posible controlar un cuadricóptero comercial únicamente a través del análisis de imagen de la cámara que porta.

Palabras clave

Drone, Cuadricóptero, control, Tracking, Visión Artificial

Índice

1 Introducción 13
1.1 Motivación del proyecto 13
1.2 Objetivos del proyecto 13
1.3 Vicomtech 14
1.4 Cronograma del proyecto 14
2 Estado del arte 16
2.1 Vehículos UAV autónomos para captura de contenidos 16
2.1.1 Airdog 16
2.1.2 Lily 17
2.1.3 DJI Inspire 1 18
2.1.4 Bebop Drone 19
2.2 Tecnologías para el desarrollo del proyecto 20
2.2.1 Visión por computador en robots móviles 20
2.2.2 Tecnologías de visión artificial 22
2.2.3 Tracking de personas 23
2.2.4 CMT 24
2.3 Dinámica de los cuadricópteros 26
3 Desarrollo 34
3.1 Cuadricóptero seleccionado: Bebop Drone 34
3.1.1 SDK de desarrollo 35
3.1.2 Protocolo de comunicaciones del Bebop Drone 36
3.1.3 Análisis del protocolo 38
3.2 Herramientas utilizadas 39
3.2.1 IDE de desarrollo 39
3.2.2 FFmpeg 39
3.2.3 OpenCv 2.4.10 39
3.3 Set Up del entorno de desarrollo 39
4 Arquitectura y funcionamiento de la aplicación 41
4.1 Funcionamiento de la aplicación 42
5 Diseño de pruebas 50

6 Realización de pruebas 51
6.1 Tracking en interiores con una única persona u objeto 51
Tracking de una persona impidiendo el movimiento del drone, detector BRISK: 55
Tracking de un objeto impidiendo el movimiento del drone, detector FAST: 57
Tracking de una persona en interiores impidiendo el movimiento del drone, detector FAST: 59
Tracking de un objeto en interiores permitiendo movimiento Yaw 64
6.2 Tracking en interiores cambio de color de ropa 68
Reacción ante eliminación de una prenda, detector BRISK 69
Reacción ante eliminación de una prenda, detector FAST 71
6.3 Tracking en exteriores en condiciones no controladas 74
Tracking de una única persona en exteriores con detector BRISK 74
Tracking de una única persona en exteriores, disminuimos la distancia de seguridad 80
D:\pruebas\outdoor\tercera prueba 80
Tracking de una persona con detector FAST 82
6.4 Tracking en exterior cruce de personas 84
Cruce de personas con detector BRISK, prueba 1 84
Cruce de personas con detector BRISK, prueba 2 88
Tracking exterior cruce de personas FAST 90
6.5 Tracking exterior a un objeto voluminoso 93
Tracking exterior a coche con detector BRISK 93
Tracking exterior a coche FAST 97
7 Conclusiones 99
7.1 Problemas encontrados en el desarrollo 100
7.2 Posibles líneas de investigación 101
8 Referencias 103

Índice de figuras

Figura 1. Logotipo de Vicomtech-IK4 14
Figura 2. Diagrama Gantt del proyecto 15
Figura 3. Airdog 17
Figura 4. Drone lily 18
Figura 5. Drone DJI Inspire 1 19
Figura 6. Bebop drone 19
Figura 7. Funcionamiento de la visión estereoscópica 21
Figura 8 Robot autónomo que usa el sensor kinect para la navegación 22
Figura 9. Puntos característicos votando por el centro 24
Figura 10. Consenso para buscar el centro del objeto 25
Figura 11. Movimientos del cuadricóptero 27
Figura 12. Polos del sistema en lazo abierto 31
Figura 13. Diagrama de bloques del sistema 31
Figura 14. Sistema inestable con control proporcional 32
Figura 15. Respuesta escalón con control PD 32
Figura 16. Polos del sistema con un controlador PD 33
Figura 17. Estabilización de imagen del Bebop Drone 34
Figura 18. Ejecución del ejemplo del SDK proporcionada por Parrot 35
Figura 19. Analizador Ardrone3 PCAP, análisis de aplicación FreeFlight 38
Figura 20. Funcionamiento de la aplicación de control 41
Figura 21. Funcionamiento del sistema de control del drone 42
Figura 22. Selección del objeto o persona a trackear 43
Figura 23. Tracking en funcionamiento 43
Figura 24. Cálculo de desviación del Yaw 45
Figura 25. Simulación del control del Yaw 46
Figura 26. Ejecutando aterrizaje de emergencia, la ventana de emergencia se encuentra en la esquina inferior derecha 48
Figura 27. Trackeando un objeto en interiores 51
Figura 28. Funcionamiento del Tracker 52
Figura 29. Yaw simulado 53
Figura 30. Pitch simulado 54
Figura 31. Roll con el drone en reposo 54
Figura 32. Tracking de una persona sin movimiento del drone 55
Figura 33. Funcionamiento del tracker interior, una persona 56
Figura 34. Pich, interior, una persona 56
Figura 35. Tiempo de procesado del tracker para cada frame 56
Figura 36. Tracking de un objeto indoor con detector FAST 57
Figura 37. Tracker funcionando 58

Figura 38. Tiempo de procesamiento de cada frame 58
Figura 39. Detector Fast pierde el objeto 58
Figura 40. Tracking de una persona en interiores con detector FAST 59
Figura 41. Funcionamiento del tracker 59
Figura 42. Tiempo de procesado 59
Figura 43. Tracker con detector FAST perdido en interiores 60
Figura 44. Tracking indoor con fondo e iluminación controlado 61
Figura 45. Tiempo de funcionamiento del tacker, detector BRISK en condiciones controladas 61
Figura 46. Tiempo de procesamiento de cada frame, detector BRISK en condiciones controladas 62
Figura 47. Tracking indoor controlado detector FAST 63
Figura 48. Funcionamiento del tracker 63
Figura 49. Tiempo de procesado por cada frame 64
Figura 50. Tracking de un objeto permitiendo el movimiento Yaw 65
Figura 51. Tiempo de funcionamiento del tracker 66
Figura 52. Objeto desaparece 66
Figura 53. Cambio de Yaw en el tracking 67
Figura 54. Pitch actual del drone y comando simulado 68
Figura 55. Funcionamiento del detector BRISK ante cambio de color de ropa 69
Figura 56. Tracking inicial 70
Figura 57. Cambio de prenda 70
Figura 58. Falso positivo del tracker 71
Figura 59. Funcionamiento del tracker 72
Figura 60. Instante inicial del tracking 72
Figura 61. Cambio de prenda 73
Figura 62. Fallo del tracker 73
Figura 63. Tracking en exteriores 75
Figura 64. Funcionamiento del tracker 75
Figura 65. Distancia relativa 75
Figura 66. Yaw exteriores 76
Figura 67. Pitch exteriores 76
Figura 68. Tracking perdido por unos instantes 77
Figura 69. Posición estimada del drone 77
Figura 70. Velocidad estimada del drone 77
Figura 71. Funcionamiento del tracker 79
Figura 72. Distancia relativa 79
Figura 73. Yaw 79
Figura 74. Pitch 79
Figura 75. Posición estimada 79
Figura 76. Velocidad estimada 79

Figura 77. Mala detección de la anchura del sujeto 80
Figura 78. Tracking frente al sol 80
Figura 79. Funcionamiento del tracker 81
Figura 80. Distancia relativa 81
Figura 81. Yaw 81
Figura 82. Pitch 81
Figura 83. Posición estimada 81
Figura 84. Velocidad estimada 81
Figura 85. Mal funcionamiento del Tracker 82
Figura 86. Tracking con detector FAST en exteriores 82
Figura 87. Funcionamiento del tracker 83
Figura 88. Distancia relativa 83
Figura 89. Yaw 83
Figura 90. Pitch 83
Figura 91. Posición estimada 83
Figura 92. Velocidad estimada 83
Figura 93. Tracker perdiendo puntos 84
Figura 94. Tracker funcionando correctamente 85
Figura 95. Deterioro al realizar el cruce 86
Figura 96. Tracker perdido 86
Figura 97. Funcionamiento del tracker 87
Figura 98. Distancia relativa 87
Figura 99. Yaw 87
Figura 100. Pitch 87
Figura 101. Posición estimada 87
Figura 102. Velocidad estimada 87
Figura 103. Tracker perdido 88
Figura 104. Funcionamiento del tracker 89
Figura 105. Distancia Relativa 89
Figura 106. Yaw 89
Figura 107. Pitch 89
Figura 108. Posición estimada 89
Figura 109. Velocidad estimada 89
Figura 110. Tracker vuelve a encontrar al sujeto 90
Figura 111. Cruce de personas con detector FAST 90
Figura 112. Los puntos característicos se diluyen 91
Figura 113. El tracker ha perdido al sujeto a seguir 91
Figura 114. Funcionamiento del tracker 92
Figura 115. Roll 92
Figura 116. Yaw 92

Figura 117. Pitch 92
Figura 118. Posición estimada 92
Figura 119. Velocidad estimada 92
Figura 120. Tracking a un coche, detector BRISK 93
Figura 121. El drone adelanta al coche 94
Figura 122. El tracker se pierde 95
Figura 123. Funcionamiento del tracker 96
Figura 124. Distancia relativa 96
Figura 125. Yaw 96
Figura 126. Pich 96
Figura 127. Posición estimada 96
Figura 128 Velocidad estimada 96
Figura 129. Tracking a coche con detector FAST 97
Figura 130. Puntos característicos expandiendose en la imagen 97
Figura 131. Funcionamiento del tracker 98
Figura 132. Distancia relativa 98
Figura 133. Yaw 98
Figura 134. Pitch 98
Figura 135. Posición estimada 98
Figura 136. Velocidad estimada 98

1 Introducción

Esta memoria recoge el trabajo realizado dentro del proyecto final de máster "Aplicación de técnicas de Tracking para el control de un robot autónomo aéreo". Este proyecto se ha realizado dentro del departamento de Televisión digital y servicios multimedia del centro tecnológico Vicomtech-IK4[1].

1.1 Motivación del proyecto

Durante los últimos años hemos experimentado un gran auge en el uso de los denominados drones en distintos tipos de aplicaciones. Los drones son vehículos aéreos teledirigidos, esto es, vehículos no tripulados.

Fuera del uso militar, este tipo de robots ha generado gran interés tanto en la comunidad científica como en el mercado de la electrónica de consumo como veremos en la sección 2.1 el estado del arte.

El hecho de que los drones dispongan de cámaras cada vez de mayor calidad con las que poder grabar contenidos aéreos a un bajo coste lo convierten en una herramienta con gran potencial en el mundo de la producción audiovisual. Salvando los problemas de autonomía de vuelo y la actual regulación, el uso de drones será el sustituto ideal de los helicópteros en eventos deportivos de gran interés como el ciclismo.

Dentro de la electrónica de consumo es de esperar que el uso de drones como cámara de vista aérea, tenga un gran éxito en la práctica de deportes de aventura, al igual que el éxito que tienen en la actualidad las cámaras deportivas.

1.2 Objetivos del proyecto

El objetivo de este trabajo es dotar a un cuadricóptero a control remoto (drone) dotado de una cámara de vídeo de alta resolución de ciertas capacidades de colaboración autónoma con un operario humano para la realización de material audiovisual para TV.
En particular se plantean las siguientes metas:

1) Búsqueda de un cuadricóptero adecuado para las actividades que se van a realizar en el centro tecnológico Vicomtech-IK4.
2) Estudio, adquisición e integración de los sensores necesarios para la colaboración entre el drone y el operario humano.

[1] http://www.vicomtech.org

3) Implementación de un sistema de seguimiento (tracking) que permita al drone seguir de forma autónoma al operario humano.

1.3 Vicomtech

Este proyecto ha sido desarrollado gracias el apoyo del departamento de TV digital y servicios multimedia de VICOMTECH-IK4, a través del cual se ha adquirido el drone con el que desarrollar el proyecto y se ha asignado el tiempo necesario para la realización del proyecto.

Figura 1. Logotipo de Vicomtech-IK4

VICOMTECH-IK4 es un centro de investigación aplicada especializado en las tecnologías de Computer Graphics, Visual Computing y Multimedia, fundado en 2001 y localizado en el Parque Tecnológico de San Sebastián.

El departamento de televisión digital y servicios multimedia crea e integra tecnología para producciones audiovisuales con la que se puede ampliar cualquiera de las dimensiones de un vídeo o un audio: 3D y estereoscopia, rastreo de objetos en tiempo real, creación de vídeo en tiempo real y herramientas de edición automática, herramientas de gestión de flujos de vídeo, distribución y transmisión de vídeos, mediciones de calidad del vídeo y de su transmisión.

1.4 Cronograma del proyecto

En la siguiente figura podemos ver el cronograma del proyecto. A pesar de que este proyecto se presente en Septiembre de 2015, su desarrollo ha sido desde Febrero a Junio de 2015.

	Feb	Mar	Abr	May	Jun
	M1	M2	M3	M4	M5
T1 - Búsqueda y adquisición del Drone					
T2 -Instalación de entorno de desarrollo					
T3 - Pruebas de tracking					
T4 -Análisis de protocolo de comunicaciones Drone					
T 5- Desarrollo de control para Drone					
T6 - Pruebas de laboratorio					
T7 - Pruebas de campo					
T8 - Redacción de memoria					

Figura 2. Diagrama Gantt del proyecto

Como vemos el proyecto se ha dividido en 8 tareas principales:

T1 – Búsqueda y adquisición del Drone: esta tarea consistió en la búsqueda de un Drone que se adaptase a las necesidades del proyecto, una buena relación calidad/precio y un SDK para su desarrollo.

T2 – Instalación de entorno de desarrollo: esta tarea consistía en la instalación del SDK de desarrollo de la plataforma.

T3 – Pruebas de tracking: Paralelamente a la T2 y T4 se han buscado distintas implementaciones de algoritmos de tracking de personas con el código publicado.

T4 – Análisis de protocolo de comunicaciones del Drone: debido al mal funcionamiento del SDK y a la falta de documentación de desarrollo ha sido necesario realizar un análisis de las comunicaciones entre el Drone y el controlador.

T5 – Desarrollo de controlador del Drone: esta tarea es la tarea principal del proyecto donde se integran las funcionalidades de un tracker externo para que el drone pueda seguir un objeto o persona.

T6 – Pruebas de laboratorio: esta tarea consistía en probar las funcionalidades de tracking y el movimiento del robot ante el control.

T7 – Pruebas de campo: esta tarea es la validación de los desarrollos previos realizados.

T8 – Redacción de la memoria.

2 Estado del arte

En este apartado realizaremos un repaso a las distintas iniciativas comerciales existentes en el mercado audiovisual y la electrónica de consumo para el vuelo autónomo de drones, haciendo especial hincapié en las aplicaciones cuyo objetivo sea la captura de contenidos audiovisuales. Por otro lado, también repasaremos las distintas tecnologías utilizables para alcanzar los objetivos del proyecto.

2.1 Vehículos UAV autónomos para captura de contenidos

En la actualidad ya podemos encontrar distintas iniciativas comerciales cuyos objetivos son similares a los expuestos en este proyecto. Una de las aplicaciones más interesantes del uso de drones es poder grabar contenidos en la práctica de deportes de aventura. El éxito de las cámaras deportivas como la GoPro[2] implica que el uso de drones en la práctica de este tipo de deportes tendrá un gran éxito en un futuro cercano.

A continuación veremos ejemplos de este tipo de aplicaciones y los robots principales que se han valorado para el uso dentro del proyecto:

2.1.1 Airdog

Airdog[3] es una iniciativa de Kickstarter cuyo objetivo es desarrollar un drone para que el usuario pueda grabarse realizando algún tipo de deporte sin necesidad de tener que estar controlándolo. La iniciativa ha conseguido cerca de 2 millones de dólares (USD) de financiación y en Junio de 2015 ha comenzado la producción en serie de los primeros drones.

[2] http://es.gopro.com/
[3] https://www.airdog.com/

Figura 3. Airdog

Este drone usa una cámara GoPro para capturar los contenidos que el usuario desee. El pilotaje autónomo de este drone viene dado por un localizador que el usuario debe portar. Este localizador también realiza las labores de control para el despegue, aterrizaje y comenzar a grabar.

2.1.2 Lily

Otro ejemplo similar al anterior lo encontramos en Lily[4], este drone al igual que el anterior sigue al usuario a través de un localizador que realiza las funciones de control remoto. Este proyecto se encuentra en pleno desarrollo, obtiene su financiación a través de las pre-órdenes de compra y según podemos leer en su blog, se espera que las primeras unidades comerciales sean enviadas en febrero de 2016.

[4] https://www.lily.camera/

Figura 4. Drone lily

Este robot tiene integrada la cámara y además es impermeable. El resto de funcionalidades son similares a las del anterior.

2.1.3 DJI Inspire 1

El fabricante de drones DJI presentó a finales del año pasado su modelo Inspire 1. Este drone tiene la característica de ser el primer drone en integrar una cámara con resolución 4K (3840x2160 píxeles).

Figura 5. Drone DJI Inspire 1

Se valoró el uso de este drone dentro del proyecto debido a la calidad de los contenidos que puede capturar gracias a su cámara 4K. Por otro lado, esta cámara es estabilizada por un giróscopo, con lo cual se evitan todas las vibraciones inherentes al vuelo.

Finalmente no se decidió el uso de este drone dentro del proyecto por la falta de un SDK público en el arranque de este proyecto con el que poder programar vuelos autónomos a través de análisis de imagen.

2.1.4 Bebop Drone

El Bebop Drone de Parrot es la tercera generación de los drones ARDrone de Parrot. La gran mejora de este drone respecto a sus hermanos mayores reside en la calidad de imagen que puede alcanzar gracias a su cámara de 14 Megapíxeles con una lente de ojo de pez.

Figura 6. Bebop drone

Gracias a la cámara de alta calidad, a la lente de ojo de pez y a un algoritmo de estabilización de imagen, el Bebop Drone puede capturar vídeo a resolución fullHD (1920x1080 píxeles) totalmente estabilizado.

Este drone ha sido el elegido para el desarrollo del proyecto por dos principales motivos, por un lado su bajo precio y la relación calidad de imagen/precio y por otro lado por disponer de un SDK público.

Como hemos visto, existen iniciativas con los mismos objetivos que este proyecto, sin embargo, a fecha de inicio del proyecto estas iniciativas se encontraban en pleno desarrollo o en busca de financiación. Por este motivo, se descartan los dos primeros drones para el desarrollo y se decide utilizar el Bebop drone por ofrecer un SDK público.

2.2 Tecnologías para el desarrollo del proyecto

Este proyecto tiene como objetivo controlar a un cuadricóptero a través de análisis de imagen de la cámara que estos aparatos suelen poseer, por este motivo, la principal tecnología sobre la que se basa es la visión artificial.

2.2.1 Visión por computador en robots móviles

En este subapartado vamos a intentar realizar una introducción a la problemática a la que nos enfrentamos dentro de la visión por computador.

Podemos considerar que la visión por computador tiene cierto grado de madurez si tenemos en cuenta aplicaciones que funcionan desde hace ya unos años, como puede ser la detección de matrículas de coche, sin embargo, si realizamos una búsqueda en la literatura del estado del arte actual, podemos comprobar que estas tecnologías necesitan todavía una larga fase de maduración.

En la mayoría de aplicaciones de visión por computador para el tracking de personas en las que se obtienen buenos resultados se dispone de un entorno totalmente controlado, con una cámara fija y un fondo que previamente se conoce. En el caso que nos ocupa tenemos una única cámara móvil en un entorno descontrolado.

Muchos de los problemas existentes en la visión por computador aplicada a robots móviles se pueden solucionar utilizando una segunda cámara de forma que podamos tener una referencia tridimensional del espacio donde se mueve el robot.

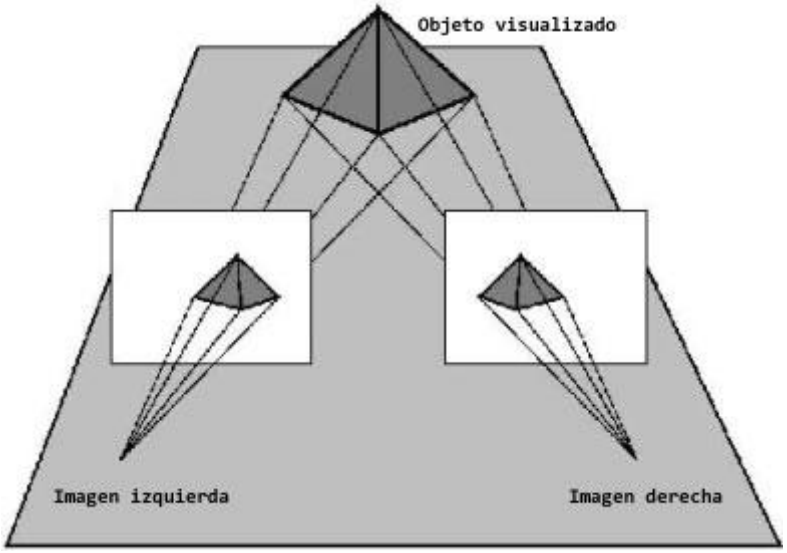

Figura 7. Funcionamiento de la visión estereoscópica

Al igual que los seres humanos tenemos dos ojos para poder tener una percepción de la profundidad en un espacio 3D, mediante una segunda cámara se puede obtener una percepción de la profundidad en el campo de la visión por computador. Este tipo de visión se conoce como visión estereoscópica.

En la literatura podemos encontrar numerosos proyectos de investigación sobre robots autónomos que utilizan cámaras estereoscópicas como sensor para controlar un robot. Concretamente y gracias al bajo coste y los buenos resultados podemos encontrar muchos proyectos (Spectrum s.f.) que usan la cámara Kinect (Microsoft s.f.) de Microsoft como subsistema de visión de un robot móvil. Esta no se trata exactamente de una cámara estereoscópica, sino que además de una cámara RGB dispone de un sensor de profundidad con el cual se obtiene la distancia a la que se encuentran los distintos objetos que aparecen en la escena.

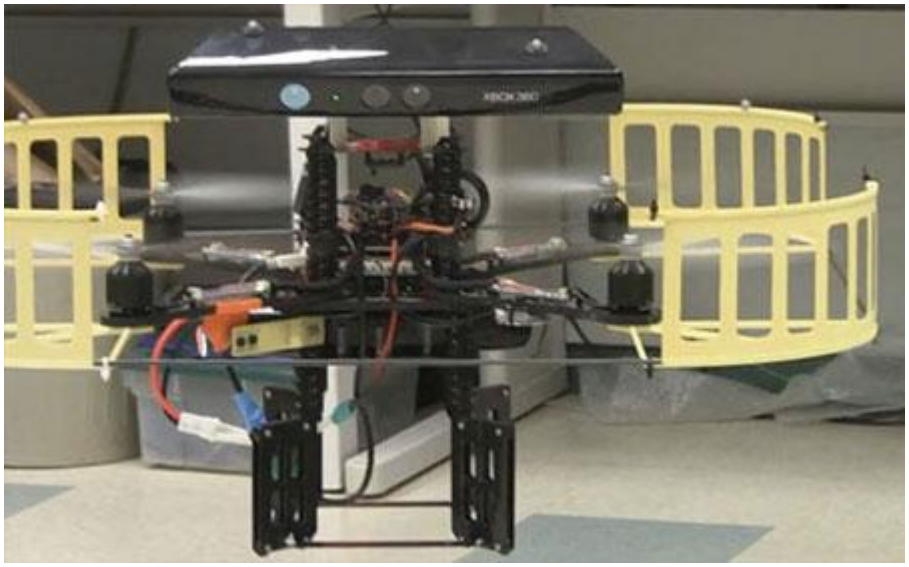

Figura 8 Robot autónomo que usa el sensor kinect para la navegación

En el comienzo de este proyecto no existía en el mercado ningún drone a un coste asumible que dispusiese de una cámara estéreo ó cámara volumétrica como la Kinect, por lo que se ha utilizado un drone con una única cámara con las limitaciones que ello supone, como la imposibilidad de estimar la distancia a la que se encuentra un objeto de tamaño desconocido. Por otro lado, es posible estimar la distancia a la que se encuentra un objeto si conocemos previamente su tamaño y tenemos la cámara calibrada.

Los equipos de investigación que trabajan con cuadricópteros para en el estudio de planificación de trayectorias y control de los cuadricópteros, suelen disponer de espacios habilitados como el que podemos ver en (Sergei Lupashin 2014). Este equipo dispone de un espacio totalmente sensorizado en donde se conoce la posición exacta de todos los cuadricópteros en cada momento. Con estos datos es posible realizar un control preciso del vehículo dentro del espacio limitado.

En nuestro caso, hemos utilizado un drone comercial con una única cámara con lo cual no podemos tener una lectura fiable de la posición en la que se encuentra el cuadricóptero ni de la distancia existente entre el objeto o persona a seguir y el vehículo.

2.2.2 Tecnologías de visión artificial

Dentro de las tecnologías de visión artificial, la comunidad científica se divide entre dos grandes plataformas, que son, Matlab Vision System Toolbox[5] y OpenCV[6].

[5] http://es.mathworks.com/products/computer-vision/
[6] http://opencv.org/

2.2.2.1 Matlab Vision System Toolbox

El entorno de desarrollo de Matlab (Matlab s.f.) ofrece una serie de herramientas para el desarrollo de aplicaciones de visión por computador. Dentro de la comunidad científica este entorno de desarrollo se ha popularizado por la facilidad para desarrollar prototipos de forma rápida ya que utilizando el lenguaje de Matlab el usuario se abstrae de cuestiones como la reserva de memoria RAM.

Dentro de Matlab disponemos de un gran número de algoritmos de visión, esto unido a la facilidad de desarrollo de prototipos, ha conseguido que Matlab sea una herramienta muy utilizada en la comunidad científica en labores de visión por computador.

Sin embargo, Matlab es un software de pago y los prototipos desarrollados en esta plataforma son difícilmente integrables en aplicaciones industriales debido a los diferentes requisitos que puedan existir.

2.2.2.2 OpenCV

Dentro de las tecnologías de análisis de imagen la librería OpenCV es la que mayor popularidad tiene en aplicaciones industriales. OpenCV cuenta con más de 47 mil miembros activos y el total de descargas de la librería supera los 9 millones. OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que aparece su primera versión alfa en el mes de enero de 1999, esta se utiliza en infinidad de aplicaciones. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

La librería OpenCV tiene diferentes interfaces para la programación de aplicaciones en distintos lenguajes (C++, C, Python y Java). El mayor potencial de OpenCV se basa en su diseño para optimizar la eficiencia computacional, esto convierte a OpenCV en la librería más utilizada en aplicaciones de visión por computador en tiempo real.

Si comparamos la misma aplicación desarrollada sobre OpenCV y Matlab, obtendremos mejores resultados de ejecución en OpenCV, sin embargo, el desarrollo suele ser más costoso utilizando esta librería ya que hay que el desarrollador se enfrenta a los problemas inherentes del lenguaje de programación C sobre el que está implementada la librería (Slavomir Matuska 2012).

2.2.3 Tracking de personas

Podemos encontrar en la literatura distintos tipos de soluciones para trackear o seguir a personas mediante análisis de imagen en vídeos.

El propio OpenCV dentro de los ejemplos de la librería tiene un detector de cuerpo humano en imágenes, este funciona a través del descriptor HoG (histogram of oriented gradients) (Triggs 2005). Este descriptor es popular a la hora de detectar objetos en imágenes. El descriptor HOG se basa en generar un vector de características a partir del cálculo de los gradientes de intensidad en la imagen en escala de grises.

El programa de ejemplo de OpenCV además del descriptor HoG, también se apoya en un clasificador SVM (OpenCV 2.4.11 s.f.) ya entrenado con vectores HoG para la detección de personas.

2.2.4 CMT

Clustering of Static-Adaptive Correspondences for Deformable Object Tracking (CMT) (Georg Nebehay 2015)es un algoritmo galardonado de seguimiento o tracking. Este tracker funciona seleccionando una zona de interés o zona a trackear.

La idea principal del funcionamiento de CMT es dividir el objeto o zona de interés en partes más pequeñas conocidas como puntos característicos. En cada frame de vídeo, se trata de encontrar de nuevo los puntos característicos que se encontraban en la zona a trackear del primer frame. Para lograr esto se utilizan dos métodos diferentes. En primer lugar se realiza un seguimiento de los puntos característicos del frame anterior al frame actual mediante la estimación del optic flow o flujo óptico. En segundo lugar, se emparejan los puntos característicos globalmente comparando sus descriptores.

Como estos dos métodos son propensos a errores, se emplea una nueva forma de buscar el consenso en los puntos característicos encontrados de forma que cada uno de estos vote por el centro del objeto, como se muestra en la siguiente figura:

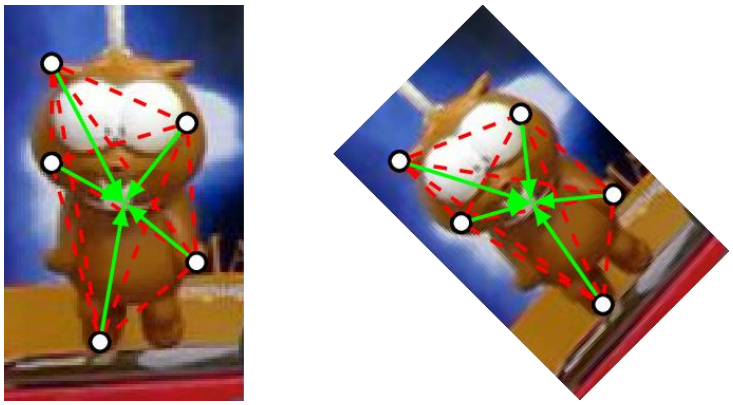

Figura 9. Puntos característicos votando por el centro

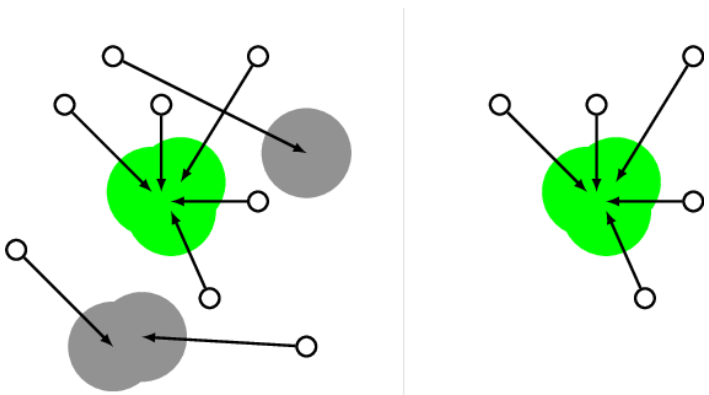

Figura 10. Consenso para buscar el centro del objeto

Los puntos característicos son puntos o píxeles que destacan en la imagen por su diferencia (nivel de intensidad) respecto a los píxeles vecinos. El conjunto de puntos característicos de una imagen nos ofrece mucha información sobre esta. Normalmente los puntos característicos de una imagen o vector característico suele usarse como entrada para entrenar u obtener información de un clasificador.

Existen distintos métodos o detectores para encontrar puntos característicos, los detectores integrados en OpenCV son: FAST, STAR, SIFT, ORB, BRISK, MSER, GFTT, HARRIS, Dense y SimpleBlob.

Explicaremos el funcionamiento de los detectores FAST y BRISK ya que son los detectores de puntos característicos que se han usado dentro del proyecto.

- Detector FAST

El detector FAST (OpenCV s.f.) es capaz de detectar características tipo esquina o corners a partir de una imagen de entrada en escala de grises. Para ello toma la intensidad del pixel a evaluar y la compara con un conjunto de 16 píxeles que lo rodean formando un círculo. Si al menos tenemos 12 píxeles vecinos cuya intensidad es mayor con un umbral que la intensidad del píxel u otros 12 píxeles con intensidad menor con el mismo umbral, el píxel es un punto característico.

- Detector BRISK: Binary Robust Invariant Sacalable Keypoints

El detector BRISK (Leutenegger 2011)es capaz de detectar características tipo esquina o corners a partir de una imagen de entrada en escala de grises. Esta detección se realiza a múltiples escalas con el fin de lograr invariancia de escala. BRISK busca los máximos no solo en el plano de imagen, sino también en el espacio de escalas. Esto se hace representando la imagen original como una familia de un parámetro de imágenes suavizadas. El parámetro corresponde a la escala de las imágenes. Por tanto, lo que se obtiene en este caso es una

familia consistente en la imagen original reescalada a distintos tamaños. La forma piramidal de esta familia hace que se conozca como pirámide del espacio de escalas. BRISK se apoya en el algoritmo FAST para la búsqueda de máximos en cada escala posteriormente, para descartar puntos estos se comparan con la escala superior y la inferior.

2.3 Dinámica de los cuadricópteros

En este apartado realizaremos una introducción a la dinámica y control de los cuadricópteros (Quadcopter Dynamics s.f.). Como ya sabemos, la dinámica de los cuadricópteros se define según el empuje de sus cuatro rotores. Como la mayoría de los vehículos aéreos, su movimiento está definido a través de tres ángulos conocidos como ángulos de Euler que son Roll, Pitch y Yaw. Como podemos ver en la Figura 11, el roll define el movimiento lateral del vehículo, el Pitch define el movimiento hacia delante y hacia atrás, mientras que a través del Yaw se define el giro del cuadricóptero respecto al eje Z. El movimiento vertical se define como Gaz y es equivalente al empuje de los cuatro rotores Figura 11 (caso a).Los cuatro motores son equidistantes tanto uno del otro como del centro mismo de la estructura, lo cual lo convierte en un vehículo simétrico, en el que cada uno de los motores produce empuje y par, es por ello que para asegurar que el vehículo se eleve y pueda desplazarse en el aire es necesario que el sentido de giro de los motores y el tipo de hélice a utilizar en cada motor correspondan a una de las configuraciones ilustradas en la Figura 11. Finalmente, esta simetría permite un mejor control, desplazamiento y estabilidad en el vehículo. La dinámica de la aeronave, al tratarse esta de un cuerpo sólido tridimensional, puede modelarse matemáticamente como una fuerza y tres momentos.

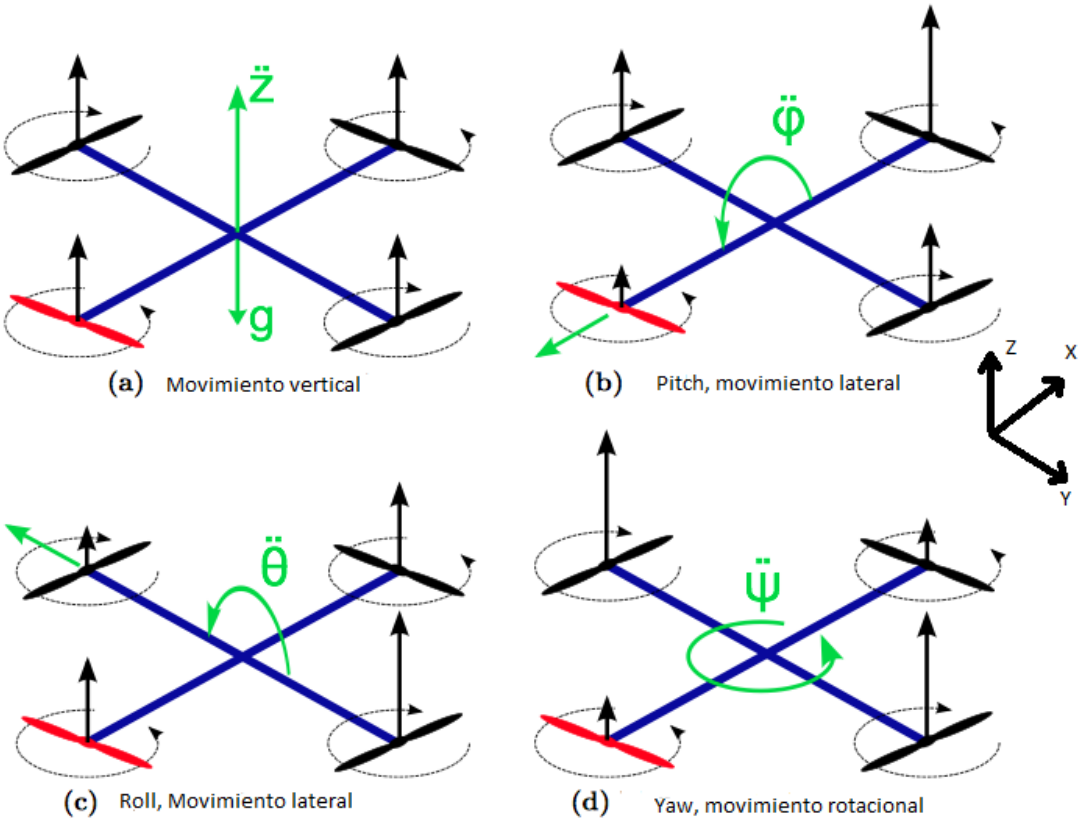

Figura 11. Movimientos del cuadricóptero

Bajo este esquema las coordenadas X, Y y Z representan la posición del centro de masa de la aeronave en relación a su estructura, mientras que θ, ϕ y ψ representan los ángulos de Euler: Roll o θ, determina la rotación de cuadricóptero alrededor del eje Y (caso c). El Pitch o ϕ determina la rotación alrededor del eje X (caso b) y por último el Yaw o ψ alrededor del eje Z (caso d), que representan la orientación del cuadricóptero.

El control de los ángulos Roll y Pitch es imprescindible para la estabilización del vehículo en una posición determinada. Este problema de estabilización ya viene resuelto por el fabricante de los cuadricópteros comerciales.

A continuación veremos las ecuaciones de la dinámica de un cuadricóptero de forma genérica:

Definimos la posición del centro de masas del cuadricóptero como $x = (x, y, z)^T$ y la velocidad del centro de masas como $v = \dot{x} = (\dot{x}, \dot{y}, \dot{z})^T = (V_x, V_y, V_z)^T$. Por otro lado tenemos los ángulos de Euler $\theta = (\phi, \theta, \psi)$ y su derivada en el tiempo $\dot{\theta} = (\dot{\phi}, \dot{\theta}, \dot{\psi})$. Dadas las características del cuadricóptero, utilizaremos dos sistemas de referencia, el primero, el sistema inercial dado por la posición del centro de masas del vehículo desde el punto del observador y el otro el sistema de referencia del propio vehículo que viene determinado por los ángulos de Euler.

Podemos relacionar el sistema de referencia inercial con el del cuadricóptero a través de la matriz R que transforma elementos del sistema de referencia del drone al sistema inercial (Quadcopter Dynamics s.f.) (Luukkonen 2011):

$$R = \begin{bmatrix} C_\phi C_\psi - C_\theta S_\phi S_\psi & -S_\phi C_\psi - C_\theta C_\phi S_\psi & S_\theta S_\psi \\ C_\theta S_\phi C_\psi + C_\phi S_\psi & C_\theta C_\phi C_\psi - S_\phi C_\psi & -S_\theta C_\psi \\ S_\theta S_\phi & S_\theta C_\phi & C_\theta \end{bmatrix}$$

Ecuación 1. Matriz de roación entre el sistema de referencia del drone y el inercial

Donde C_x y S_x representan Cos(x) y Sin(x) respectivamente.

Esta matriz de rotación se obtiene a través de los ejes XYZ y los ángulos de Euler row, pitch y yaw. Tenemos que la matriz es ortonormal siendo $R^{-1}=R^T$, siendo R^T la rotación entre el sistema de referencia del cuadricóptero y el sistema inercial.

Una vez tenemos definidos los sistemas de referencia definimos las fuerzas y pares que actúan sobre el cuadricóptero para poder modelar las ecuaciones de movimiento del cuadricóptero.

Cada uno de los motores del cuadricóptero tiene una velocidad angular w_i que genera una fuerza f_i en la dirección del eje del rotor. La velocidad angular y la aceleración del rotor también crean un par T_M sobre el eje del rotor:

$$f_i = kw_i^2$$
$$T_{M_i = bW_i^2} + I_M \dot{w}_\iota$$

Ecuación 2. Fuerzas y par de los rotores del cuadricóptero

Donde k y b son constantes características del tipo de rotor e I_M es el momento de inercia del rotor. Podemos descartar el efecto de $\dot{w}_\iota$ al considerarse pequeño.

La fuerza combinada de los cuatro rotores genera un empuje T en la dirección del eje z.

$$T = \sum_{i=1}^{4} f_i = k \sum_{i=1}^{4} w_i^2$$

Ecuación 3. Empuje del cuadricóptero en el eje z

Teniendo en cuenta el sentido de giro de cada rotor, el par total en el eje z viene dado por:

$$\tau_\psi = k(w_1^2 - w_2^2 + w_3^2 - w_4^2)$$

Ecuación 4 Par ejercido en el eje z

Los pares sobre el roll y pitch se derivan de la mecánica estándar. Podemos seleccionar arbitrariamente los rotores 1 y 3 para que sean el eje sobre el que actúa el roll, entonces tenemos:

$$\tau_\theta = Lk(w_1^2 - w_3^2)$$

Ecuación 5. Par ejercido, roll

Donde L es la distancia desde el centro del cuadricóptero a cualquiera de los rotores, recordemos que lo consideremos simétrico.

Y su correspondiente en el eje del pitch:

$$\tau_\phi = Lk(w_2^2 - w_4^2)$$

Ecuación 6. Par ejercido, Pitch

Uniendo las 3 ecuaciones anteriores tenemos el vector de par desde el sistema de referencia del cuadricóptero como:

$$\tau_q = \begin{bmatrix} Lk(w_1^2 - w_3^2) \\ Lk(w_2^2 - w_4^2) \\ k(w_1^2 - w_2^2 + w_3^2 - w_4^2) \end{bmatrix} = \begin{bmatrix} I_y\ddot{\theta} \\ I_x\ddot{\phi} \\ I_z\ddot{\psi} \end{bmatrix}$$

Ecuación 7. Vector de par de fuerzas en el sistema de referencia del vehículo

Donde I_x, I_y e I_z representan los momentos de inercia en el eje x, y, z respectivamente.

Para obtener las ecuaciones de la dinámica del cuadricóptero hacemos uso de la segunda ley de Newton:

$$m\ddot{x} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ k\sum_{i=1}^{4} w_i^2 \end{bmatrix}$$

Ecuación 8 Segunda ley de Newton en el caso del cuadricóptero

Donde x es el vector de posición (x,y,z) y R la matriz de rotación entre el sistema de referencia inercial y el del cuadricóptero. Despejando las ecuaciones para cada eje

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m} \begin{bmatrix} S_\theta C_\phi C_\psi + S_\phi S_\psi \\ S_\theta C_\phi S_\psi - S_\phi C_\psi \\ C_\theta C_\phi \end{bmatrix}$$

Ecuación 9. Dinámica del cuadricóptero

Suponiendo que ya tenemos el Yaw controlado y apuntando al objetivo, asumimos sin pérdida de generalidad que $\psi = 0$. Si no actuamos sobre el roll y no tenemos encuenta el efecto del viento ni turbulencias que afecten al drone tenemos que $\theta = 0$. Si estamos frente al objetivo y colocamos el sistema de referencia inercial del drone de forma que x=0.

Teniendo en cuenta lo anterior y a través de la Ecuación 9., tenemos que la aceleración del drone moviéndose hacia delante será proporcional al seno del ángulo del pitch. Por lo tanto nuestro sistema de control únicamente debe actuar sobre el pitch. Si orientamos correctamente el vehículo, tenemos que:

$$\ddot{x} = Ksin(\phi)$$

Si linealizamos suponiendo que el pitch es pequeño:

$$\ddot{x} = K\phi$$

Ecuación 10 Ecuación de control de la distancia absoluta al objetivo

Por otro lado al estar el cuadricóptero a una altura constante consideramos su velocidad y aceleración igual a cero, $\dot{z} = \ddot{z} = 0$ y aplicándolo a la Ecuación 9. tenemos:

$$0 = -g + \frac{T}{m}Cos(\phi)$$

Considerando que ϕ es pequeño obtenemos que el valor de la constante K es igual a la constante gravitatoria que:

$$K = \frac{T}{m} = g$$

Por último si consideramos que el objetivo está quieto a una distancia *d* del drone tendremos que:

$$\ddot{d} = g\phi$$

Ecuación 11 Ecuación de control de la distancia absoluta al objetivo

Como vemos nos encontramos ante un sistema de control de doble integrador. A partir de la ecuación anterior, podemos ver que la función de transferencia de del modelo del pitch a controlar es:

$$G(s) = \frac{g}{s^2}$$

Ecuación 12 Función de transferencia para el control del pitch

Como podemos comprobar este tipo de modelo se puede estabilizar utilizando un controlador PID. Analizando el sistema en lazo abierto vemos que este tiene dos polos en el origen que van al infinito a lo largo del eje imaginario.

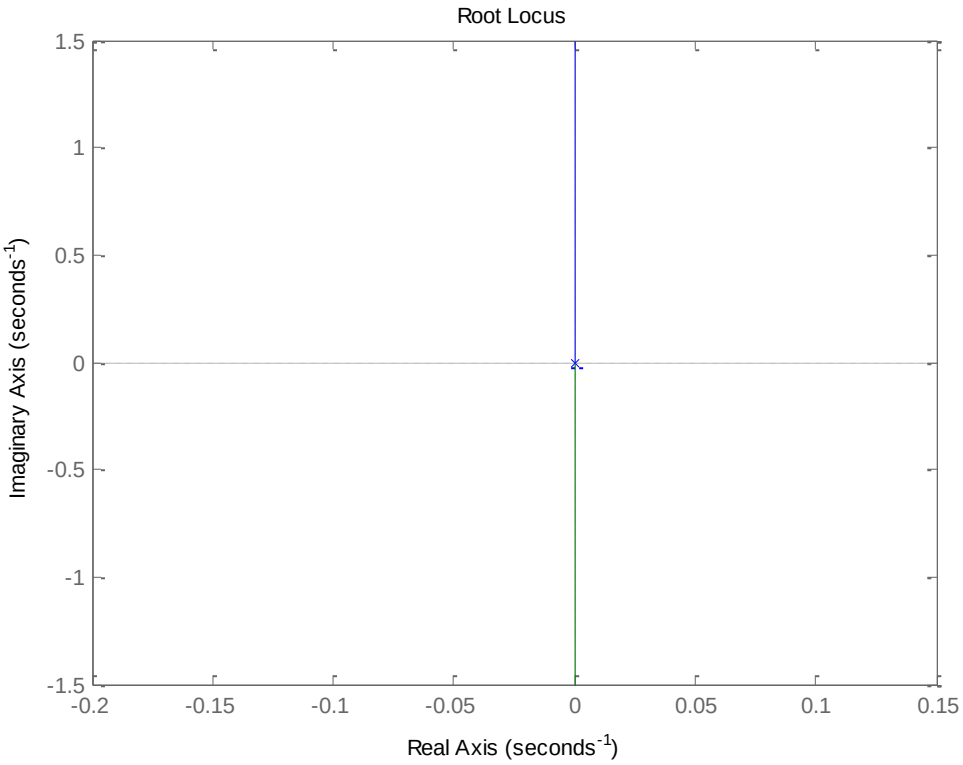

Figura 12. Polos del sistema en lazo abierto

Recordemos que la función de transferencia de un controlador PID viene dada por:

$$PID(s) = k_p(1 + \frac{1}{T_i s} + T_d s)$$

Ecuación 13 Función de transferencia del controlador PID

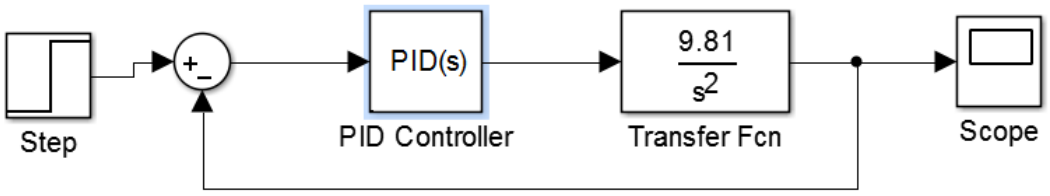

Figura 13. Diagrama de bloques del sistema

Si utilizamos únicamente el control proporcional con K_p=10 vemos que el sistema es inestable:

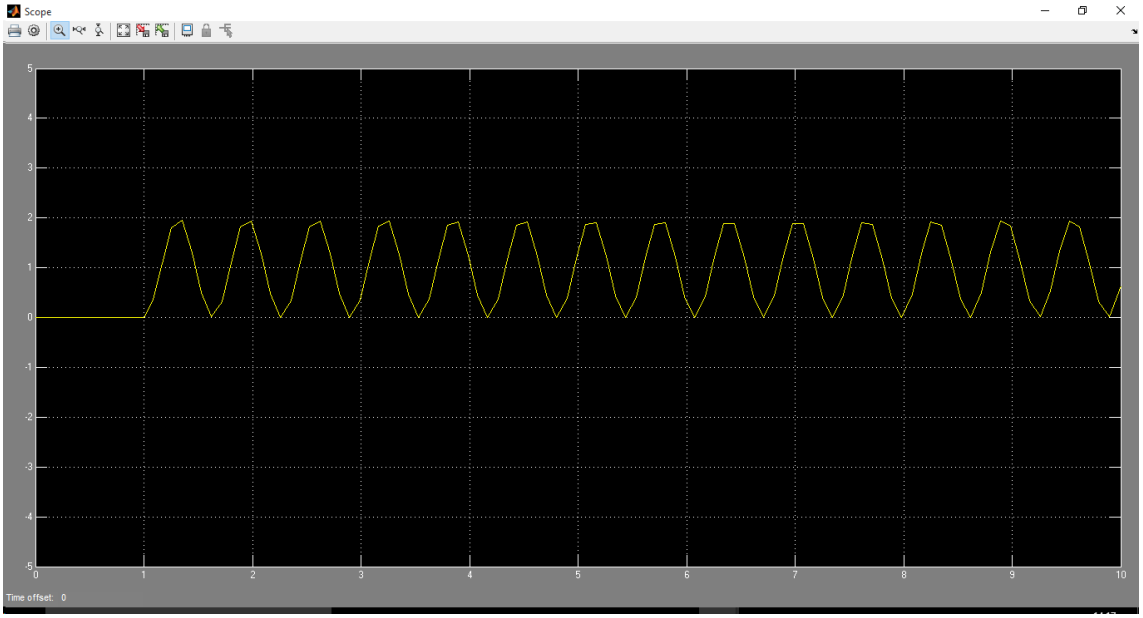

Figura 14. Sistema inestable con control proporcional

Si añadimos el control derivativo con K_d=1 tenemos que el sistema está estabilizado.

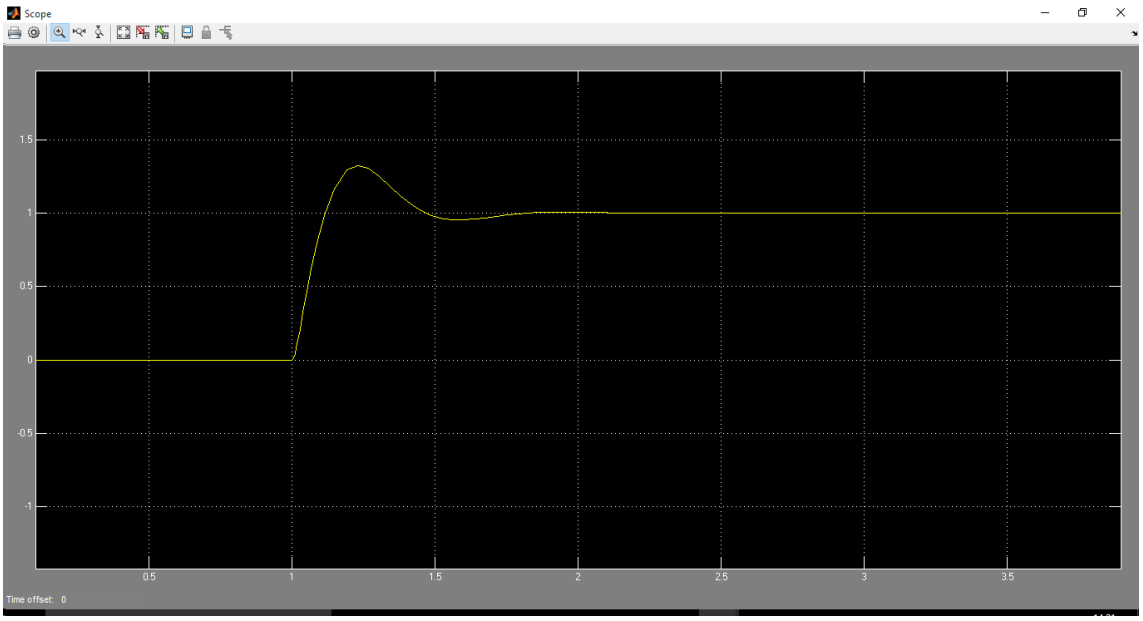

Figura 15. Respuesta escalón con control PD

Podemos comprobar que al introducir el controlador derivativo los polos del sistema se encontrarán en el eje real negativo, por lo tanto el sistema es estable:

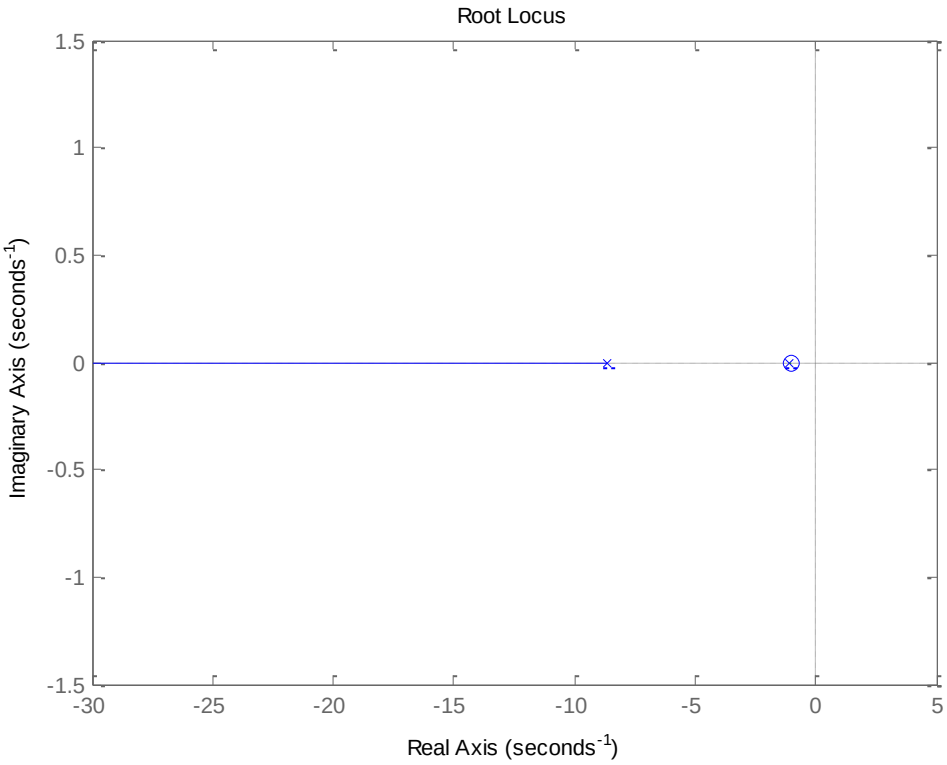

Figura 16. Polos del sistema con un controlador PD

Como hemos comprobado, el sistema se puede estabilizar con un controlador PD por lo tanto podremos controlar el Pitch de nuestro drone en el desarrollo del proyecto.

Para el control del Yaw tenemos a partir de la Ecuación 7 que la aceleración del Yaw es dependiente del empuje dado por los cuatro motores del cuadricóptero:

$$\ddot{\psi} = \frac{k}{I_z}(w_1^2 - w_2^2 + w_3^2 - w_4^2)$$

Ecuación 14 Dinámica del Yaw

Si consideramos ese empuje como una variable de nuestro sistema y podemos controlar esta variable y dado que el cuadricóptero es un sólido rígido su momento de inercia en el eje z será una constante, entonces tenemos:

$$\ddot{\psi} = K\gamma$$

Ecuación 15 Dinámica del Yaw

Por lo que nos encontramos ante otro sistema de control de doble integrador y como hemos demostrado para el control de la distancia este sistema es estable utilizando un controlador PD.

3 Desarrollo

3.1 Cuadricóptero seleccionado: Bebop Drone

Como ya hemos adelantado en el apartado del Estado del arte, el drone elegido para el desarrollo del proyecto ha sido el Bebop Drone de Parrot. La elección de este drone para el desarrollo del proyecto viene dada por la relación calidad de imagen/precio y la disponibilidad de un SDK con el que poder desarrollar los objetivos del proyecto. En este apartado vamos a ver con mayor profundidad alguna de las características de este drone.

El Bebop Drone de Parrot es un drone robusto y ligero, gracias a su composición de ABS (Acrilonitrilo butadiene estireno) reforzado con fibras de vidrio, únicamente pesa 400 gramos y puede soportar caídas y choques.

Este drone tiene un procesador dual core Parrot P7 para el cálculo de la telemetría y estabilización. Por otro lado, tiene un procesador gráfico Quad Core con un estabilizador de imagen digital verdaderamente fiable.

Para garantizar una estabilidad óptima sin comprometer la manejabilidad, Bebop Drone realiza una fusión de datos provenientes de numerosos sensores de 3 ejes: acelerómetro, giroscopio y magnetómetro. Un sensor de ultrasonidos de un alcance de 8 m, un sensor de presión, así como una cámara vertical miden la velocidad de desplazamiento.

Parrot Bebop Drone viene equipado con una cámara de fotos de 14 Megapíxeles con un objetivo "fisheye" (ojo de pez), realiza vídeos y fotos en 180° de una calidad extraordinaria. Con una estabilización de imagen completamente digital, Bebop Drone le permite filmar imágenes desde el aire, cualesquiera que sean los movimientos inherentes de los cuadricópteros. Es resiste a las salpicaduras y al polvo. Las cámaras deportivas, a menudo, muestran distorsiones en las líneas horizontales. Gracias al objetivo fisheye y al algoritmo de estabilización de imagen, la cámara de Parrot Bebop Drone no distorsiona la imagen. En consecuencia, el horizonte no sufre el efecto de curvatura como suele ocurrir en las cámaras deportivas.

Figura 17. Estabilización de imagen del Bebop Drone

3.1.1 SDK de desarrollo

Parrot publicó en el repositorio Github[7] un SDK de desarrollo para el Parrot Bebop Drone y sus mini drones, Jumping Sumo y Rolling Spider. Este SDK contiene las librerías necesarias para poder desarrollar todas las funcionalidades que podemos encontrar en la aplicación de control que proporciona Parrot para el control de su familia de drones. Esta aplicación es FreeFlight3 y la podemos encontrar en GooglePlay y el AppStore.

Lamentablemente, la escasa documentación proporcionada por Parrot para su SDK y que éste sea el mismo para diferentes drones dificultaba su uso en el desarrollo de este proyecto. Por este motive se han buscado alternativas en la comunidad de desarrolladores. De esta forma se encontró el SDK finalmente usado. La librería que se ha usado para el control del Parrot está escrita en Python y es proporcionada por un investigador checo llamado Martin Dlouhý[8] (Dlouhý s.f.). Este SDK no oficial tiene implementadas las funcionalidades básicas para el manejo del drone, desde la decodificación del streaming de video a los controles del pilotaje. También se encuentra dentro de este SDK algo más de información que este investigador ha recopilado realizando ingeniería inversa.

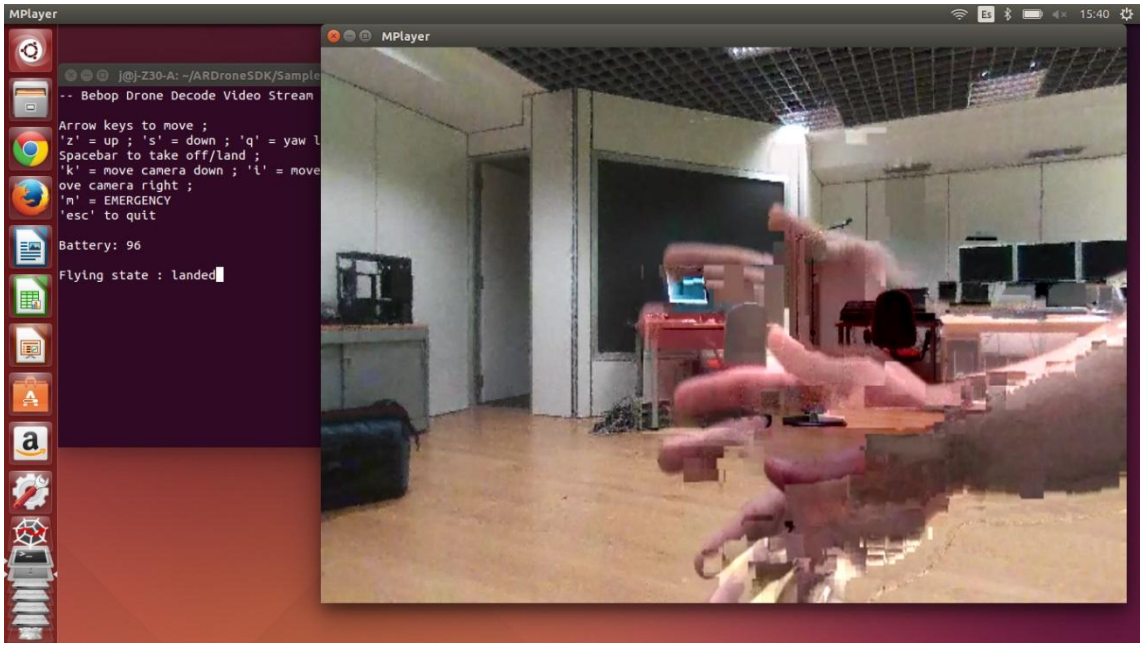

Figura 18. Ejecución del ejemplo del SDK proporcionada por Parrot

Por otro lado, al probar la aplicación de ejemplo que proporciona Parrot en su SDK, tal y como podemos ver en la figura anterior, realiza una mala decodificación del streaming de video. Con un streaming de esas características sería imposible aplicar un tracking al vídeo.

[7] https://github.com/Parrot-Developers
[8] https://github.com/robotika/katarina

3.1.2 Protocolo de comunicaciones del Bebop Drone

En la documentación del SDK oficial (Parrot s.f.) no encontramos demasiada información del protocolo de comunicaciones del Bebop Drone (Developers s.f.), sin embargo, gracias al trabajo de ingeniería inversa de distintos investigadores que están trabajando con el Bebop Drone podemos encontrar la información necesaria:

El Drone genera una red wifi a la que el cliente que ejecuta la aplicación de control se tiene que conectar. La comunicación entre el Drone y la aplicación de control se basa en UDP salvo el descubrimiento del drone que utiliza TCP.

Se utilizan dos puertos para la comunicación entre el drone y la aplicación de control, estos puertos son el 54321 para el envío de comandos al drone y el Puerto 43210 para el envío de los datos de navegación del drone a la aplicación.

Todos los datos se transfieren a través de estos puertos, incluido el streaming de video. Únicamente tenemos un Puerto para enviar datos desde la aplicación y otro para recibirlos. Los diferentes tipos de paquetes de datos se identifican a través de los dos primeros bytes, los tipos de datos que se pueden recibir son:

- Datos estándar: la telemetría que envía el drone
- Datos que necesitan un ACK de vuelta
- Datos de video: fragmentos de frames codificados en H.264
- Datos Ping-Pong: utilizados por el drone para conocer el estado de la comunicación

Cabecera							Datos								
Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	.....	Byte n
Tipo	ID	Sec	Tamaño del paquete				Datos								

Tabla 1. Protocolo de comunicaciones

La cabecera de cada paquete tiene 7 bytes de longitud e incluye una secuencia creciente de cada tipo paquete. El resto de bytes depende del tipo de paquete a enviar y el mensaje enviado.

Comandos de movimiento:

Estos vienen definidos por el SDK oficial en un archivo XML, entre estos comandos podemos encontrar comandos de despegue, aterrizaje, etc.: en el CD adjunto a la memoria podemos encontrar el archivo XML con los comandos descritos en el SDK dentro de la ruta D:\documentacion\comandos.xml.

Datos de posición y dinámica:

Los datos de telemetría que envía el drone tienen una estructura similar a la de los comandos. Los datos de telemetría que recibimos del drone son:

- Roll, Pitch y Yaw (en radianes)
- Altitud (en metros)
- Posición X,Y,Z (en metros respecto el punto de despegue)
- Velocidad V_x, V_y, V_z (en metros/Segundo)

Estos datos son medidos a través de los múltiples sensores que son fusionados por el propio drone. Estos sensores son: giróscopo, acelerómetro, barómetro, GPS y cámara vertical.

Como se trata de una fusión de sensores, la fiabilidad de los datos de posición y velocidad no es demasiado alta como para que el control del drone se base en ellos.

Datos con solicitud de ACK:

Cada cierto tiempo el drone envía datos de telemetría y solicita un ACKNOWLEDGEMENT que hay que devolver al drone para que la comunicación entre el drone y la aplicación de control sea fluida.

Datos Ping-Pong:

Aproximadamente dos veces por Segundo el drone envía un paquete de datos de Ping, este debe ser respondido (Pong) generando un paquete con la marca de tiempo dada en el ping. Estos datos son usados por el drone para comprobar el estado de la conexión.

Comandos de movimiento PCMD:

Los comandos del movimiento del drone incluyen valores para el control del Roll, Pitch, Yaw y Gaz, todos ellos con valores que van desde -100 a 100.

La velocidades máximas de vuelo se determinan enviando comandos especiales al drone, estos comandos permanecen en la memoria del drone una vez son enviados. Estos comandos sirven para establecer la velocidad máxima de subida y la inclinación máxima que nos equivaldría a las velocidades máximas de Pitch y Roll. La velocidad de giro máxima en Yaw también se define mediante un comando especial.

Una característica especial del envío de este tipo de comandos es que una vez se comienza a enviar estos, es necesario que se envíen con una frecuencia fija de 40Hz, en caso contrario, el drone estima que existe una mala conexión y deja de enviar paquetes de datos de video.

Datos de video:

Los frames de video codificados en H.264 se envían en fragmentos de aproximadamente 1Kb. Cada paquete de datos de video contiene información de si se trata de un frame I (frame con una imagen completa) o P (frame que necesita un frame I para poder codificarlo), del número de frame al que pertenece y del número total de fragmentos que forman el frame actual. Una vez recibido cada frame es necesario enviar un paquete especial de ACK.

3.1.3 Análisis del protocolo

Dado que la documentación sobre el protocolo de comunicaciones del Bebop Drone es escasa, una herramienta para analizar la secuencia de comandos necesarios para pilotar el drone ha sido de una gran ayuda. Esta herramienta ARDRONE3 PCAP (ARDrone-PCAP s.f.) ha sido desarrollada por la comunidad de programadores de software libre y se basa en la librería WinPcap al igual que el analizador de tráfico IP Wireshark[9].

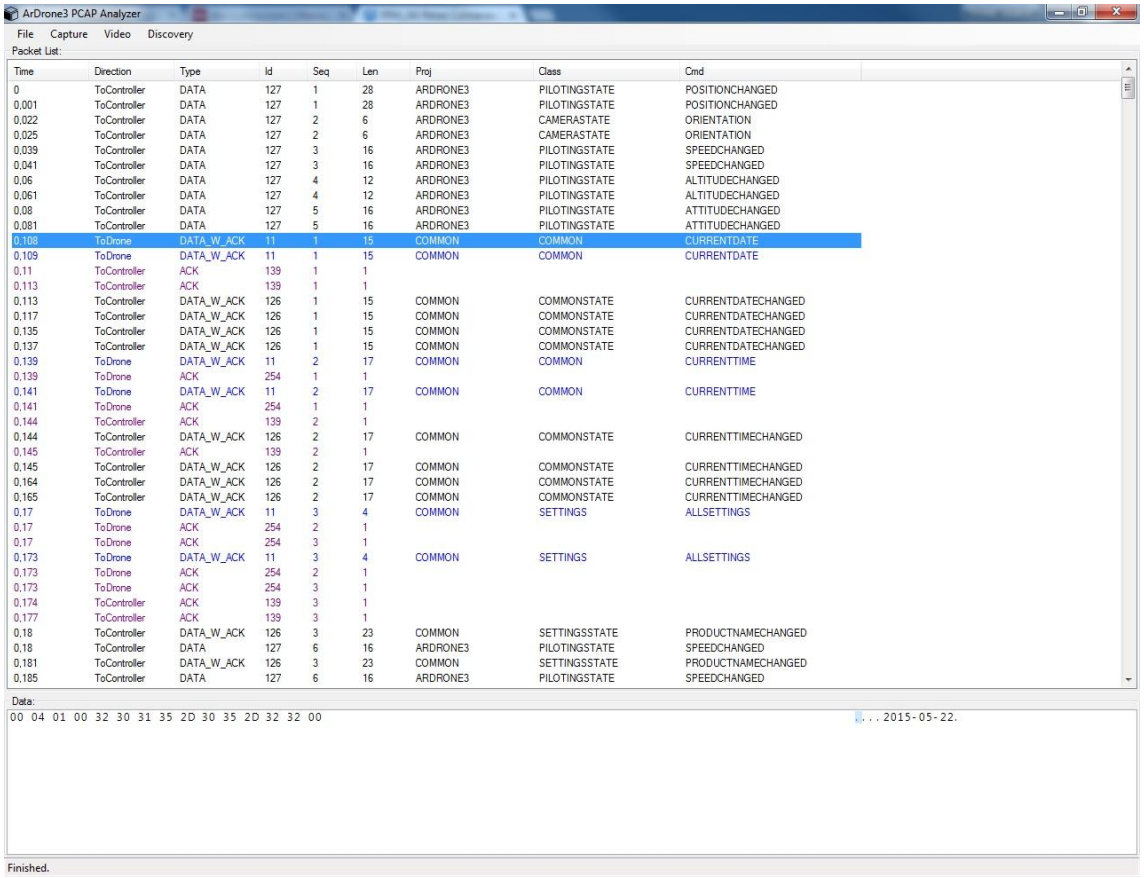

Figura 19. Analizador Ardrone3 PCAP, análisis de aplicación FreeFlight

El analizador del protocolo del Bebop Drone ha sido de gran ayuda para poder estabilizar el drone. Para conocer exactamente la secuencia de comandos que la aplicación de control envía al drone, se ha realizado una captura del tráfico entre la aplicación oficial de Parrot para Android y el drone. Posteriormente analizando este tráfico se ha podido comprobar que la secuencia de comandos no era la correcta y por este motivo el drone no se estabilizaba automáticamente.

[9] https://www.wireshark.org/

3.2 Herramientas utilizadas

En este apartado realizaremos un repaso a las herramientas utilizadas para el desarrollo del proyecto. En primer lugar, el lenguaje de programación utilizado ha sido Python. Como sabemos, Python es un lenguaje de programación interpretado.

3.2.1 IDE de desarrollo

Ya que el lenguaje de programación utilizado ha sido Python el IDE de desarrollo utilizado ha sido Spider, que viene como IDE por defecto en la distribución de Python Anaconda. Se ha utilizado esta distribución desde un principio ya que incluye numerosas librerías de Python, con lo que la integración de las diferentes dependencias necesarias para el proyecto se simplifica.

3.2.2 FFmpeg

FFmpeg es un proyecto libre para la decodificación, codificación y streaming de vídeo y audio. Su uso dentro del proyecto está directamente ligado a la decodificación de los frames H.264 que envía el drone. Para su integración con Python, se ha utilizado código proporcionado por el SDK no oficial utilizado (Dlouhý s.f.).

3.2.3 OpenCv 2.4.10

OpenCV es una librería libre de visión artificial. Su uso en el proyecto está directamente ligado al uso del tracker CMT. Por otro lado también se usa esta librería para poder visualizar los frames decodificados que envía el drone.

3.3 Set Up del entorno de desarrollo

En el primer periodo del proyecto, el sistema operativo utilizado ha sido Windows, concretamente Windows 8.1, pero después de las primeras pruebas llevadas a cabo, se ha podido comprobar que existe algún problema en la gestión de las comunicaciones a través de sockets gestionados por Python en Windows. Por este motivo se perdía cada cierto tiempo el streaming de vídeo con lo cual el tracking era inviable.

Como todo el software está basado en Python y este lenguaje es independiente del sistema operativo se probó la aplicación en Ubuntu 14.04 64 bits, obteniendo un streaming con mayor

estabilidad que el que se tenía en Windows 8.1, por este motivo todas las herramientas utilizadas se han compilado e integrado en Ubuntu.

En el Anexo 1 podemos encontrar un pequeño manual con los pasos para la instalación de todas las dependencias del proyecto.

4 Arquitectura y funcionamiento de la aplicación

En la siguiente figura podemos ver el funcionamiento de la aplicación de control del Bebop Drone.

La aplicación desarrollada dentro del proyecto es una aplicación multiproceso donde tenemos tres hilos principales:

- Gestión de comunicación: este hilo se encarga de la comunicación con el Drone, recibiendo los datos de telemetría y enviando los paquetes de datos necesarios para una correcta comunicación, ACK y Pong. Este hilo también se encarga de enviar los comandos de movimiento con la frecuencia exacta de 40Hz.
- Decodificación de vídeo: este hilo procesa los datos de vídeo que llegan desde la gestión de la comunicación. Cada vez que llega un fragmento de vídeo, este módulo se encarga de su procesado, decodificación y lo almacena en una cola para que el tracker pueda utilizarlo.
- Tracking: este es el módulo principal, sigue en cada frame el objeto que hayamos seleccionado. Si el objeto se encuentra en el frame calcula el movimiento que ha de realizar el drone y genera el comando necesario.

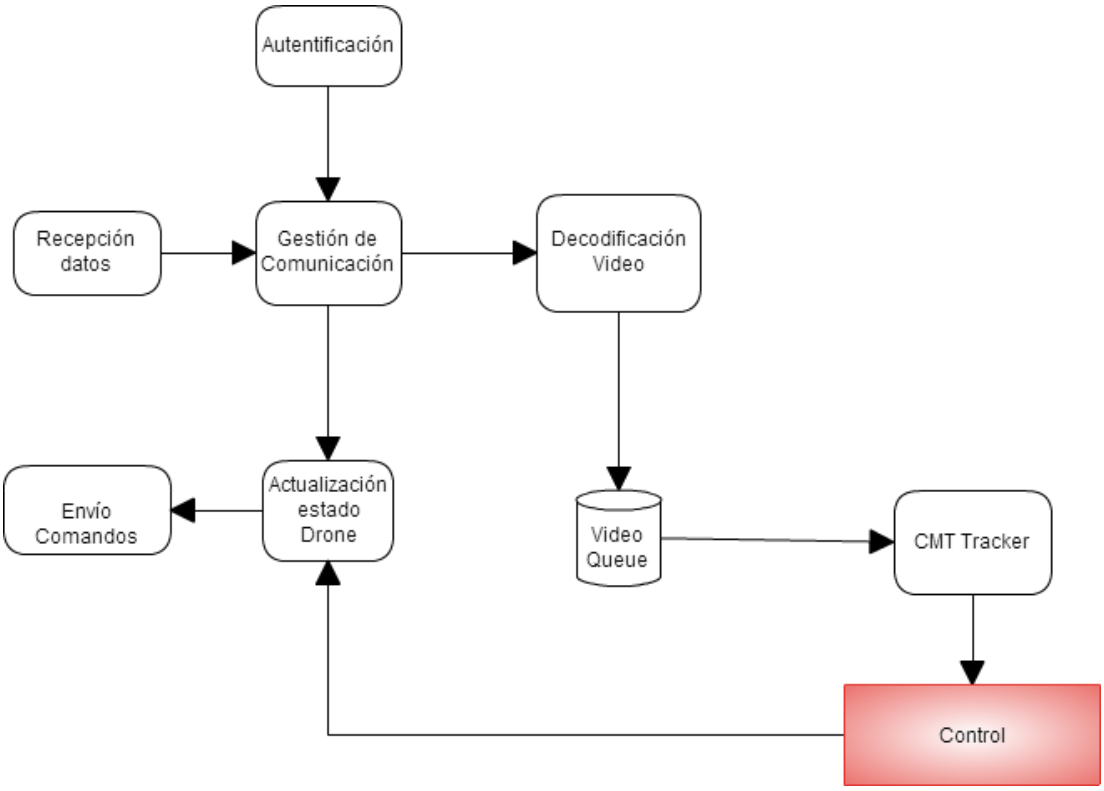

Figura 20. Funcionamiento de la aplicación de control

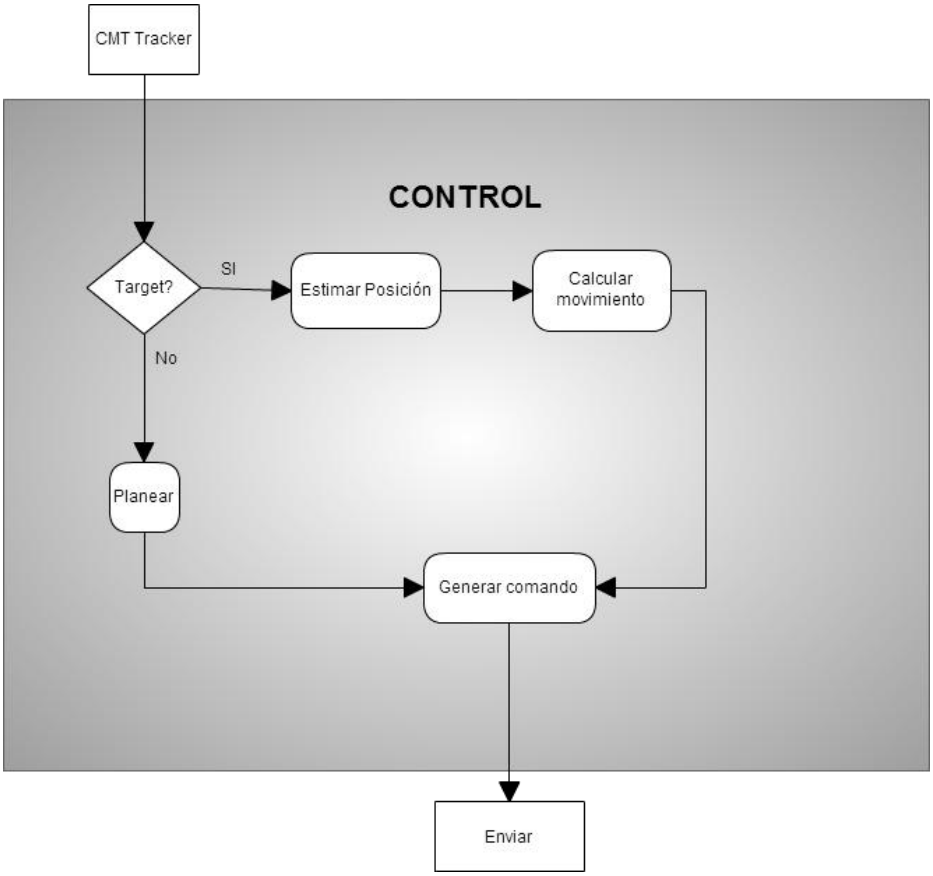

Figura 21. Funcionamiento del sistema de control del drone

4.1 Funcionamiento de la aplicación

En primer lugar el PC debe estar conectado a la red wifi que genera el drone, en este caso la red wifi que genera el drone es BebopDrone-L029058. Una vez se ejecuta la aplicación se inicializa la comunicación entre el drone y la aplicación. Los primeros comandos que se envían son los de establecer la fecha y hora del drone, esto es necesario para sincronizar ambos aparatos. Los siguientes comandos son requeridos por el drone para una correcta comunicación, en estos comandos se le solicitan el estado de toda la telemetría y por otro lado se envía la configuración de serie del drone. Existen elementos de la configuración que se quedan almacenados en la memoria del drone, entre estos datos encontramos desde el nombre del Drone, en este caso BebopDrone-L029058, el número de serie del drone, la versión de software que tiene instalada y los parámetros de velocidades máximas.

Una vez se encuentra establecida la comunicación, se inicializan los distintos hilos que forman parte de la aplicación. El drone despega y se establece a una altura fija, un metro en pruebas de laboratorio, que es el despegue normal del drone y 3 metros en las pruebas en el exterior.

Pasados unos segundos cuando ya tenemos activo el streaming de vídeo comenzamos a visualizar el streaming que realiza el drone en la pantalla, si tenemos la ventana de vídeo activa se abrirá otra ventana con un frame congelado donde podremos seleccionar el objeto o persona que queremos trackear.

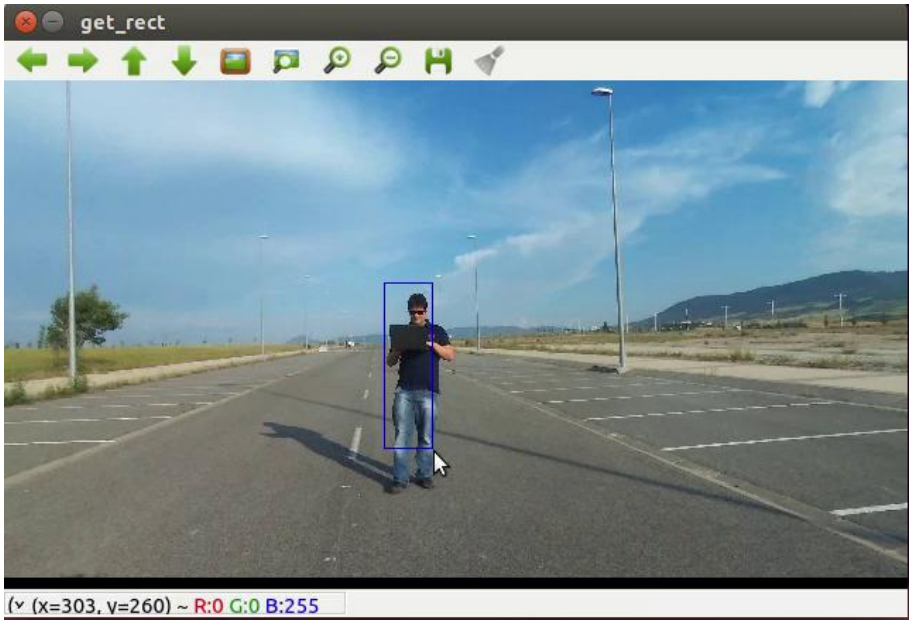

Figura 22. Selección del objeto o persona a trackear

Cuando hemos seleccionado un objeto en el frame, se inicializa el tracker CMT, este tracker si estima que el objeto está en la escena nos devuelve el tamaño del rectángulo y su centro donde considera que se encuentra la zona de interés que previamente hemos seleccionado.

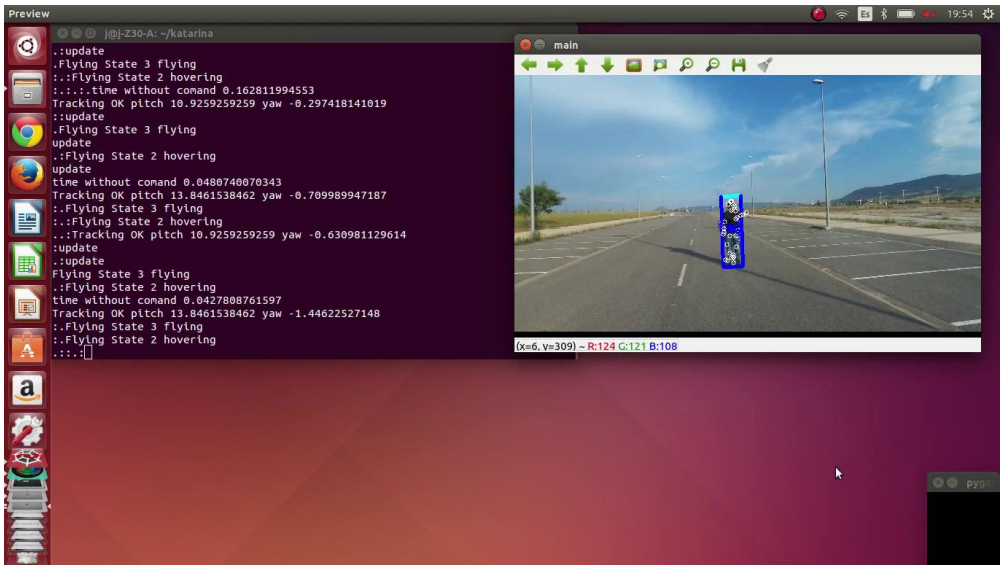

Figura 23. Tracking en funcionamiento

Una vez tenemos el nuevo centro y tamaño de la zona de interés, calculamos el movimiento necesario del drone para seguir el objeto o persona. En este caso, estamos controlando el Yaw para que el objeto a trackear se encuentre siempre centrado en la pantalla y el Pitch para que el drone se encuentre en una distancia de seguridad respecto a la persona o el objeto. Con estos dos movimientos es posible tener siempre el objeto o persona siempre centrado en la imagen y a una distancia controlada. Se ha decidido no utilizar el movimiento Roll por varias razones, la principal es para simplificar el control del cuadricóptero. Como el algoritmo de tracking utilizado puede soportar deformaciones de perspectiva del objeto a seguir es posible controlar el vehículo sin utilizar el roll. Otro motivo sería de cara al futuro trabajo, si se plantea un detector de obstáculos en la trayectoria del vehículo a través de análisis de imagen este no podría utilizarse si se realizan movimientos en horizontal como los que obtenemos con el roll ya que no podemos ver los posibles obstáculos que el vehículo tiene a los lados.

Control del Yaw objetivo:

En primer lugar, miramos si el objeto se encuentra a la izquierda o derecha de la imagen. El streaming en directo que recibimos desde el drone tiene una resolución de 360x640 píxeles, por lo que consideramos que el objeto está a la izquierda si su centro se encuentra en en x < 320 píxeles, al contrario está a la derecha de la imagen si el centro cumple x> 320. Para evitar giros innecesarios en las cercanías del centro, si el centro del objeto se encuentra en la banda de 300-340 píxeles se considera que el drone ya está centrado.

Una vez que tenemos la posición del centro calculamos el ángulo de desvío respecto al centro. Para ello nos apoyamos en el ángulo de visión de la cámara que hemos medido previamente en el laboratorio. Esta medida se realiza dejando el drone a una distancia fija de la pared y midiendo cuanto espacio de la pared vemos en el streaming que nos ofrece el drone. De esta forma hemos obtenido que el ángulo de visión de la cámara del drone es de aproximadamente 77 grados en horizontal.

Teniendo en cuenta la resolución y el ángulo de visión, podemos calcular el ángulo de desviación respecto al centro de cualquier punto de la imagen. Si tomamos como referencia un observador que se encuentra a una distancia x del centro de la imagen tenemos el siguiente par de ecuaciones:

$$\alpha = \text{atan}\,(d/x)$$

$$\beta = \text{atan}\,(\frac{320}{x})$$

Como β es la mitad del ángulo de visión que hemos medido ántes, tenemos que $\beta = 38.5\ grados$ y es un parámetro característico de la cámara del drone.

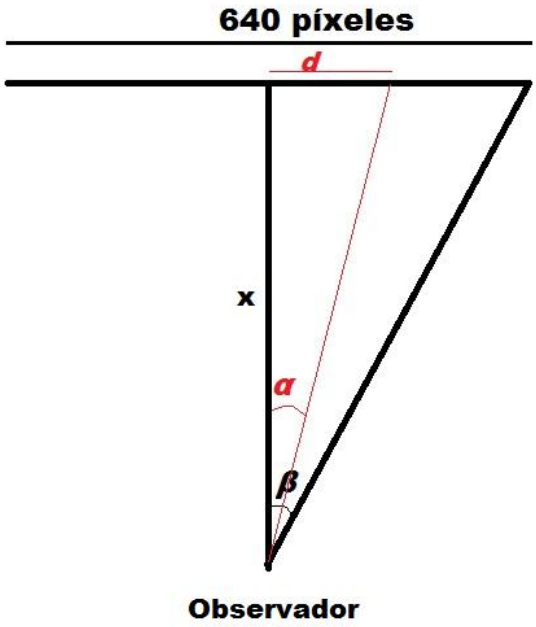

Figura 24. Cálculo de desviación del Yaw

Igualando la distancia x en las dos ecuaciones anteriores, obtenemos el ángulo de desviación respecto al centro de la imagen en función de la distancia al centro de la imagen, la resolución horizontal de la cámara y la mitad del ángulo de visión:

$$\alpha = \text{atan}\,(d * \tan\left(\frac{\beta}{320}\right))$$

Ecuación 16. Desviación respecto al Yaw

Siendo *d* igual la distancia absoluta en píxeles desde el centro de la imagen a el objetivo y β la mitad del ángulo de visión, unos 38.5 grados.

Si el objeto está a la izquierda de la imagen, nuestro Yaw objetivo será:

$$\text{GoalYaw= Yaw} - (\alpha)$$

Si se encuentra a la derecha:

$$\text{GoalYaw=Yaw} + (\alpha)$$

El control del ángulo Yaw se realiza a través de un controlador PD que se resetea cada vez que cambia el Yaw objetivo. Los valores de las contantes del controlador son Kp=100 y Kd=2. En la simulación realizada (Figura 25) vemos que la respuesta escalón tiene mucho rebose, en la

práctica prima que el drone se oriente lo más rápido posible a un rebose, por lo que se han mantenido estos parámetros de ganancia proporcional tan elevada. De esta forma conseguimos mantener centrado en la imagen el objeto a trackear.

Desconociendo los parámetros internos del control del Bebop Drone, sabemos que podemos actuar sobre la velocidad del Yaw. Siendo W la derivada del Yaw respecto al tiempo y a partir de la Ecuación 15 tenemos:

$$W = \dot{\psi}$$
$$\dot{W} = Ku$$

Introduciendo un controlador PD, tenemos que la ecuación de control es:

$$\dot{W} = K(k_p(\psi - \psi_d) - k_d W)$$

Ecuación 17 Ecuación de control de la velocidad del Yaw

Entonces tenemos que se trata de un sistema lineal y estable como podemos ver en la siguiente simulación.

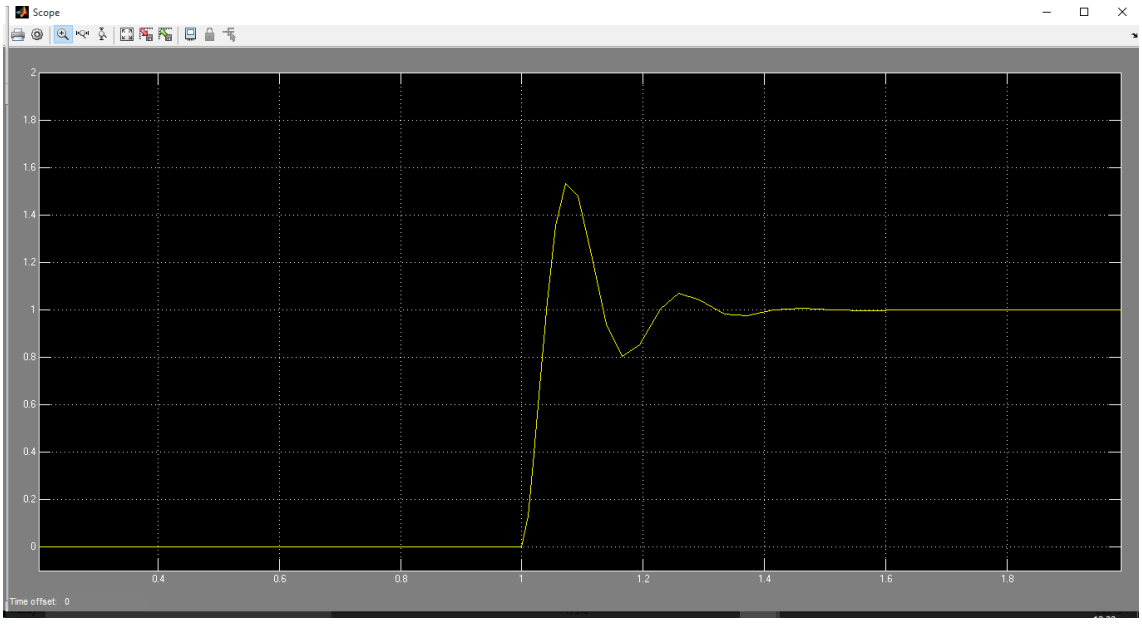

Figura 25. Simulación del control del Yaw

Control del Pitch:

En el caso del Pitch, la estrategia de control es que el drone se encuentre en un rango de distancia determinado del objeto o persona a seguir. Como en este caso se trata de una aplicación de tracking genérica pudiendo seguir objetos de distintos tamaños que no conocemos a priori, la solución se basa en mantener un rango de distancia determinado por la

distancia inicial a la que se encuentra el objeto respecto al drone cuando se inicializa el tracking. Estableciendo un rango de distancia cuando se inicializa el tracking, nos aseguramos que el drone se mantendrá siempre a una distancia de seguridad mínima dada por la distancia inicial.

Suponiendo que la cámara del drone tiene una distancia focal fija, la distancia *d* de la cámara a la imagen vendría dada por:

$$d = \frac{W * F}{Wpix}$$

Ecuación 18 Distancia a un objeto

donde *W* sería el ancho real del objeto trackeado, *F* la distancia focal y *Wpix* el ancho en píxeles en la imagen.

La idea principal es mantener la distancia *d* en un rango determinado por la distancia inicial d_1, por ejemplo, utilizaremos un rango arbitrario entre 1 y 1.3 veces la distancia inicial:

$$d_1 < d < 1.3 * d_1$$

Estos es, si *d* es proporcional a *d1*, *d= K * d1*, queremos mantener a *K* entre 1 y 1.3, si trasladamos esto a la ecuación de la distancia focal tenemos que

$$d = \frac{W_{pix}}{W_{pix1}} * d_1$$

Ecuación 19. Relación entre distancias

Por lo tanto nuestro control se debe basar en el ancho en píxeles del objeto trackeado partido su ancho en píxeles inicial para que se cumpla:

$$\boldsymbol{1 < W_{pix}/W_{pix1} < 1.3}$$

Para mantener al drone en esta distancia relativa, estamos usando un control proporcional con zona muerta de forma que el pitch sea:

$$pitch = \begin{cases} -k * (d_1 - d) & si\ d < d_1 \\ -k * (1.3 * d_1 - d) & si\ d > 1.3 * d_1 \\ 0 & Resto\ de\ los\ casos \end{cases}$$

Ecuación 20 Control del pitch en base a la distancia

Trasladándolo a la relación entre anchura en píxeles del objeto partido la anchura en píxeles del objeto inicial:

$$r = \frac{W_{pix}}{W_{pix1}}$$

$$pitch = \begin{cases} -K * (1 - r) & si\ r < 1 \\ -K * (1.3 - r) & si\ r > 1.3 \\ 0 & Resto\ de\ los\ casos \end{cases}$$

Ecuación 21 Control del pitch en base a la relación entre píxeles

Cuando el drone está volando y trackeando un objeto o una persona, pueden darse distintas circunstancias que impidan un seguimiento normal. Para ello se han diseñado una serie de controles que ordenan al drone mantenerse planeando si se dan las siguientes situaciones:

- El tracker CMT no obtiene resultados
- El tracker CMT da un resultado ilógico, por ejemplo se haya duplicado la anchura del objeto respecto al anterior

De esta forma se consigue el drone no realice ningún movimiento violento si el tracker pierde el objetivo a seguir.

Por otro lado, también para mejorar la seguridad, en caso de mal funcionamiento del programa se ha implementado un control de emergencia que envía la orden de aterrizar al drone. Para ejecutar esta orden de emergencia, tenemos que tener activa la ventana de emergencia y pulsar cualquier tecla del teclado. Como una vez se establece la zona a trackear no hay que realizar ninguna acción más en la aplicación, tenemos que activar la ventana de emergencia, llegados a este punto pulsando cualquier tecla enviaremos el comando de emergencia al drone. Cuando el tracker está en funcionamiento las ventanas de mensajes y de vídeo son meramente informativas.

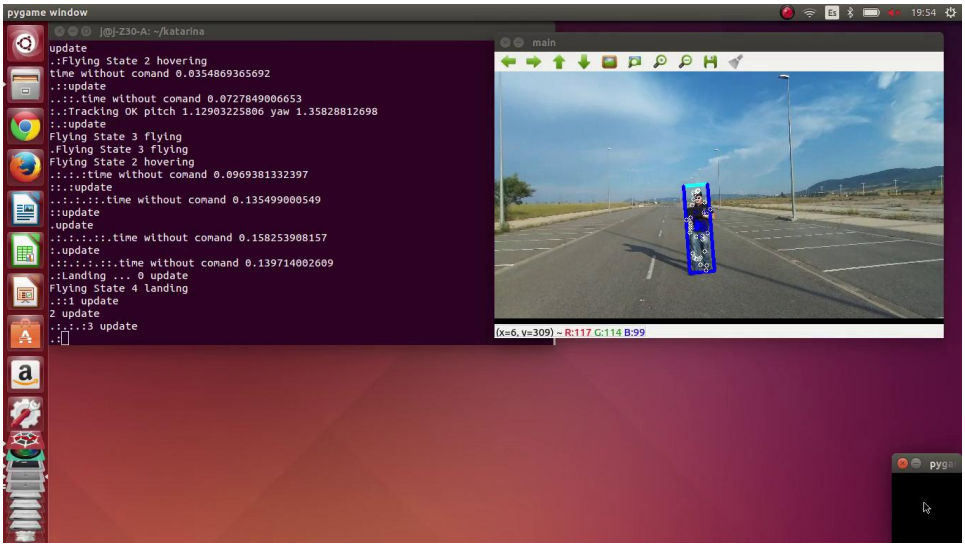

Figura 26. Ejecutando aterrizaje de emergencia, la ventana de emergencia se encuentra en la esquina inferior derecha

La aplicación desarrollada supone la integración de varias tecnologías y distintas librerías, comenzando por OpenCV para la parte de visión por computador, Python como lenguaje de programación, Numpy la librería necesaria para poder tratar imágenes junto con OpenCV, PyGame que gestiona las interrupciones de emergencia y FFMPEG como framework multimedia para decodificar el vídeo. La correcta integración y funcionamiento de todas estas tecnologías es un trabajo costoso, por lo que se le ha restado importancia a la apariencia gráfica de la aplicación centrándonos en las funcionalidades.

A forma de Log, la aplicación genera 4 archivos de texto cada vez que se ejecuta para su posterior análisis. Estos archivos incluyen datos numéricos separados por comas.

- Datos.txt:
  - Tiempo actual
  - Roll actual
  - Pitch actual
  - Yaw actual
  - Comando pitch enviado
  - Comando yaw enviado
  - Yaw objetivo
- Hover.txt: indica en el tiempo actual si el tracker ha funcionado 1 o pierde el objeto 0.
- Position.txt: posiciones actuales de x, y, z
- Velocidad.txt: velocidades V_x, V_y, V_z
- Time.txt: tiempos de procesado del tracker en cada frame

A través de estos cinco archivos se han generado offline con scripts en Matlab las gráficas que veremos en el capítulo 6 Realización de pruebas.

5 Diseño de pruebas

Al inicio del proyecto se diseñó un banco de pruebas con distintos escenarios para comprobar el funcionamiento del sistema de tracking y comprobar la respuesta que tuviese el drone ante esos escenarios. Los escenarios propuestos son:

- Tracking en interiores: una única persona sobre una iluminación artificial
- Tracking en interiores doble, que exista el cruce entre dos personas
- Tracking en exterior: se buscarán distintas condiciones de luz del día. Día soleado cara y contra el sol, día nublado, posibilidad de toma nocturna con iluminación artificial o
    - Tracking de una única persona
    - Tracking de cruce de personas
- Cambio de color de ropa de la persona a trackear durante la captura. Quitarse un abrigo por ejemplo.
- Paso de sol a sombra, en el que el sujeto a seguir pase de estar muy iluminado a una zona de sombra a y viceversa para ver si el sistema lo "pierde"
- Reflejos, en el que se intentaría "cegar" al robot con un reflejo del sol (en un cristal, luna de coche, etc...), es bastante común en exteriores.
- Oclusión o salida fuera del campo de visión. El sujeto desaparece de la escena durante algunos frames y luego vuelve a aparecer (por ejemplo pasando detrás de un árbol o saliéndose del campo de visión del drone y volviendo a entrar.

Como estos escenarios dependen únicamente del tracker y su funcionamiento, muchas de estas pruebas se han llevado a cabo dejando al drone en estado estacionario planeando a una altura constante y enviando el comando necesario para negar movimientos de roll y pitch utilizando en alguna prueba únicamente el yaw, de esta forma se han podido implementar todas las pruebas tanto en interiores como en exteriores.

6 Realización de pruebas

Dentro de cada prueba se indicará la ruta en el DVD adjunto donde se encuentran los resultados del log, el código para generar las gráficas y un video demostrativo.

6.1 Tracking en interiores con una única persona u objeto

Dado que en interiores el espacio es limitado y con numerosos obstáculos, en las pruebas realizadas en interiores se ha limitado el uso del Pitch o bien, impidiendo este movimiento o introduciendo una ganancia al controlador proporcional realmente pequeña de forma que los movimientos hacia delante o hacia atrás sean realmente cortos y suaves.

Por otro lado, en interiores debido al poco espacio disponible, en primer lugar se ha comprobado el funcionamiento del tracker trackeando un objeto y posteriormente se ha intentado trackear parte del cuerpo de una persona.

Todas las pruebas realizadas incluyen una captura en vídeo de la ejecución de la aplicación, algunas de estas se entregarán en el DVD adjunto a la memoria del proyecto.

Tracking de un objeto impidiendo el movimiento del drone, detector BRISK:

D:\pruebas\indoor\prueba1

En esta primera prueba, únicamente queremos comprobar el funcionamiento del tracker y simular los comandos de Yaw y Pitch que mandaríamos al drone. También queremos comprobar el número de veces que el tracker encuentra el objeto.

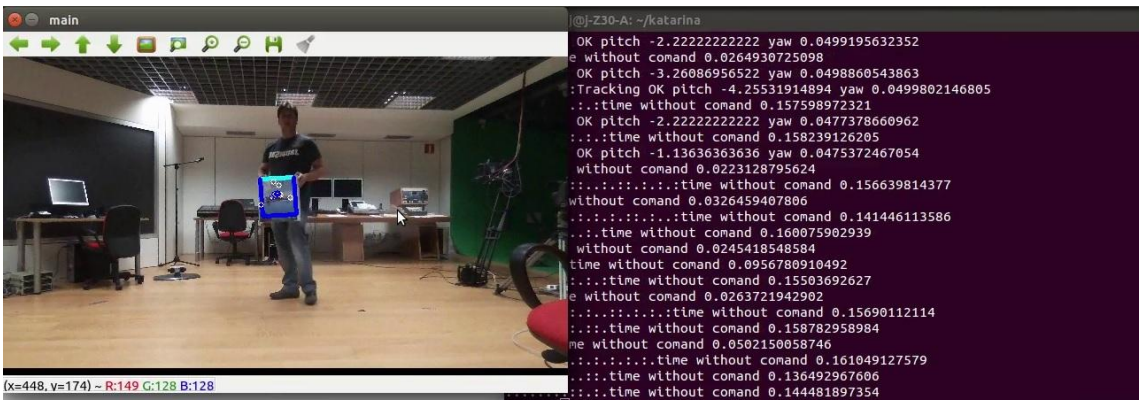

Figura 27. Trackeando un objeto en interiores

A continuación y al igual que en el resto de pruebas incluiremos gráficas con los parámetros leídos y enviados al drone. En este caso como el movimiento del drone está negado, las gráficas incluyen los valores de las señales de control del Pitch y Yaw pero si haber sido enviados al drone.

En primer lugar tenemos una gráfica con los tiempos en los está funcionando con valor=1, el resto del tiempo o bien el tracker todavía no está en funcionamiento o ha perdido el objeto. Lamentablemente no existe ninguna forma de saber si la respuesta del tracker es un falso positivo, si detecta realmente el objetivo. Únicamente comprobamos si nos da un resultado o no.

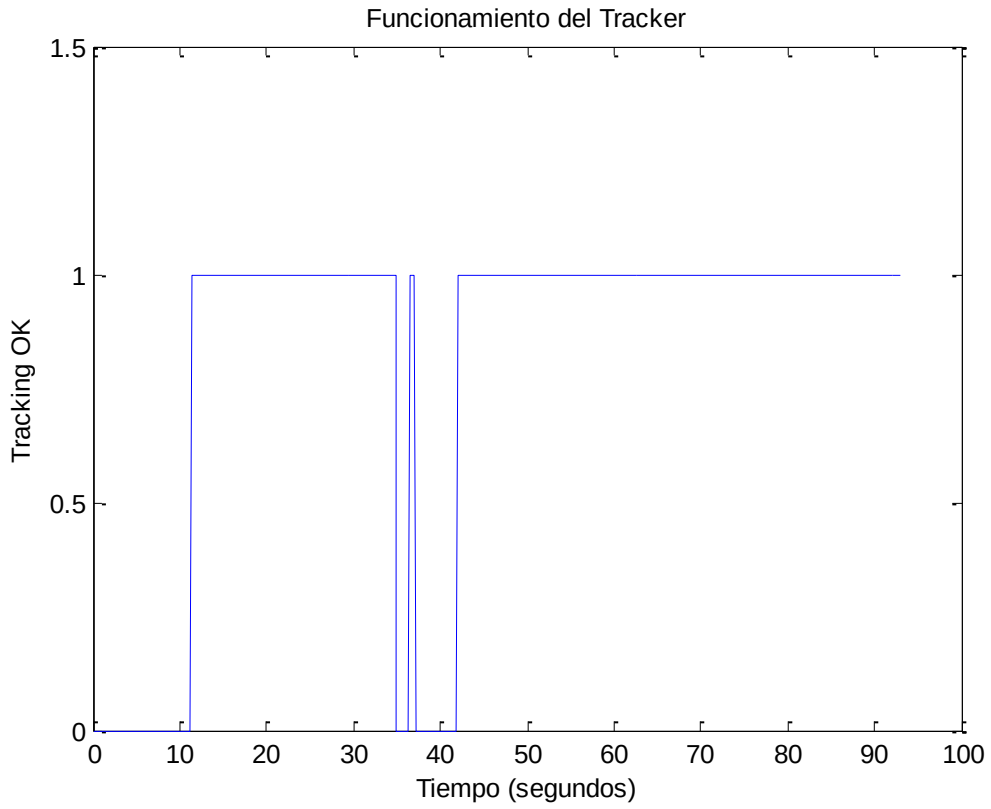

Figura 28. Funcionamiento del Tracker

En la siguiente gráfica tenemos el Yaw actual medido, el enviado al drone como señal de control con valores comprendidos en el rango [-100,100] y el Yaw objetivo, que es el Yaw al que debe estar el drone para estar orientado hacia el objetivo. Los valores de Yaw son en radianes y se miden en radianes desde del polo norte magnético.

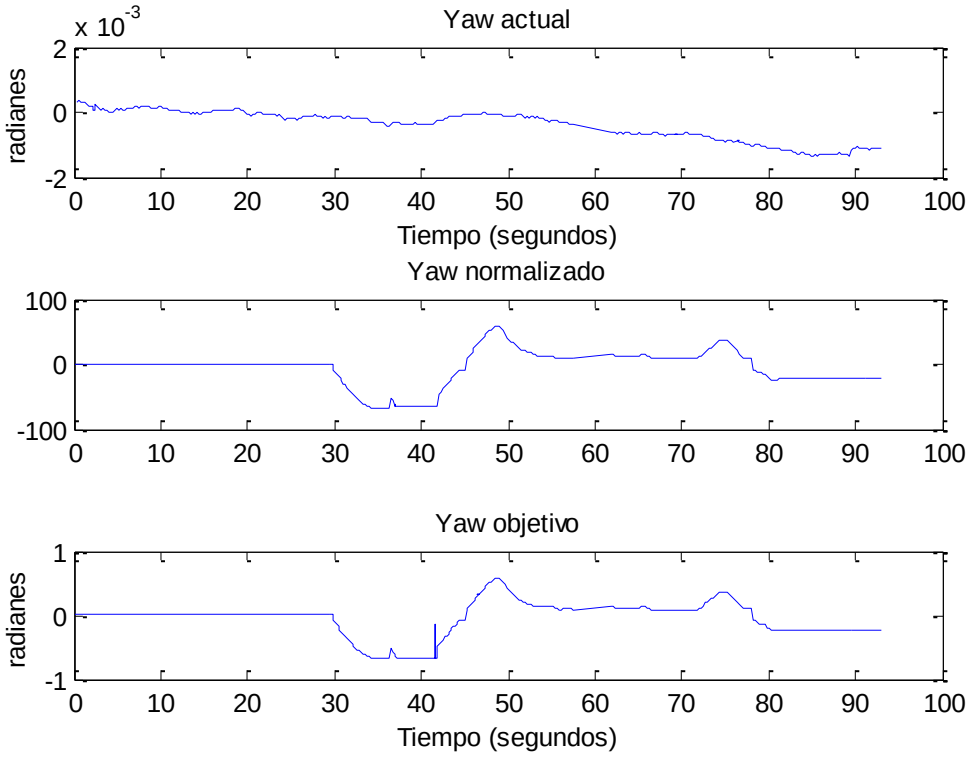

Figura 29. Yaw simulado

Observando las gráficas sobre el Yaw simulado vemos que en las cercanías del segundo 42 existe una anomalía en el Yaw objetivo, esto se debe a que el tracker en ese tiempo ha dejado de funcionar y ha dado unos resultados anómalos al recuperar el objeto, por ese motivo durante un instante el Yaw objetivo que se ha calculado difiere de la realidad.

También tenemos el control del Pitch, en este caso únicamente estamos simulando el valor del pitch que enviaríamos al drone como señal de control:

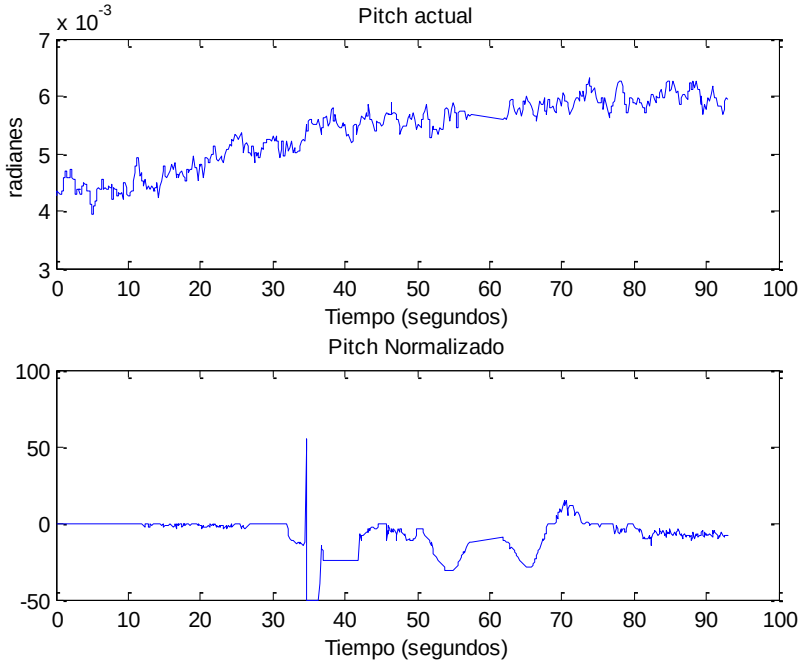

Figura 30. Pitch simulado

Por último, para hacernos una idea del ruido existente en los sensores del drone, también hemos medido el Roll, recordemos que en este caso el drone está quieto.

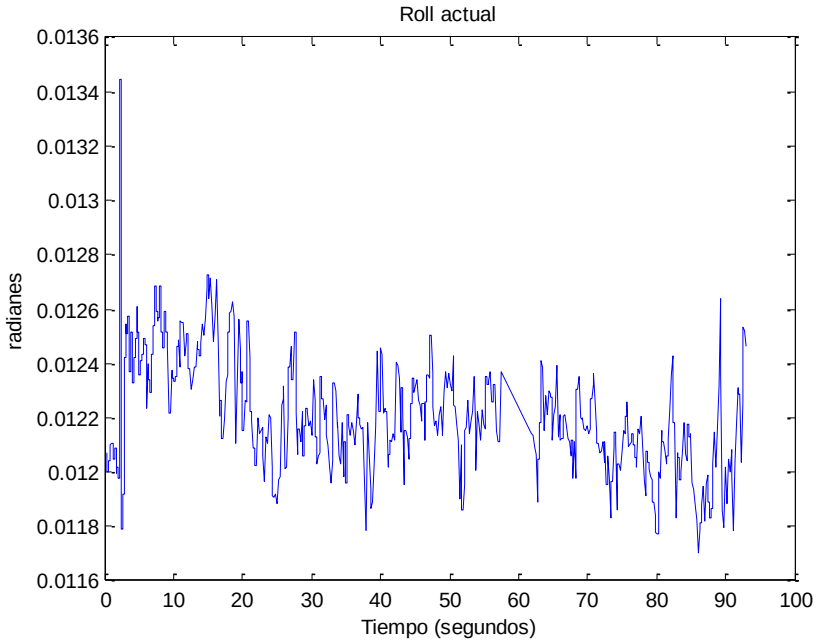

Figura 31. Roll con el drone en reposo

Analizando el ruido del roll vemos que su media está en 0.0122 y tiene una varianza de 4.7946e-08. El valor medio del roll tiene mucha diferencia a su valor real que debería ser cero ya que el drone se encuentra quieto sobre una superficie lisa y plana, una mesa.

En las inmediaciones del segundo 60 se han perdido valores, tal vez por un fallo en la comunicación entre el PC y el drone o porque la CPU del drone se encontraba saturada y dejó de enviar información sobre su telemetría. Este fallo se produce durante unos 4 segundos y el número de datos aproximado que se han perdido es de 40. El drone no envía su telemetría a un ritmo constante y el envío de estos datos al igual que el streaming se detiene en caso que su CPU esté saturada. Hay que recordar que el drone estabiliza la imagen por software, esto es un trabajo de un alto coste computacional y cuando hay variaciones muy bruscas en la imagen se puede dar este tipo de problemas.

Tracking de una persona impidiendo el movimiento del drone, detector BRISK:

D:\pruebas\indoor\prueba2

En esta prueba hemos realizado los mismos pasos que en la anterior pero siguiendo el cuerpo de una persona.

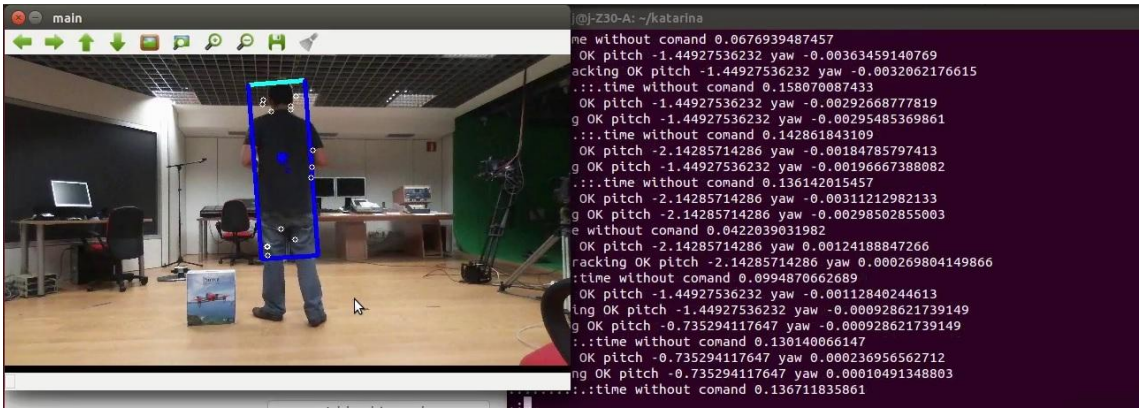

Figura 32. Tracking de una persona sin movimiento del drone

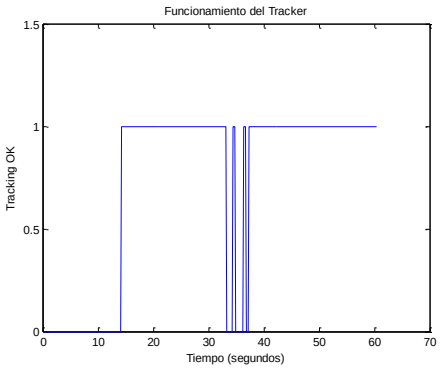

Figura 33. Funcionamiento del tracker interior, una persona

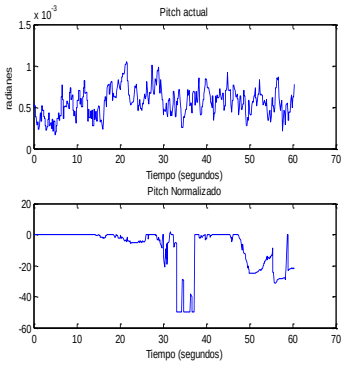

Figura 34. Pich, interior, una persona

Para comprobar posibles retardos, comprobamos el tiempo que tarda el tracker en procesar cada frame:

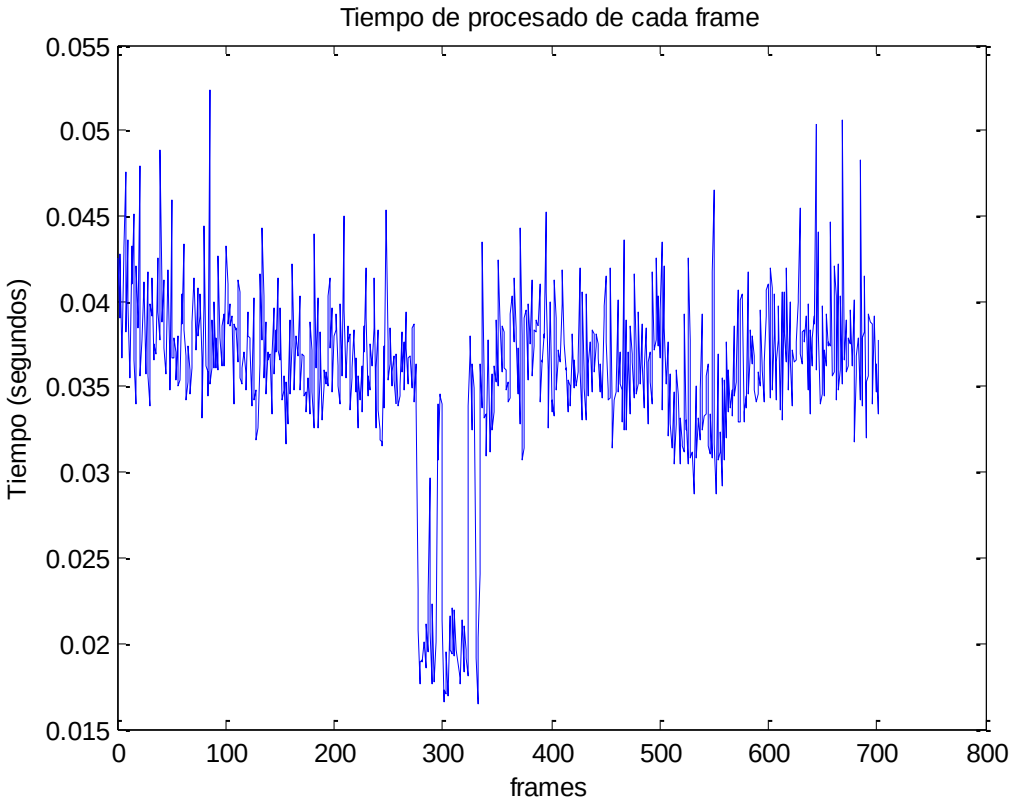

Figura 35. Tiempo de procesado del tracker para cada frame

Como vemos con el detector BRISK el tracker tarda como máximo alrededor de 0.055 segundos en procesar cada frame. El valor medio de tiempo de procesado de un frame es de

0.0363 segundos con una varianza de 2.9e-5. En la zona del frame 300 podemos ver que en varios frames el tiempo de procesado baja a la mitad, esos frames coinciden con el periodo en el que el tracker no encuentra el objeto en la imagen. Se da el caso de que el tracker falla y no encuentra los puntos característicos en la imagen, por lo que no tiene que hacer más operaciones y el tiempo de procesado del frame baja.

Tracking de un objeto impidiendo el movimiento del drone, detector FAST:

D:\pruebas\indoor\prueba3

En esta prueba hemos modificado el detector de características usado por el detector FAST. Este detector nos devuelve una cantidad muy superior de puntos característicos, esto debería ser una ventaja, sin embargo aumenta el tiempo de procesado de cada frame ya que el tracker tiene que comparar más puntos en las imágenes. En escenarios con muchos objetos como es el lugar donde se ha realizado la prueba es fácil que los puntos característicos se dispersen del objeto.

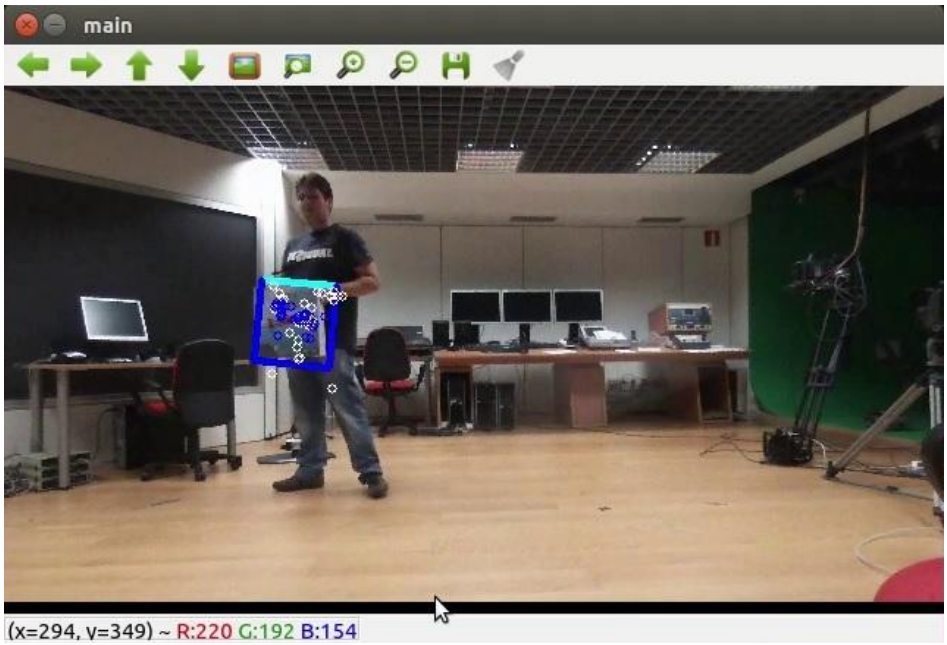

Figura 36. Tracking de un objeto indoor con detector FAST

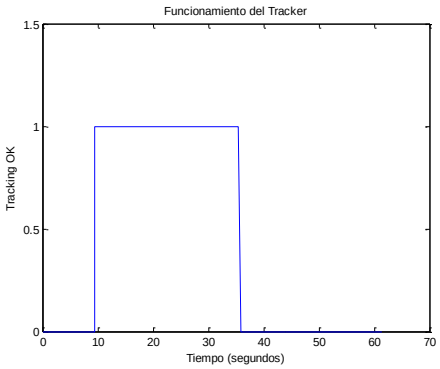

Figura 37. Tracker funcionando

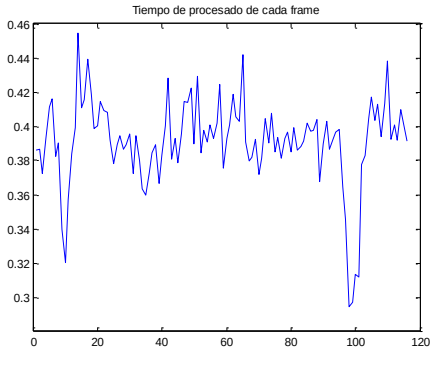

Figura 38. Tiempo de procesamiento de cada frame

Como vemos el tiempo de procesado con el detector de características FAST puede llegar hasta los 0,46 segundos, que es aproximadamente 10 veces mayor que el detector BRISK.

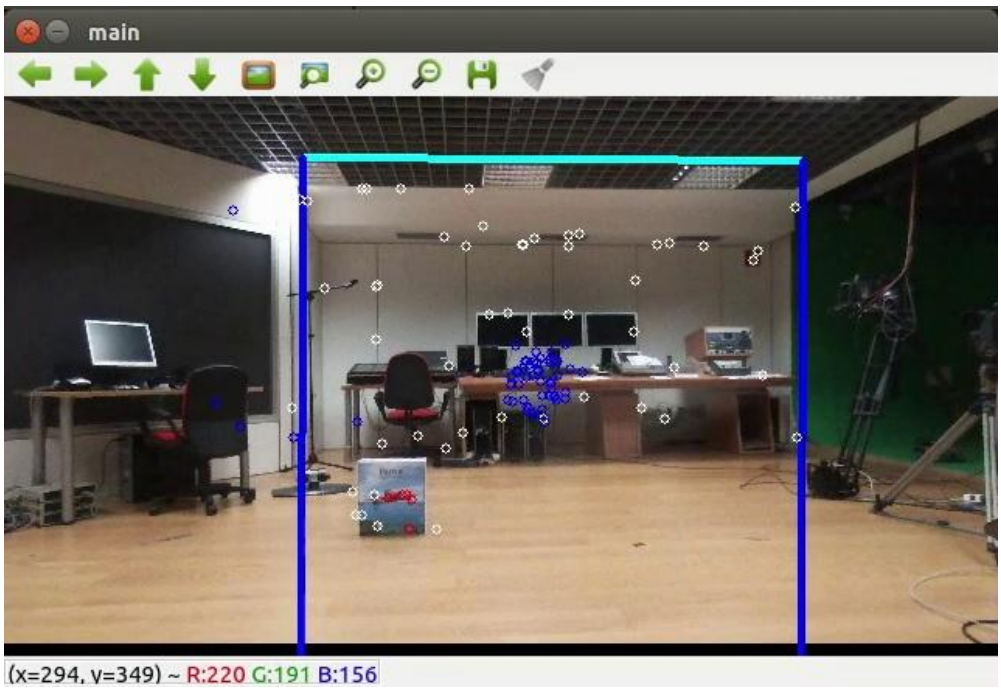

Figura 39. Detector Fast pierde el objeto

Por otro lado, vemos que una vez el tracker ha perdido el objeto no ha sido capaz de encontrarlo. Esto puede ser debido a la cantidad de objetos existentes en el espacio donde se ha realizado la prueba y a la escasa iluminación existente en ese espacio.

Tracking de una persona en interiores impidiendo el movimiento del drone, detector FAST:

D:\pruebas\indoor\prueba4

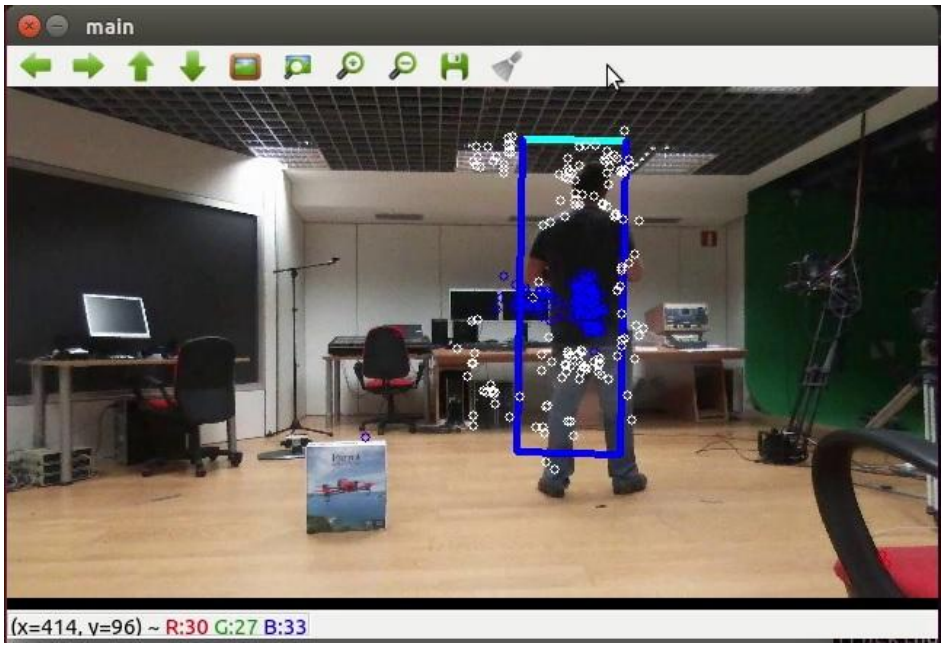

Figura 40. Tracking de una persona en interiores con detector FAST

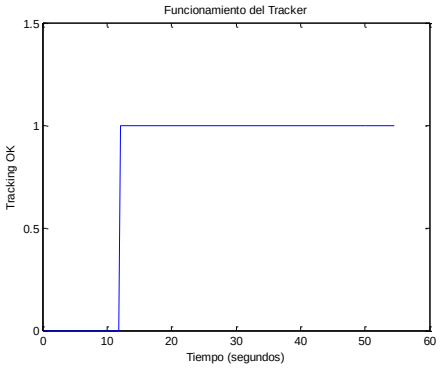

Figura 41. Funcionamiento del tracker

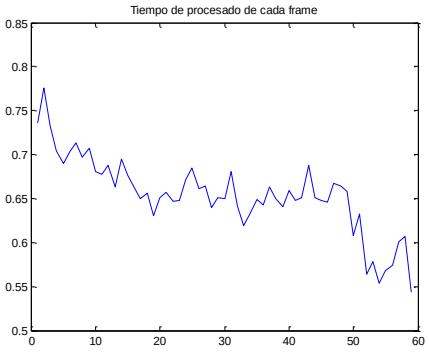

Figura 42. Tiempo de procesado

En el caso del tracking de una persona con el detector FAST en interiores vemos que aparentemente ha funcionado mejor, sin embargo la cantidad de puntos característicos se han perdido en el fondo de la sala como podemos ver en la siguiente figura. Por otro lado, vemos que el elemento a trackear, al ser más complejo tiene mayor número de puntos característicos, lo cual aumenta el tiempo de procesado, superando en este caso los 0.75 segundos por frame al comenzar la ejecución y con un valor medio de tiempo de procesado por frame de 0.65 segundos, con una varianza de 0.002.

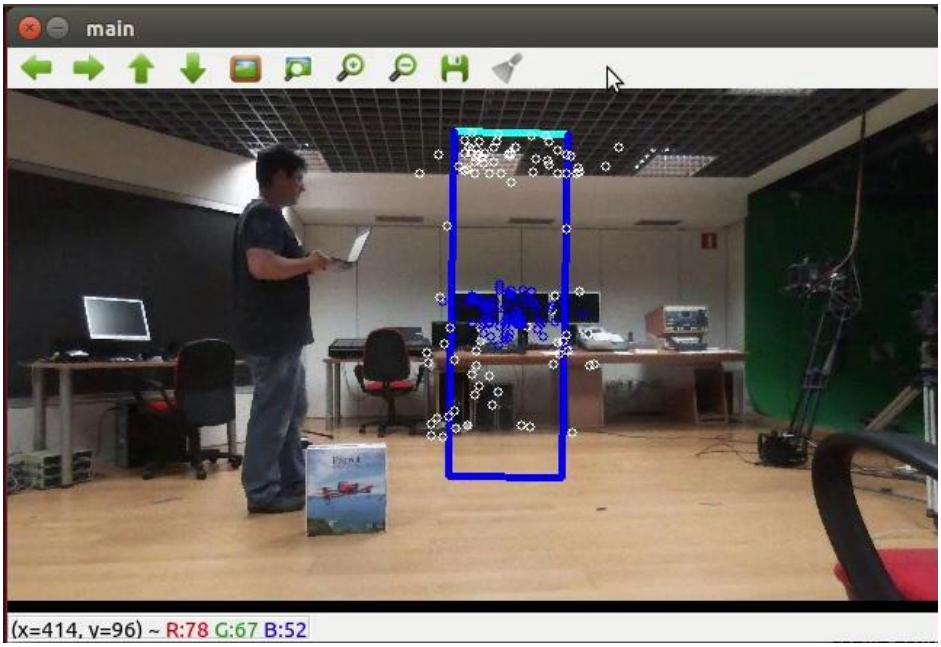

Figura 43. Tracker con detector FAST perdido en interiores

Tracking de una persona en interiores impidiendo el movimiento del drone, iluminación y fondo controlados, detector BRISK:

D:\pruebas\indoor\prueba5

Ya que los detectores de características se "pierden" en escenarios complicados, hemos introducido una prueba para el tracker dentro de un escenario controlado. En este caso se trata de un croma con fondo verde y una iluminación adecuada. En este caso el tracker funciona correctamente, pudiendo entrar y salir del escenario sin que se confunda con otros elementos.

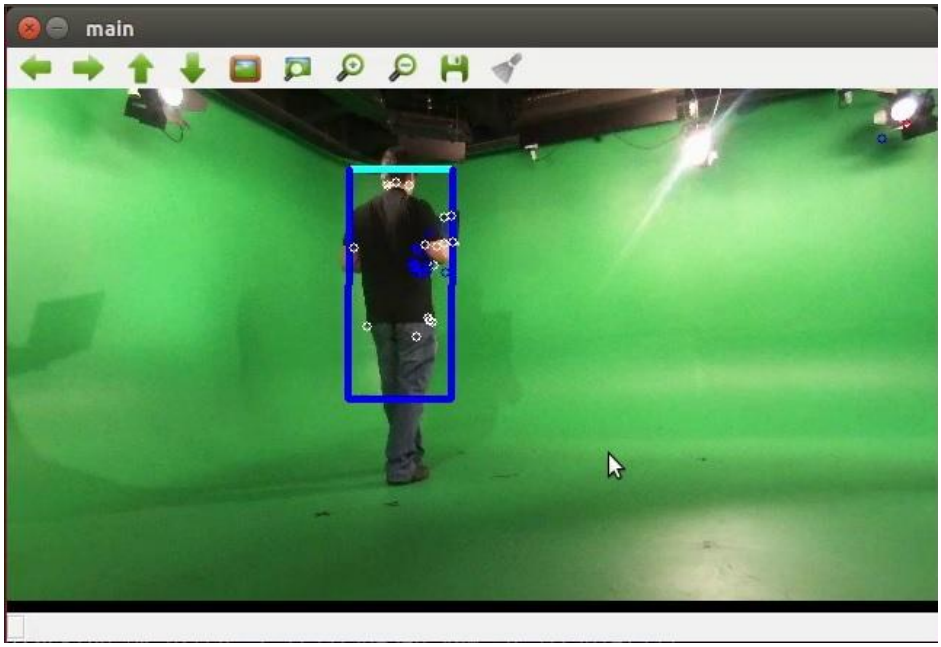

Figura 44. Tracking indoor con fondo e iluminación controlado

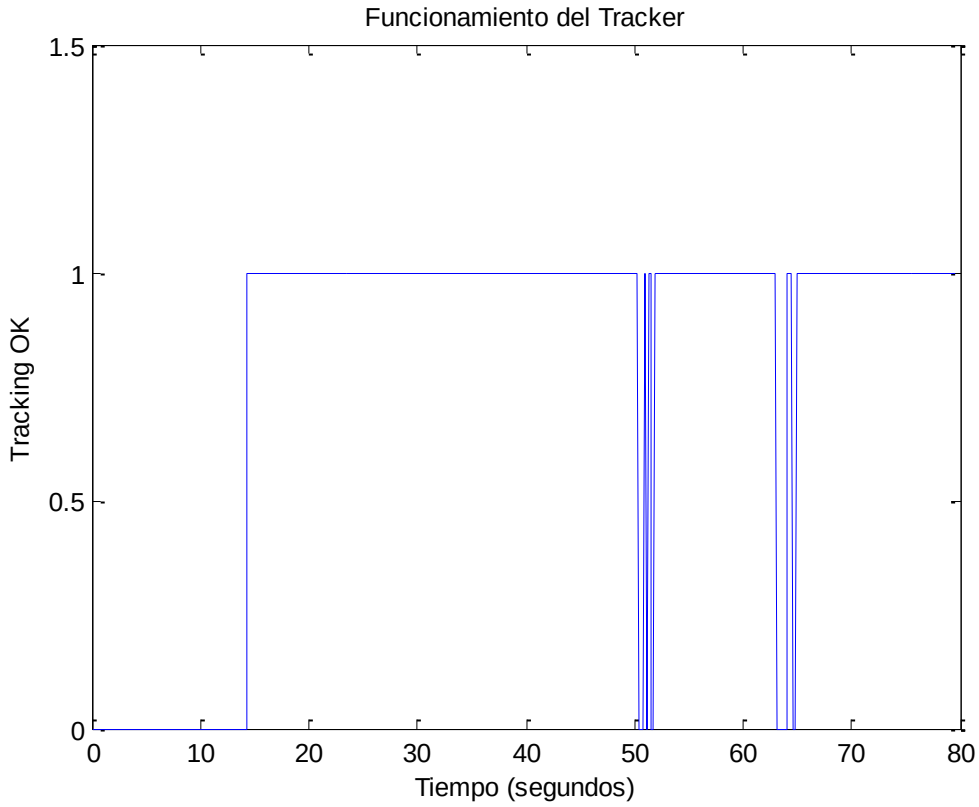

Figura 45. Tiempo de funcionamiento del tacker, detector BRISK en condiciones controladas

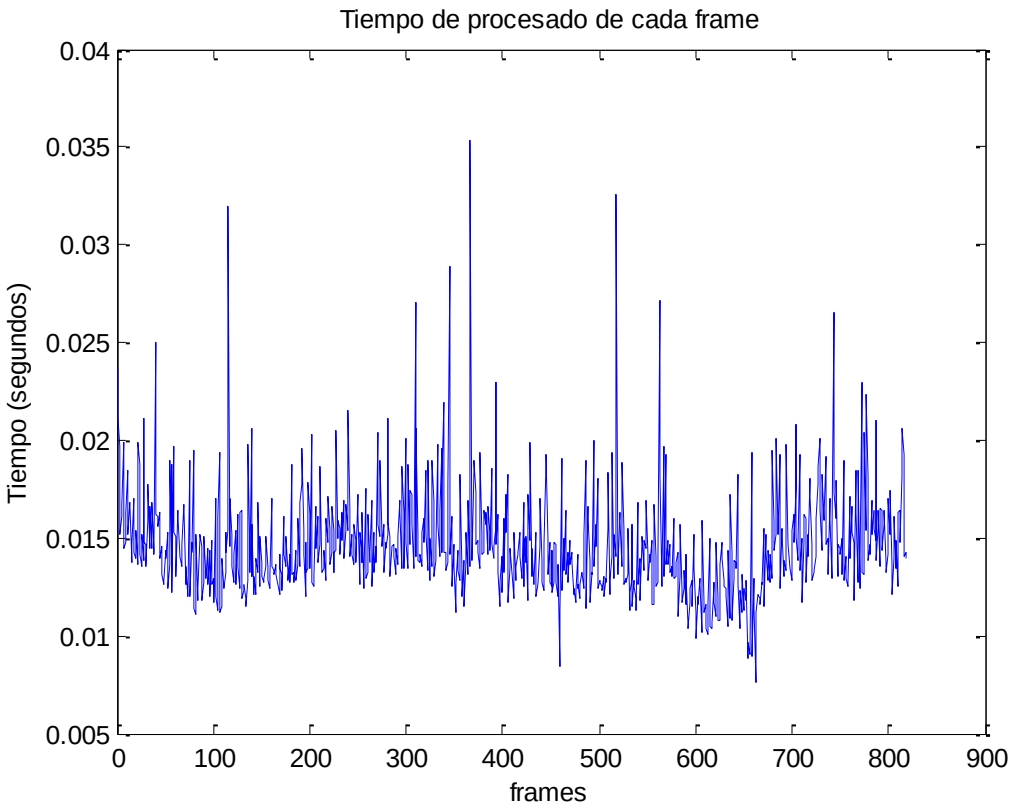

Figura 46. Tiempo de procesamiento de cada frame, detector BRISK en condiciones controladas

Podemos comprobar que el tiempo de procesado de cada frame baja considerablemente a cerca de la mitad respecto a la misma prueba en un entorno interior real. En este caso tenemos un valor medio de tiempo de procesado de 0.0142, con una varianza de 7.7348e-06. También vemos que el algoritmo de tracking funciona mucho mejor al no existir muchos objetos en la escena.

Tracking de una persona en interiores impidiendo el movimiento del drone, iluminación y fondo controlados, detector FAST:

D:\pruebas\indoor\prueba6

Figura 47. Tracking indoor controlado detector FAST

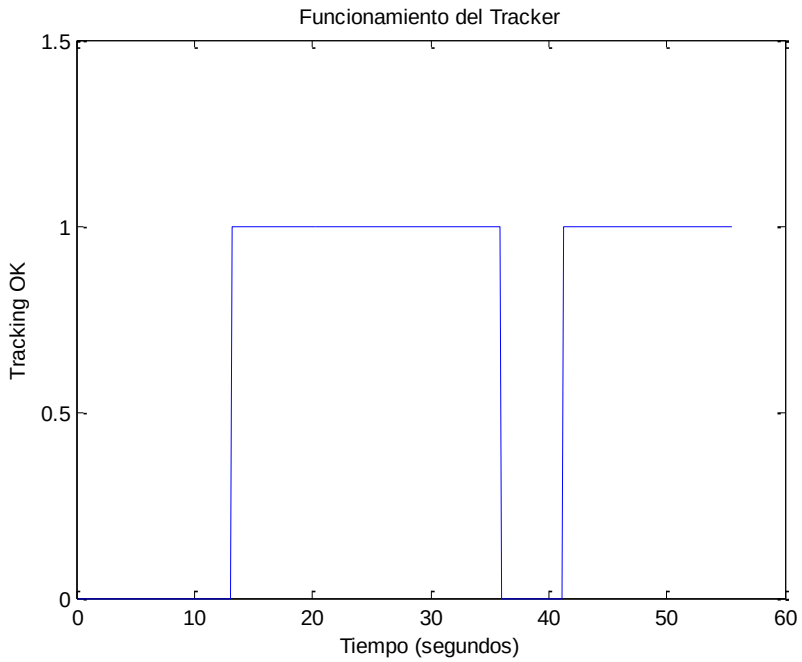

Figura 48. Funcionamiento del tracker

En este caso el tracker funciona correctamente, pudiendo salir de la escena y siendo detectado correctamente una vez se vuelve a entrar.

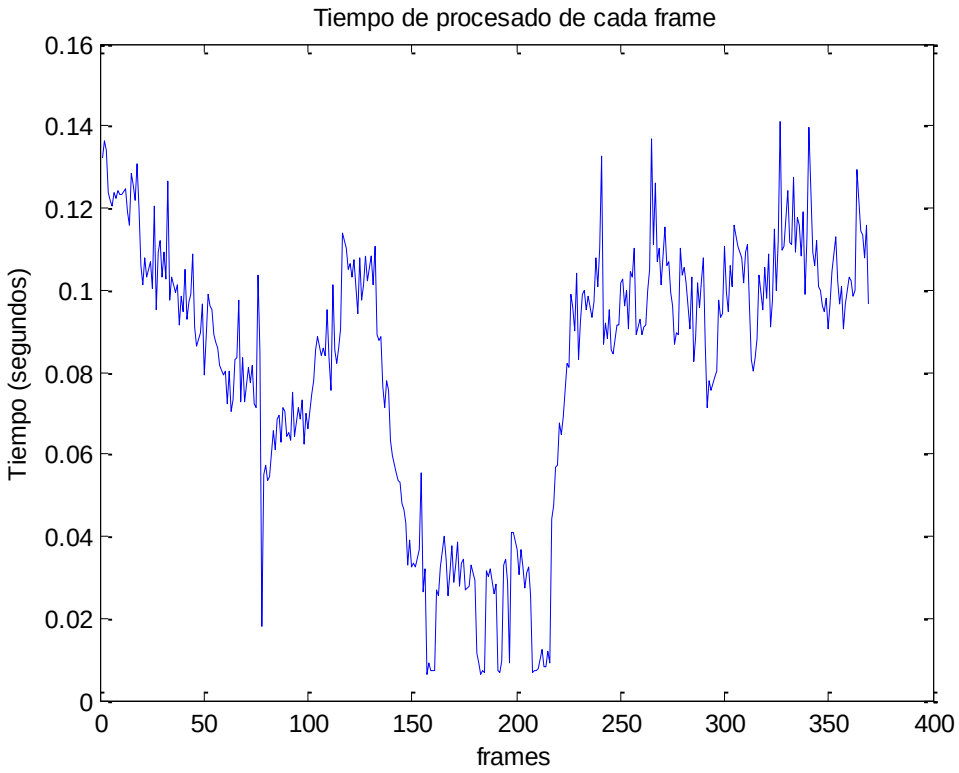

Figura 49. Tiempo de procesado por cada frame

El tiempo de procesado de cada frame también disminuye al ser un escenario simple con pocos objetos. El valor medio del tiempo de procesado es de 0.0912 segundos con una varianza de 0.0011.

Como hemos comprobado en un entorno controlado y con una correcta iluminación el algoritmo de tracking funciona mejor, siguiendo correctamente la persona y realizando los cálculos necesarios en menor tiempo.

Una correcta iluminación influirá en gran medida al funcionamiento del tracker, ya que los puntos característicos son puntos donde el gradiente de intensidad es máximo, al estar el objeto a trackear correctamente iluminado, sus bordes crearán sombras donde obtendremos estos puntos.

Tracking de un objeto en interiores permitiendo movimiento Yaw

D:\pruebas\indoor\prueba7

Una vez comprobado el funcionamiento del tracker CMT en distintas condiciones en interiores, se ha realizado una nueva prueba trackeando un objeto con el detector BRISK. En este caso se ha permitido el movimiento Yaw del drone para comprobar que el objeto a seguir permanece centrado en el vídeo capturado por el drone. Hemos utilizado el detector BRISK porque es el que mejor se ha comportado siguiendo un objeto en bajas condiciones de iluminación.

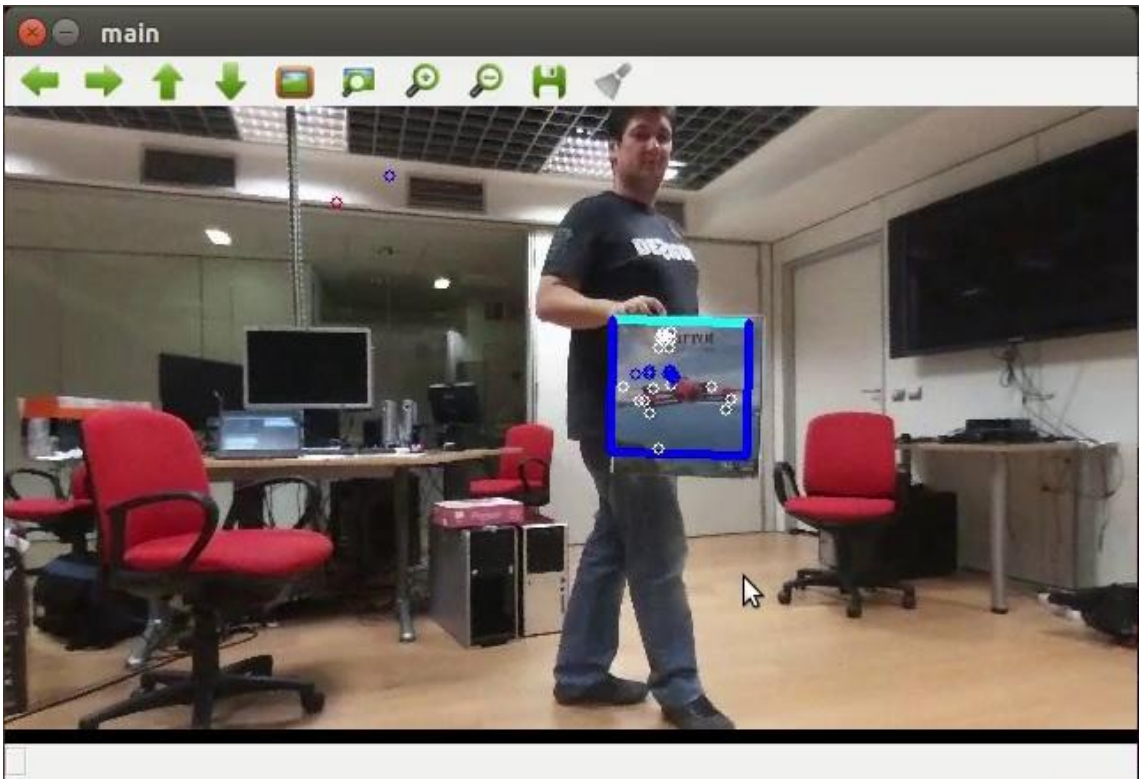

Figura 50. Tracking de un objeto permitiendo el movimiento Yaw

En esta prueba el objeto se ha movido unos 180 grados por delante de la posición de partida del drone. Para comprobar la respuesta del drone ante la pérdida del objeto se ha "ocultado" éste girándolo para que el tracker no sea capaz de detectarlo, esto lo podemos comprobar en las cercanías del segundo 50 viendo como el tracker deja de funcionar.

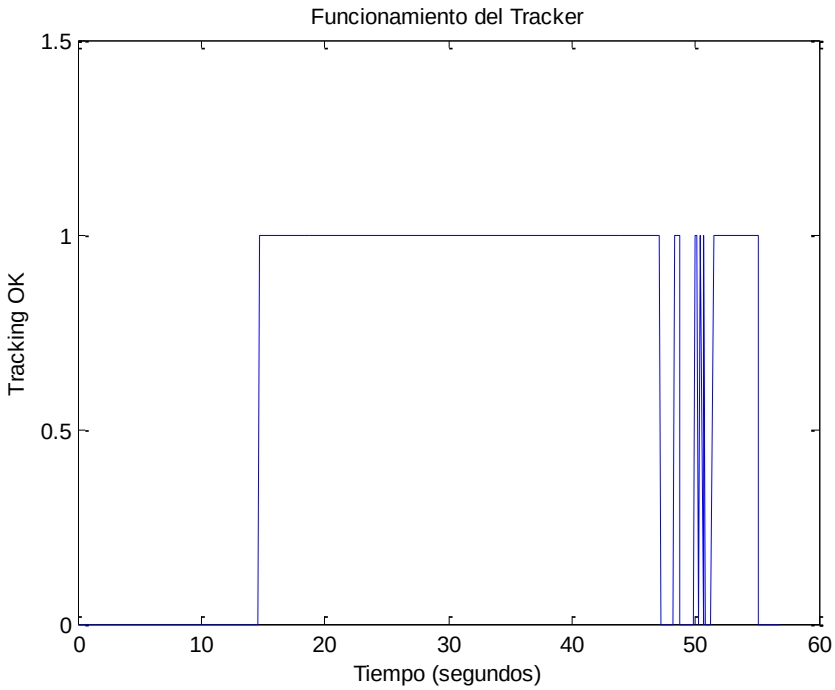

Figura 51. Tiempo de funcionamiento del tracker

Figura 52. Objeto desaparece

El ángulo Yaw que mide el Bebop Drone es el ángulo desde el polo norte magnético, por ese motivo en la gráfica de la Figura 53 vemos que la medida del Yaw actual comienza cerca de -2 radianes. Mientras el objeto se mueve el Yaw del robot va cambiando adecuadamente para seguir al objeto.

Pasado el segundo 50 podemos comprobar cómo se le ha enviado un pico al comando de Yaw y cómo también se ha calculado un Yaw objetivo muy diferente al anterior. Este es el momento donde se ha ocultado el objeto, al hacerlo el tracker en un instante ha detectado que el objeto se encuentra en una posición muy diferente a la previa. En este caso se trata de un falso positivo. Pero como los comandos de movimiento que se envían al drone tienen una frecuencia de 40Hz, este falso positivo no ha tenido repercusiones en el correcto seguimiento del objeto.

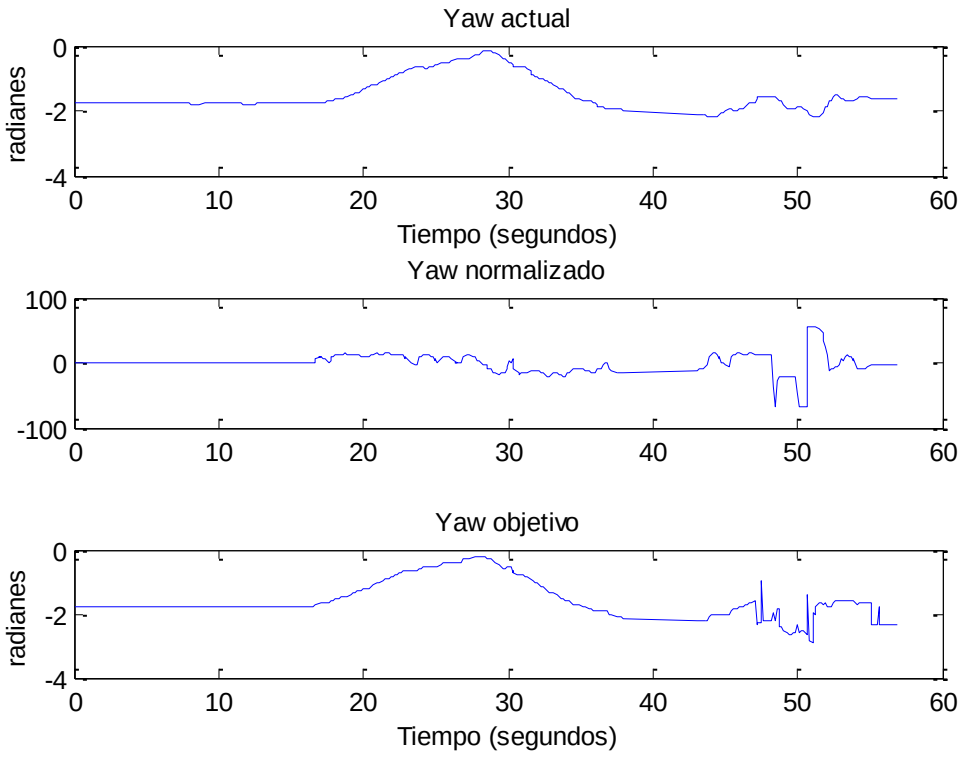

Figura 53. Cambio de Yaw en el tracking

Por último vamos a ver si la aplicación detecta la cercanía al objeto viendo el comando Pitch que enviaría al drone en esta prueba:

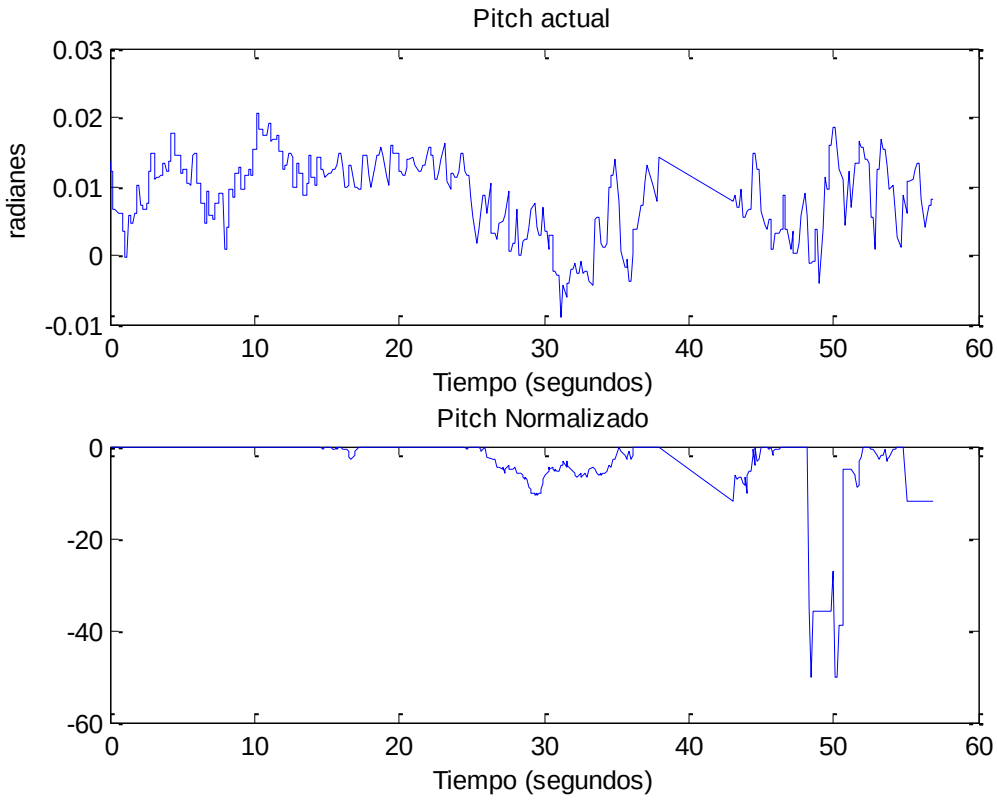

Figura 54. Pitch actual del drone y comando simulado

En la figura anterior comprobamos que el Pitch normalizado es correcto salvo en las cercanía del momento en que el objeto desaparece. Al cambiar la perspectiva del objeto, la anchura del objeto detectado cambia considerablemente, por lo que la aplicación estima que el objeto está demasiado cerca del drone y le envía comandos para que se aleje rápidamente.

Aunque no se aprecie en la Figura 53, ni en la Figura 51, vemos en sobre el segundo 40 de la prueba hemos sufrido una pérdida de datos de la telemetría. Como ya hemos mencionado, en ocasiones el drone deja de enviar tanto su telemetría como el streaming. Este fallo se produce durante unos 5 segundos.

6.2 Tracking en interiores cambio de color de ropa

D:\pruebas\indoor\prueba8

En esta prueba vamos a comprobar si al tracker CMT le afectan los cambios de color del objeto a seguir, para ello dentro del escenario controlado en interiores el sujeto al que el tracker va a

seguir se va a quitar una chaqueta en mitad de la ejecución del tracker. Esto lo haremos para los dos detectores de características: BRISK y FAST.

Reacción ante eliminación de una prenda, detector BRISK

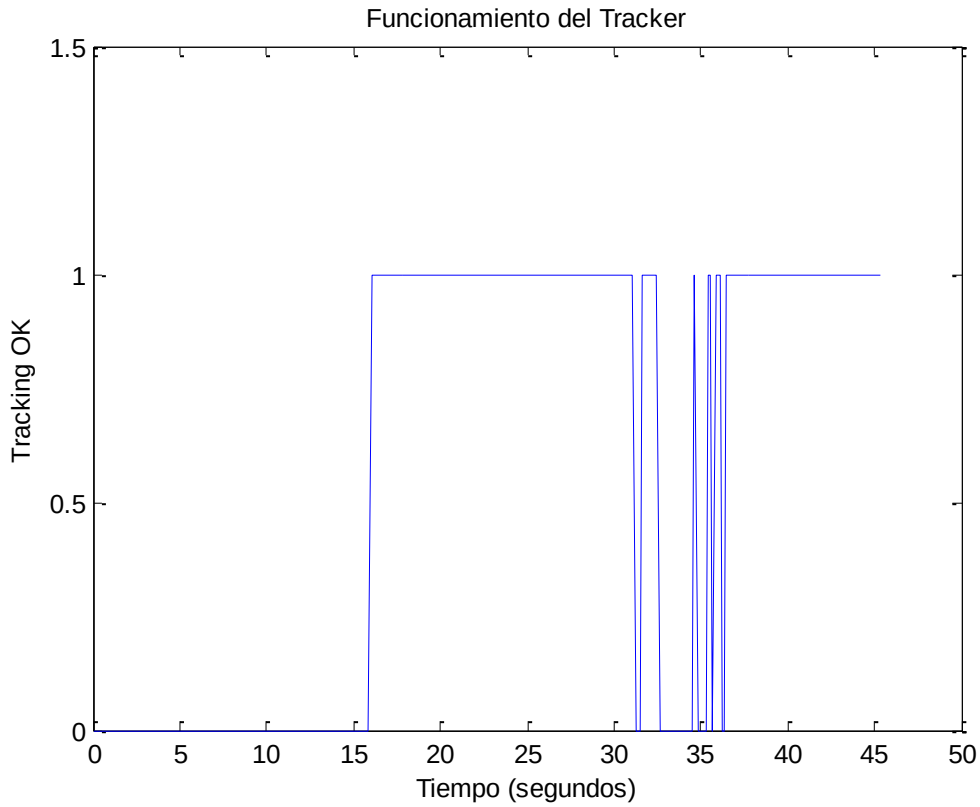

Figura 55. Funcionamiento del detector BRISK ante cambio de color de ropa

Aunque en un primer momento el tracker funciona correctamente sufre algún fallo nada más producirse la eliminación de una prenda de la persona a seguir. Posteriormente, continúa siguiendo bien a la persona.

Figura 56. Tracking inicial

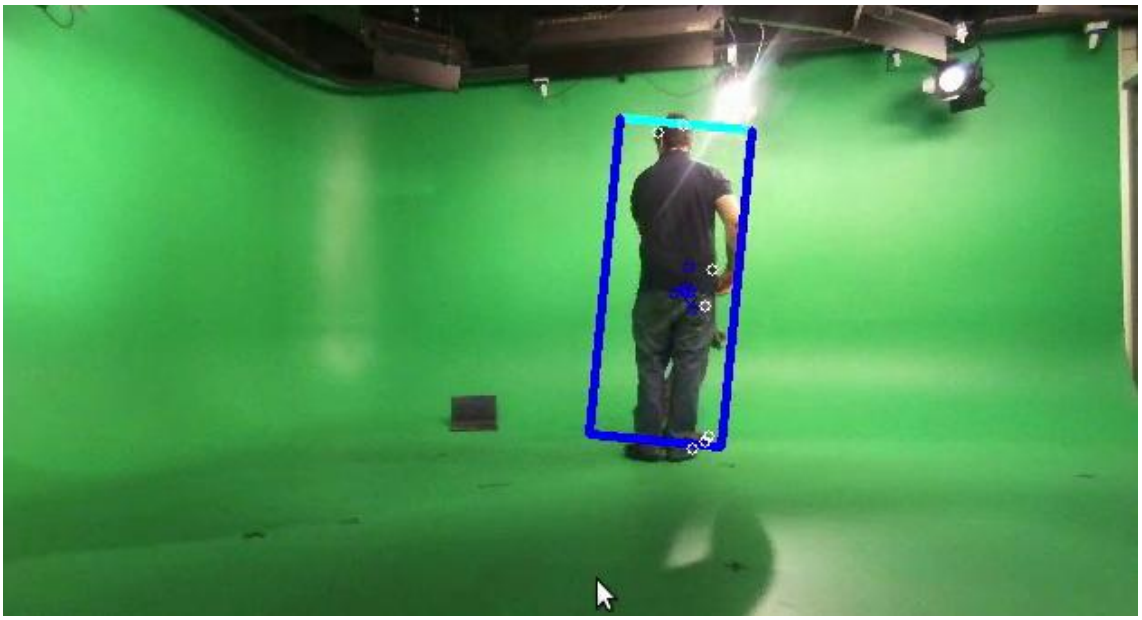

Figura 57. Cambio de prenda

Figura 58. Falso positivo del tracker

A pesar de que el tracker nos da algún error poco después de la eliminación de la prenda, el tracker responde correctamente volviendo a seguir a la persona.

Reacción ante eliminación de una prenda, detector FAST

D:\pruebas\indoor\prueba9

Aplicando el mismo procedimiento con el detector FAST, tenemos:

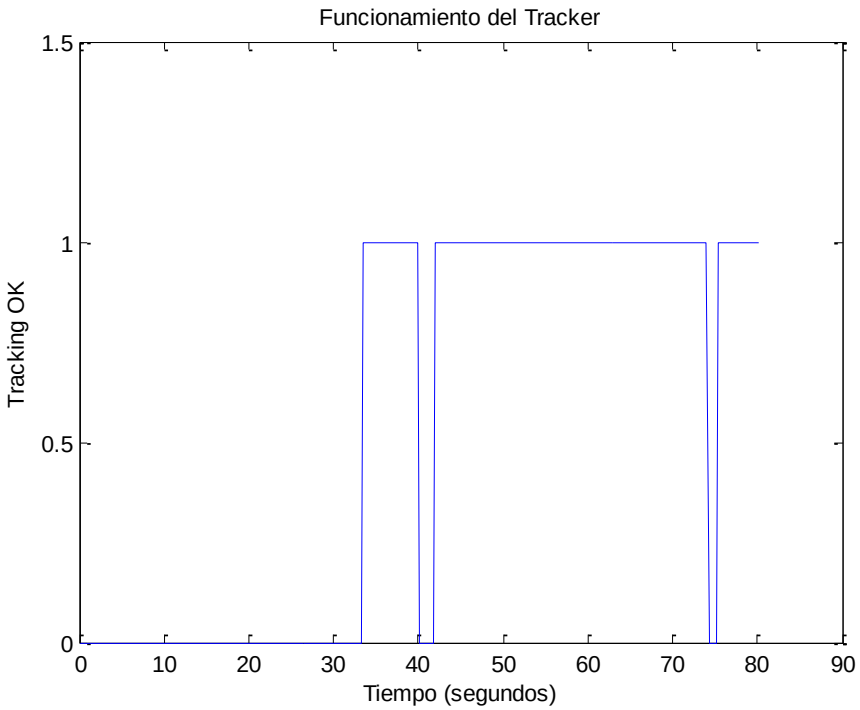

Figura 59. Funcionamiento del tracker

Podemos ver a través de la gráfica anterior que el detector FAST es menos errático en esta situación, sin embargo también tenemos algún fallo de funcionamiento al perder al sujeto.

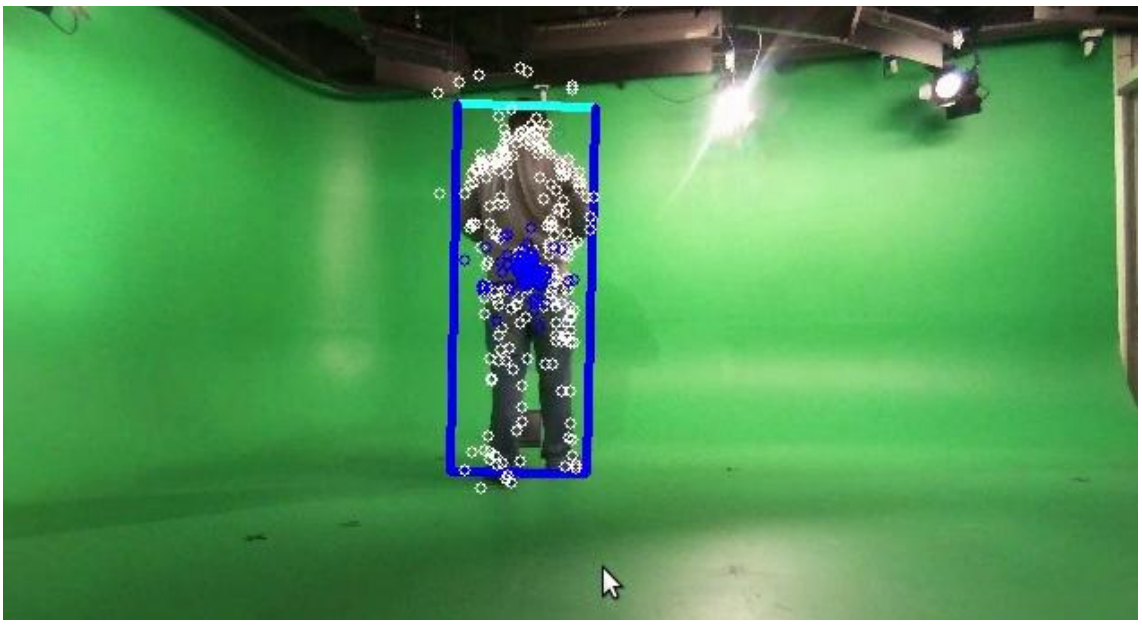

Figura 60. Instante inicial del tracking

Figura 61. Cambio de prenda

Figura 62. Fallo del tracker

A pesar de que realizando la prueba con el detector FAST también hemos obtenido fallos en el tracking, este es más estable que el anterior, ya que rápidamente vuelve a seguir a la persona, podemos considerar que el tracker es inmune al cambio de ropa de la persona.

La Figura 60 muestra el estado inicial del tracking, una vez se elimina una prenda, Figura 61 el tracker funciona correctamente, pero un instante posterior tenemos una pérdida del

streaming con lo que el tracker se pierde durante un instante (Figura 62) aunque vuelve a recuperar el objetivo al finalizar la ejecución de la aplicación.

Como hemos comprobado con estas pruebas, el tracker es inmune al cambio de color de una prenda. Este hecho se debe a que los puntos característicos que sigue el tracker coinciden con el contorno del objeto o persona a seguir y no tienen relación alguna con el color.

6.3 Tracking en exteriores en condiciones no controladas

Una vez realizadas todas las pruebas en interiores limitando los movimientos del drone, llega la hora de probar el desarrollo en exteriores donde las condiciones no estén controladas como iluminación y viento.

En este caso, también vamos a comprobar el funcionamiento del tracker con los dos detectores de características BRISK y FAST. Para las primeras pruebas únicamente seguimos una persona.

Las pruebas en exteriores se han realizado en una zona industrial en desuso para evitar la afluencia de gente y que en caso de accidente los daños sean menores.

Dentro de cada prueba se incluye la ruta al CD adjunto a la memoria técnica donde se encuentran al igual que en las pruebas anteriores los datos de log, el código en Matlab para generar las gráficas y en este caso un vídeo de cada prueba.

Tracking de una única persona en exteriores con detector BRISK

D:\pruebas\outdoor\primera prueba

En esta primera prueba en exteriores se sigue el cuerpo de una persona con el detector de características BRISK. Las condiciones ambientales son un día soleado con una velocidad de viento de entre 10 y 15 km/h.

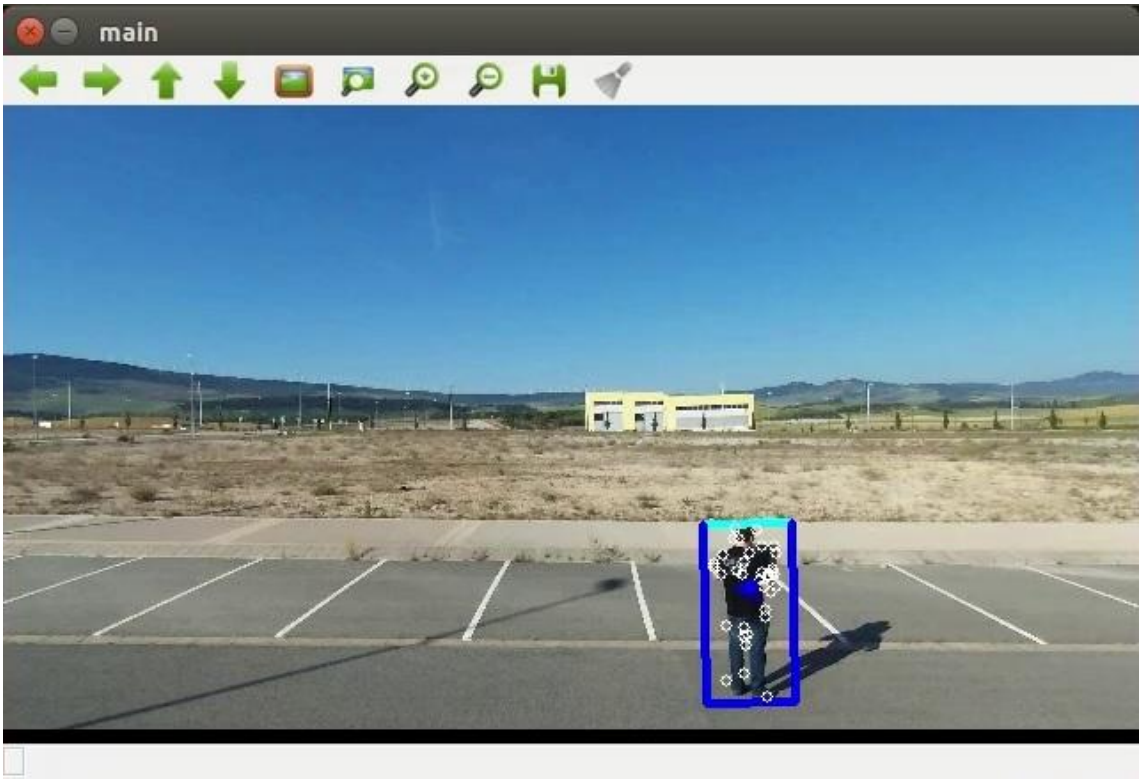

Figura 63. Tracking en exteriores

Como vemos a partir de las gráficas, el tracker funciona la mayor parte del tiempo. En este caso, como hemos aumentado la distancia de seguridad elevando el drone a unos 3 metros, la persona a seguir debe estar más alejada por lo que el tracker se puede ver influenciado por el tamaño de la zona a seguir.

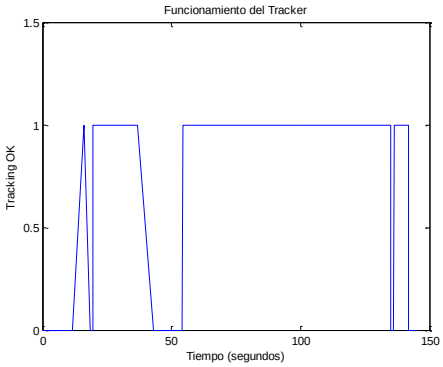

Figura 64. Funcionamiento del tracker

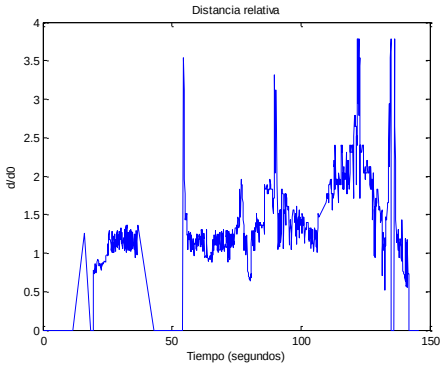

Figura 65. Distancia relativa

Cerca del segundo 50 vemos que el tracker no está funcionando, en este momento la aplicación ha perdido la comunicación con el drone por unos instantes y el drone continúa moviéndose. Al ser una aplicación con múltiples procesos, se ha dado el caso que el proceso de

lectura de datos del drone (telemetría y streaming) deja de recibir información del drone, pero el proceso de envío de comandos sigue activo enviando el último comando continuamente. Esto es un pequeño problema de la librería utilizada que tras la primera ronda de pruebas en exterior se ha solventado ordenando al drone que se mantenga quieto en Yaw y Pitch si pierde la telemetría.

En la Figura 65 vemos la distancia relativa respecto a la distancia inicial proporcionada por el tracker. Recordemos que la estrategia de control consiste en mantener una distancia entre uno y 1.3 veces la distancia inicial. En los momentos que el tracker no funciona guardamos la distancia relativa como cero. En la figura podemos ver picos que coinciden con la Figura 67 (Pitch normalizado), estos picos se deben a falsos positivos del tracker donde el tamaño del objeto es muy pequeño por unos instantes y por lo tanto la aplicación considera que está muy lejos y el valor de control del Pitch enviado es alto.

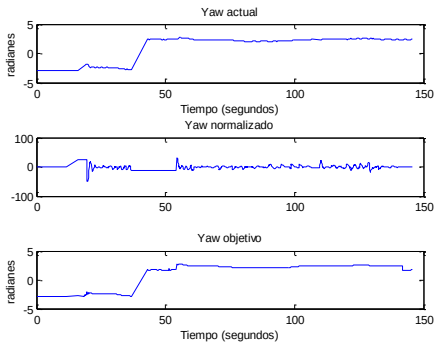

Figura 66. Yaw exteriores

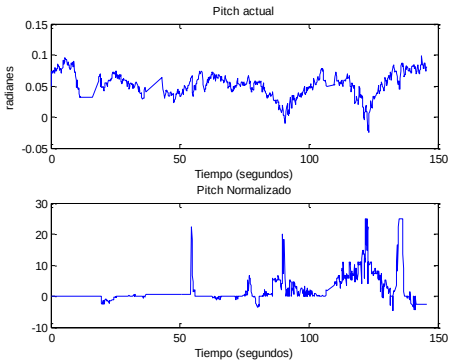

Figura 67. Pitch exteriores

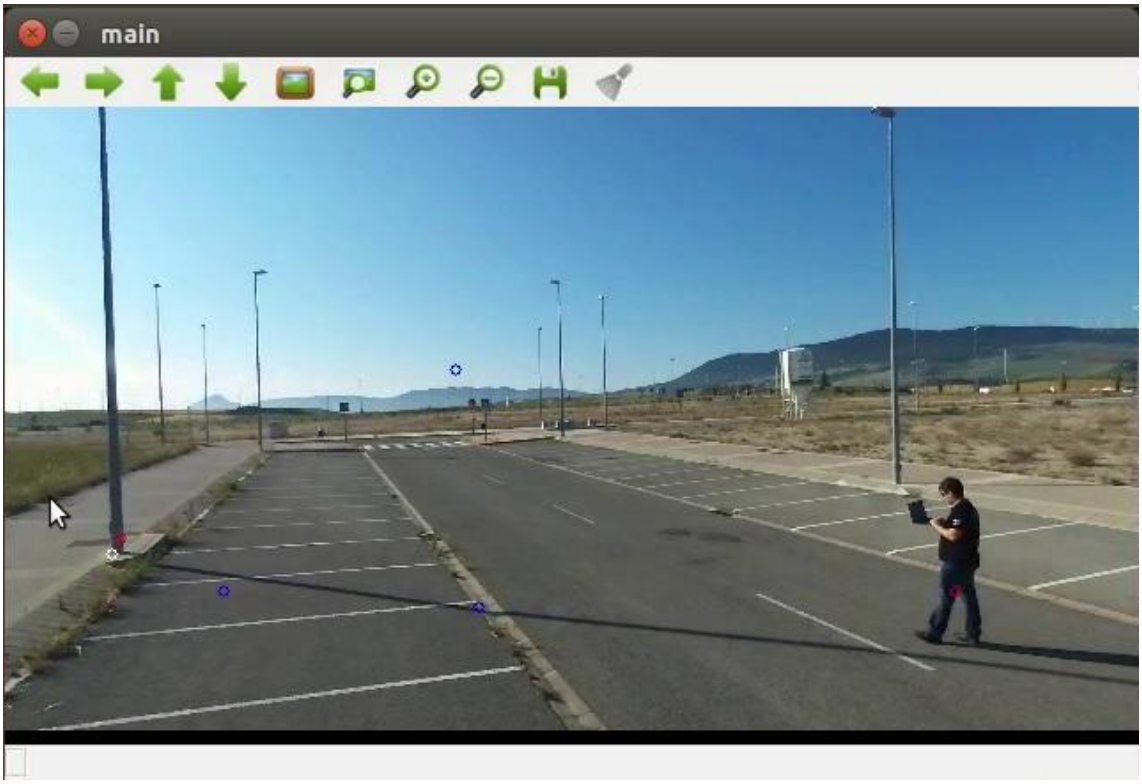

Figura 68. Tracking perdido por unos instantes

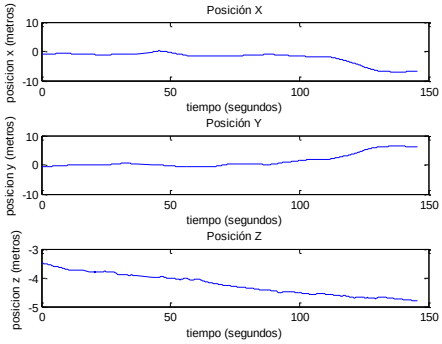

Figura 69. Posición estimada del drone

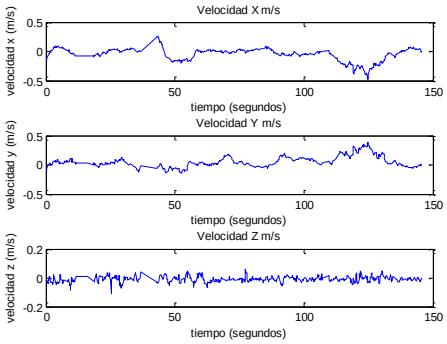

Figura 70. Velocidad estimada del drone

Salvo por la pérdida de conexión, el tracker funciona correctamente con una única persona utilizando el detector BRISK.

Tracking de una única persona en exteriores con detector BRISK, aumento de ganancia del controlador PITCH

D:\pruebas\outdoor\segunda prueba

Tras la primera prueba se decide cambiar la ganancia del controlador del PITCH ya que este estaba configurado para interiores realizando movimientos muy lentos. El drone está configurado para una inclinación máxima en PITCH de 15 grados, lo que equivaldría a una velocidad máxima de unos 6 m/s.

La inclinación máxima del drone que es equivalente a la velocidad máxima que puede alcanzar, es un parámetro que se configura en los comandos iniciales cuando comienza la comunicación entre el drone y la aplicación de control. La inclinación máxima del Bebop Drone es de 30 grados que equivale a una velocidad máxima de 11.11 m/s.

En pruebas realizadas en laboratorio se ha utilizado una ganancia del controlador Pitch muy pequeña de forma que el Pitch normalizado fuese pequeño y por lo tanto el movimiento del drone hacia delante o hacia atrás fuese pequeño. Esta ha sido una forma de evitar posibles colisiones con objetos dentro del laboratorio en las pruebas que se ha permitido el movimiento Pitch. El objetivo de estas pruebas ha sido comprobar si alejándose el objeto a seguir del drone el controlador del Pitch funcionaba.

A partir de estas prueba, utilizamos un valor de K para la Ecuación 21 de K=50. Utilizaremos esta ganancia en el resto de pruebas.

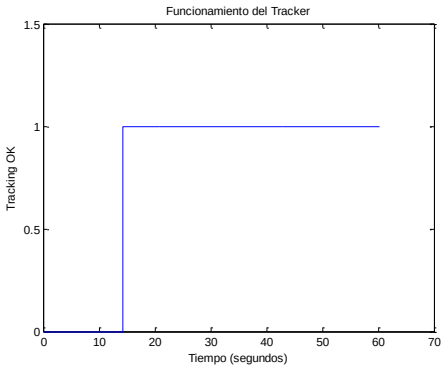

Figura 71. Funcionamiento del tracker

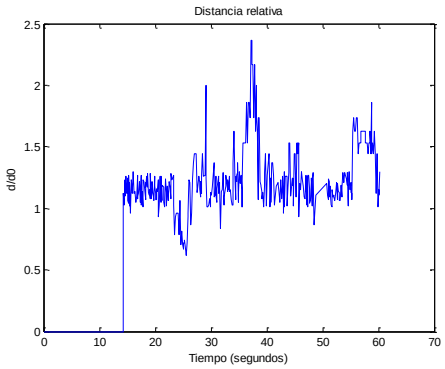

Figura 72. Distancia relativa

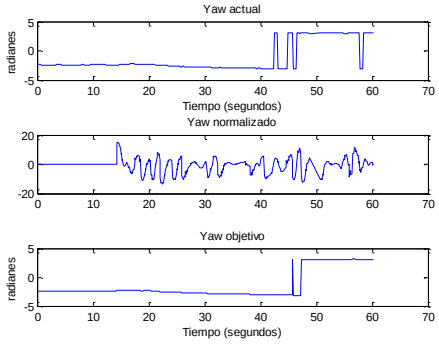

Figura 73. Yaw

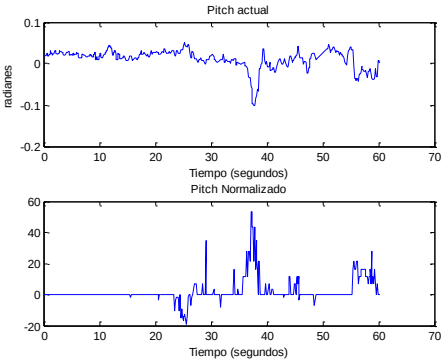

Figura 74. Pitch

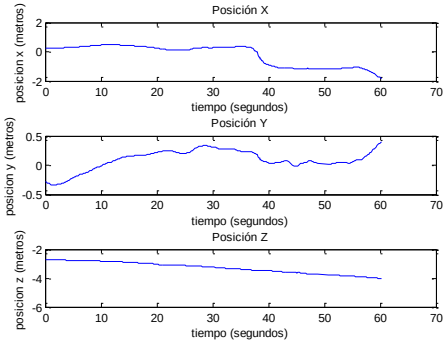

Figura 75. Posición estimada

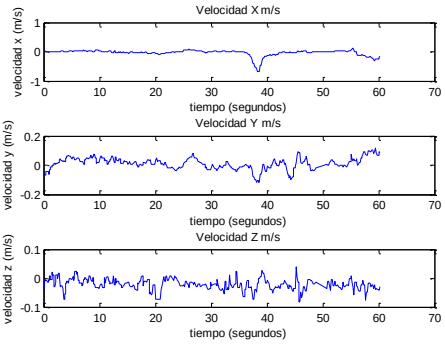

Figura 76. Velocidad estimada

Observando las gráficas anteriores podemos comprobar cómo en este caso el Pitch enviado al drone como señal de control es bastante mayor, lo cual aumenta su velocidad de movimiento. Sin embargo, podemos comprobar la dificultad de estimar la distancia desde el drone al sujeto a trackear. A medida que el sujeto se hace más pequeño en la imagen y la perspectiva hacia el drone cambia, el tracker con detector BRISK nos devuelve una anchura del objeto equivocada por lo que se dificulta el movimiento del Pitch y aumenta la distancia absoluta entre el objeto y el cuadricóptero.

En la siguiente imagen podemos ver como el tracker funciona correctamente pero nos da una mala medida de la anchura del sujeto a seguir. Ya que nuestra aplicación estima la distancia a través de la anchura del objeto a seguir es difícil mantener una distancia exacta en todo momento.

Figura 77. Mala detección de la anchura del sujeto

Tracking de una única persona en exteriores, disminuimos la distancia de seguridad

D:\pruebas\outdoor\tercera prueba

En la siguiente prueba realizada, hemos disminuido la altura del drone con el fin de que este tenga una mejor perspectiva del sujeto a seguir y exista una menor distancia. En esta prueba también hemos llevado al drone a una posición donde tiene el sol de frente con lo que este se queda cegado por el sol y el sujeto a seguir se ve oscurecido.

Figura 78. Tracking frente al sol

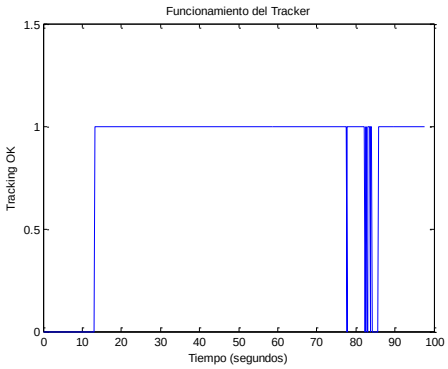

Figura 79. Funcionamiento del tracker

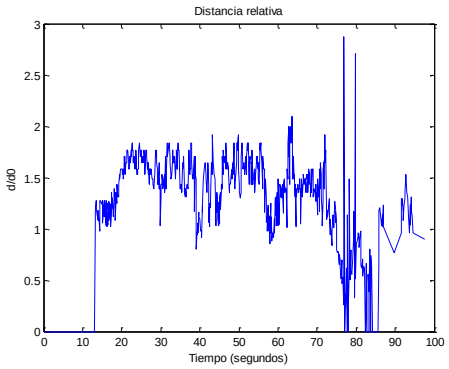

Figura 80. Distancia relativa

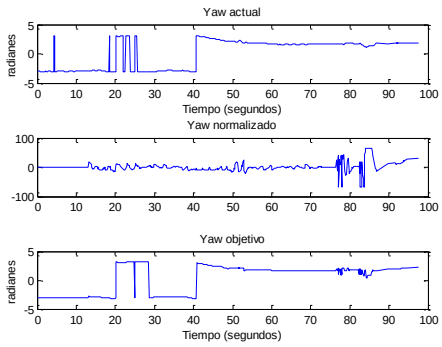

Figura 81. Yaw

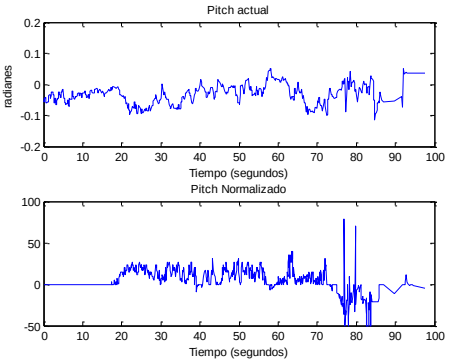

Figura 82. Pitch

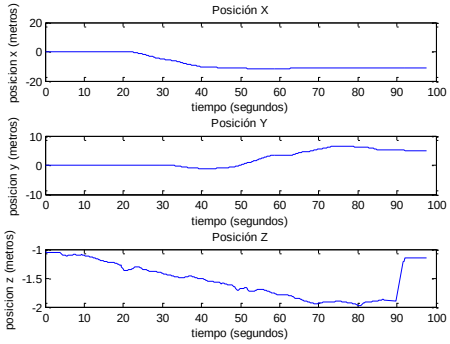

Figura 83. Posición estimada

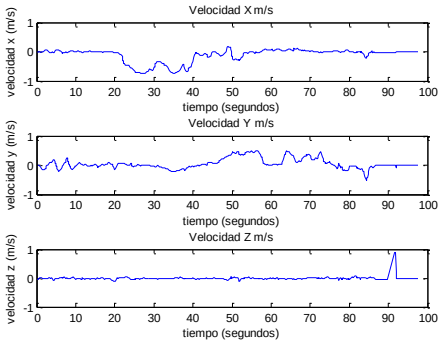

Figura 84. Velocidad estimada

La aplicación en este caso funciona correctamente salvo en las inmediaciones del segundo 80 donde estando el drone “cegado” por el sol, el sujeto se acerca y el tracker deja de funcionar correctamente.

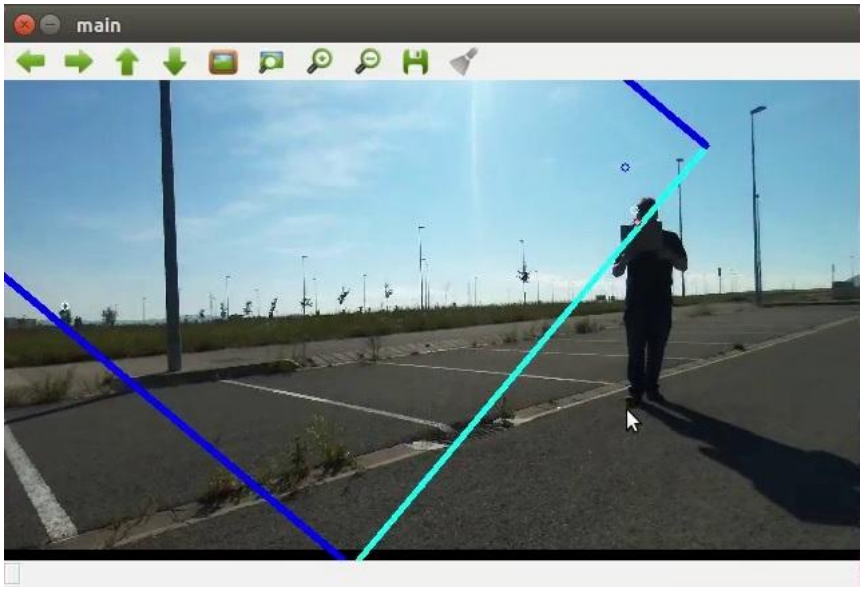

Figura 85. Mal funcionamiento del Tracker

Tracking de una persona con detector FAST

D:\pruebas\outdoor\cuarta prueba fast

Realizamos una nueva prueba para comprobar el funcionamiento del detector de características FAST en exteriores. Recordemos que utilizando este detector el tiempo de procesamiento aumenta considerablemente comparado con el detector BRISK, por lo que el comportamiento del drone puede ser más errático.

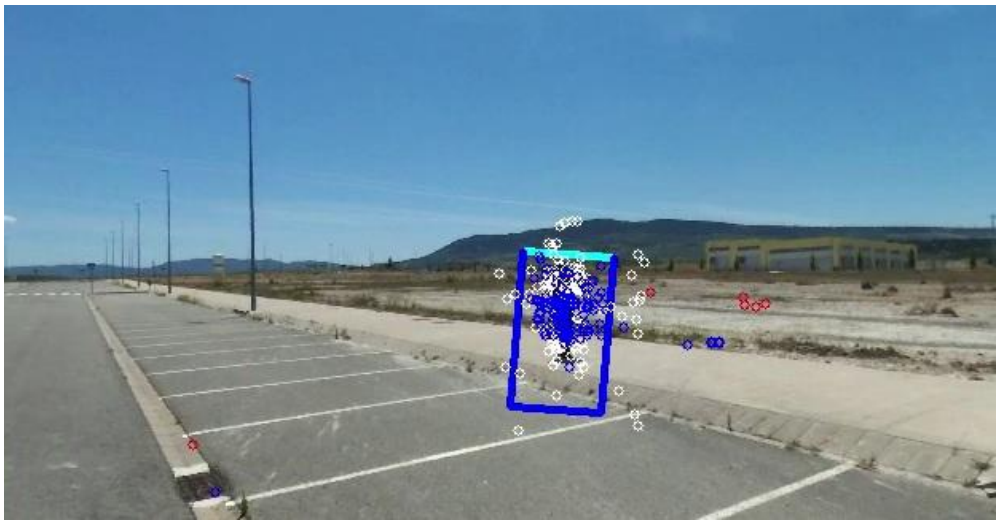

Figura 86. Tracking con detector FAST en exteriores

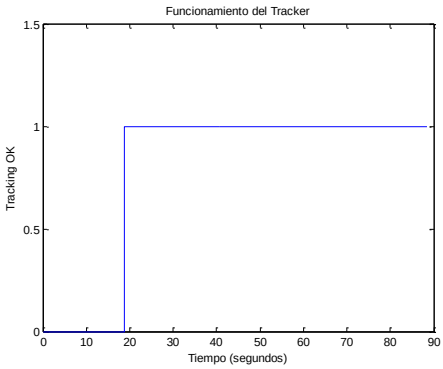

Figura 87. Funcionamiento del tracker

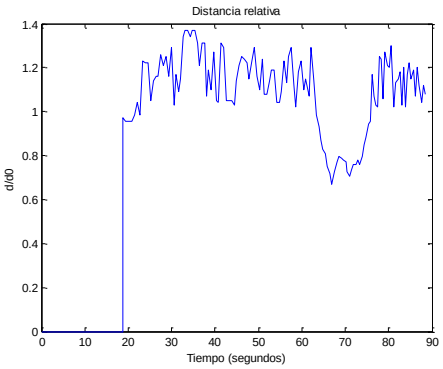

Figura 88. Distancia relativa

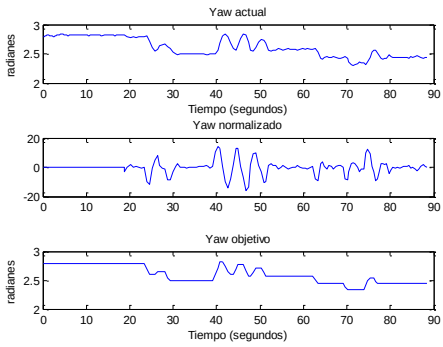

Figura 89. Yaw

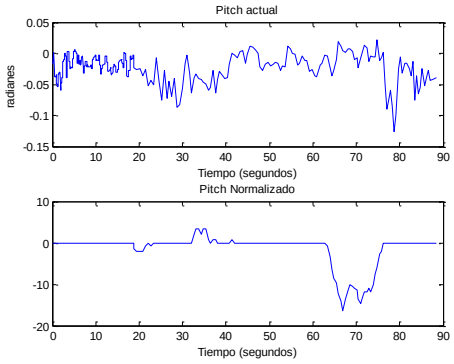

Figura 90. Pitch

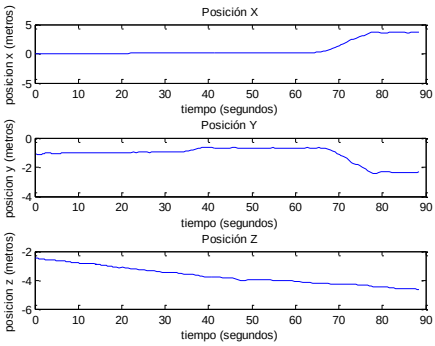

Figura 91. Posición estimada

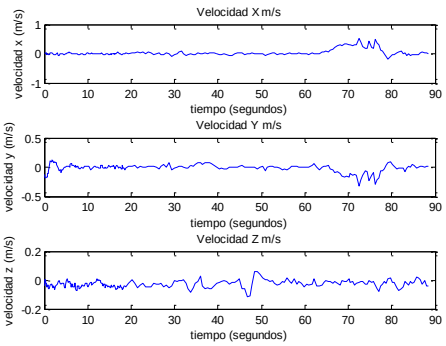

Figura 92. Velocidad estimada

Observamos que el tracker nos indica que funciona correctamente en todo momento, sin embargo, ya que este detector utiliza un mayor número de puntos característicos muchos de ellos se pierden durante el movimiento del sujeto a seguir. Por este motivo se dificulta la estimación de la distancia con lo que se aumenta el rango donde el drone no se mueve respecto al sujeto a seguir. En cambio el movimiento de Yaw funciona correctamente aunque mucho más escalonado por las diferencias que surgen en el tiempo que tarda el tracker en procesar cada imagen.

Debido al alto tiempo de procesamiento del tracker con este detector, es complicado mantener un rango de distancia controlado entre el drone y el objetivo. Como el objetivo está continuamente en movimiento y el tracker procesa unas dos imágenes por segundo, el objetivo a seguir se observa cada vez más pequeño en la imagen si este se aleja ya que el drone no puede actualizar la posición relativa del objetivo y moverse en consecuencia con la frecuencia necesaria.

Un posible método de solventar este problema sería predecir la dirección y velocidad a la que se mueve el objetivo y controlar el drone usando esta predicción. Sin embargo, utilizando este método correríamos el riesgo que el objetivo desaparezca de la escena.

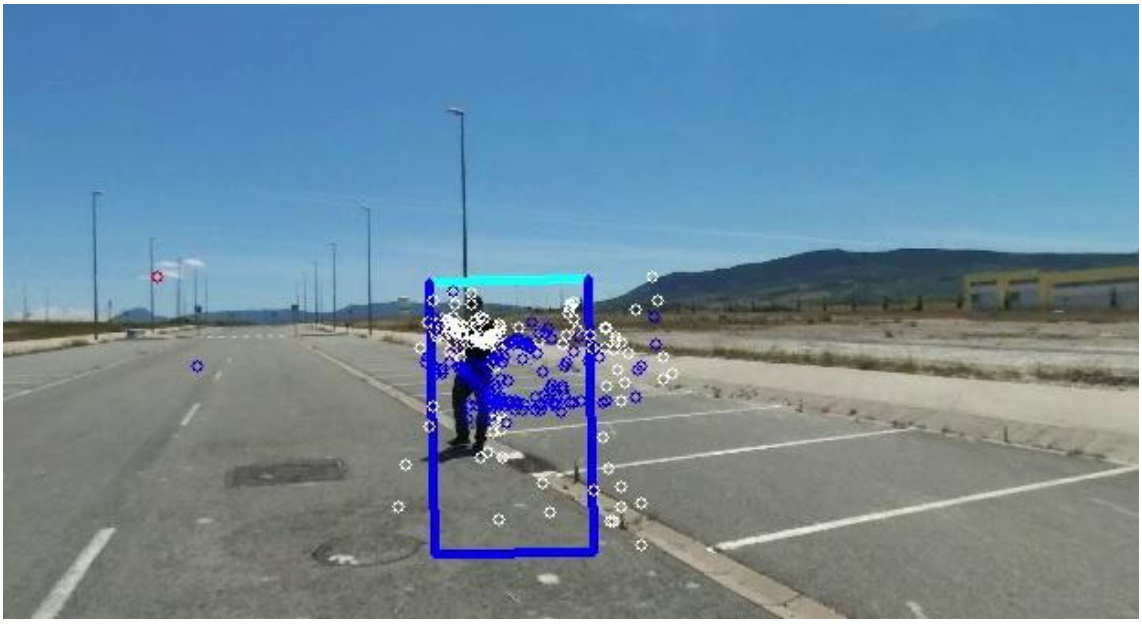

Figura 93. Tracker perdiendo puntos

6.4 Tracking en exterior cruce de personas

En este apartado vamos a comprobar qué ocurre en condiciones no controladas cuando una persona se cruza delante del sujeto a seguir. Lo haremos utilizando los dos detectores de características.

Cruce de personas con detector BRISK, prueba 1

D:\pruebas\outdoor\quinta prueba cruce1

En esta primera prueba de cruces de personas, una segunda persona ha pasado dos veces cerca del sujeto a seguir. La primera pasada el tracker ha funcionado correctamente como podemos ver en la Figura 94 . Sin embargo, en la segunda vuelta, algunos puntos característicos son arrastrados por la segunda persona (Figura 95), lo cual ensancha el rectángulo del tracker y en consecuencia el drone vuela hacia atrás. Por este motivo al alejarse el drone luego no es capaz de volver a encontrar al sujeto que estaba siguiendo, Figura 96.

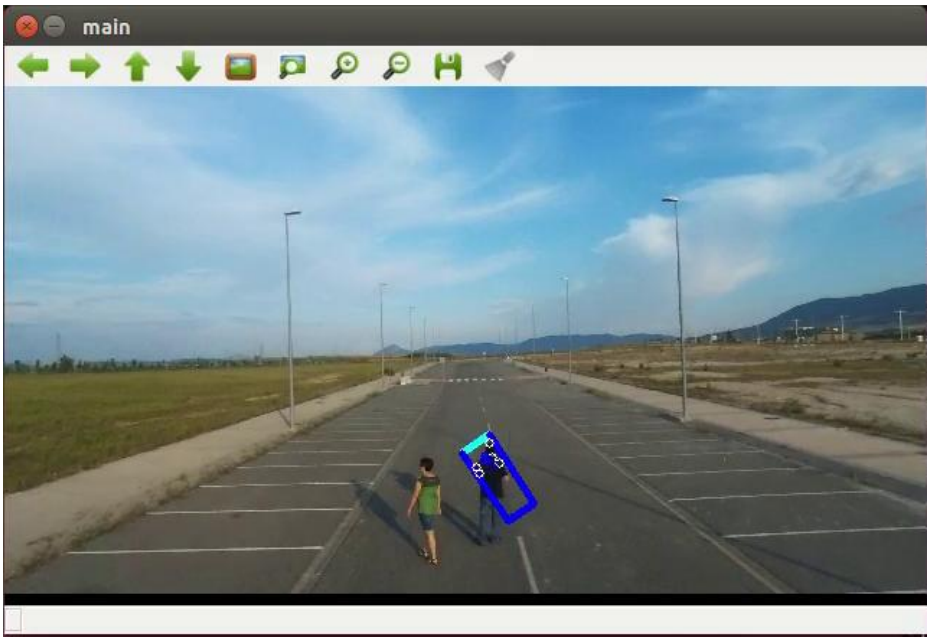

Figura 94. Tracker funcionando correctamente

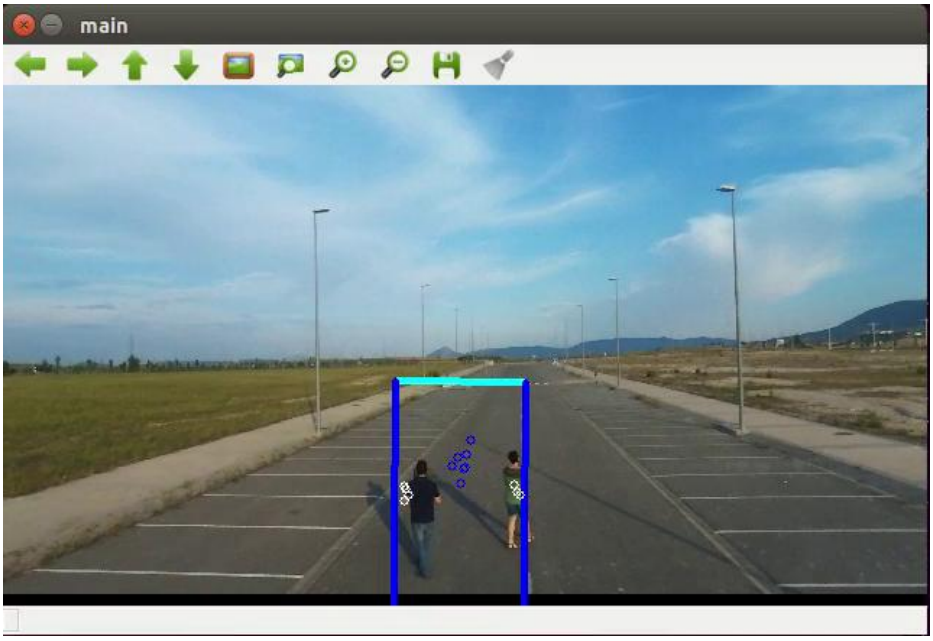

Figura 95. Deterioro al realizar el cruce

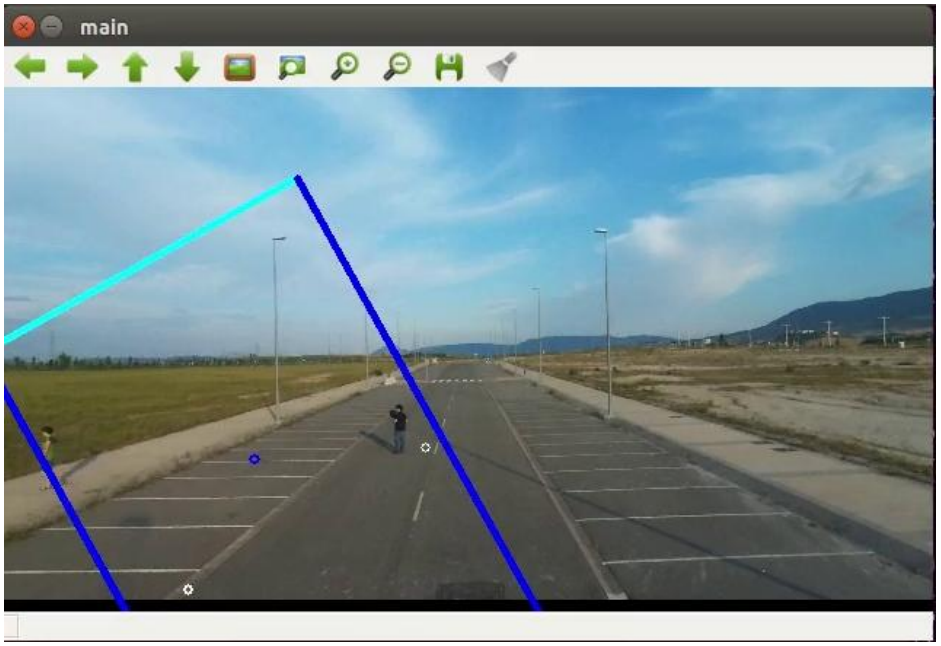

Figura 96. Tracker perdido

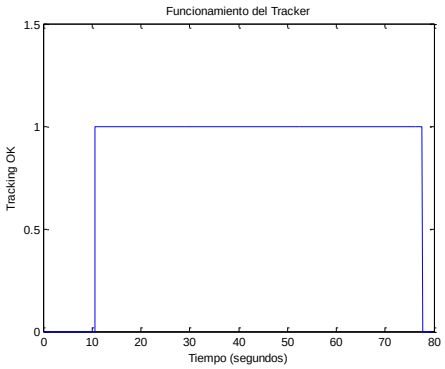

Figura 97. Funcionamiento del tracker

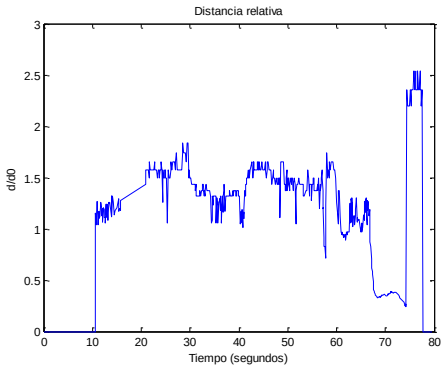

Figura 98. Distancia relativa

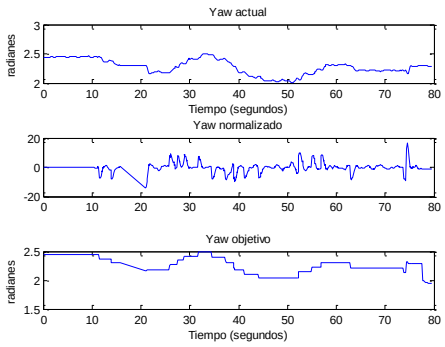

Figura 99. Yaw

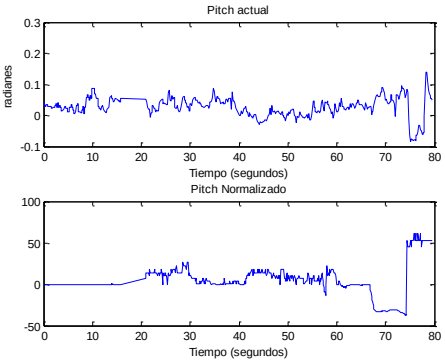

Figura 100. Pitch

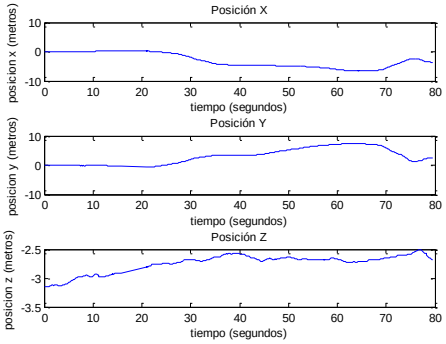

Figura 101. Posición estimada

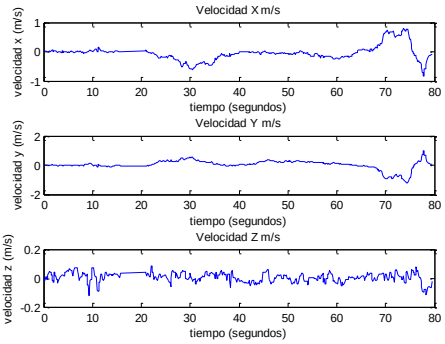

Figura 102. Velocidad estimada

A pesar de que la gráfica de la Figura 97 nos indica que el tracker ha funcionado correctamente, como ya hemos mencionado al producirse uno de los cruces la zona de la imagen a seguir se ensancha y ya no debería de ser válida. Según vemos en la gráfica del Pitch enviado, esto sucede pasado el segundo 60, aunque el funcionamiento del tracker no nos indica que este falla hasta casi el segundo 80.

En este caso para tener un tracking correcto sería necesario identificar en la escena que existe otra persona, para ello se podría utilizar el detector de personas de OpenCV (Detection s.f.) y aplicar el algoritmo de tracking únicamente a la zona donde tenemos las dos personas, de esta forma podríamos limitar la dispersión de puntos con el detector Fast. Otra opción sería limitar la velocidad de movimiento del drone en caso de que haya dos personas. En ambas opciones es necesario que el drone sepa que existe otra persona en su alcance visual.

Cruce de personas con detector BRISK, prueba 2

D:\pruebas\outdoor\sexta prueba cruce2

Realizamos otra prueba para asegurarnos del comportamiento del tracker y ver si tenemos un resultado positivo al realizar los cruces.

Figura 103. Tracker perdido

En esta ocasión el tracker ha dejado de funcionar al cruzarse la segunda persona tal y como podemos ver en la Figura 104. Una vez pasado el cruce, en esta ocasión el tracker sí que ha sido capaz de volver a encontrar al sujeto a seguir.

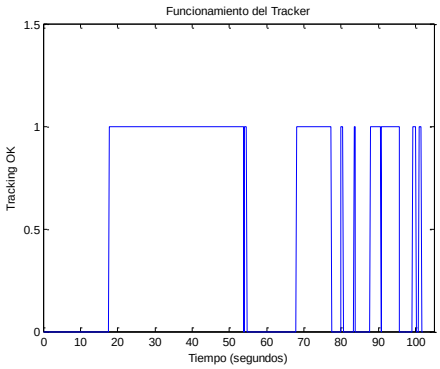

Figura 104. Funcionamiento del tracker

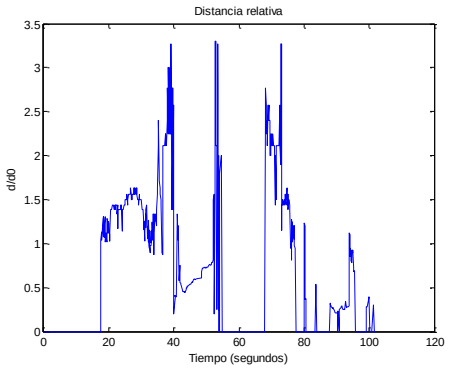

Figura 105. Distancia Relativa

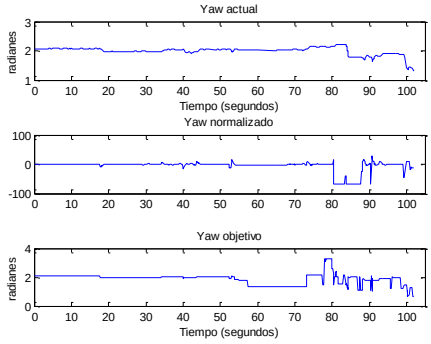

Figura 106. Yaw

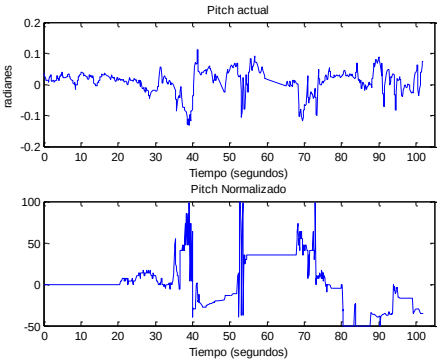

Figura 107. Pitch

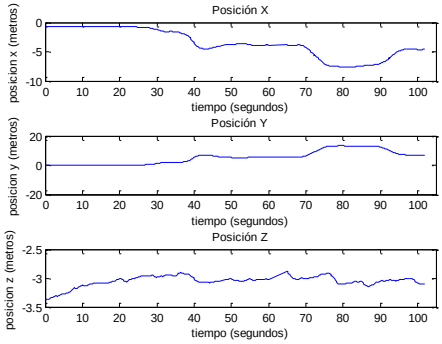

Figura 108. Posición estimada

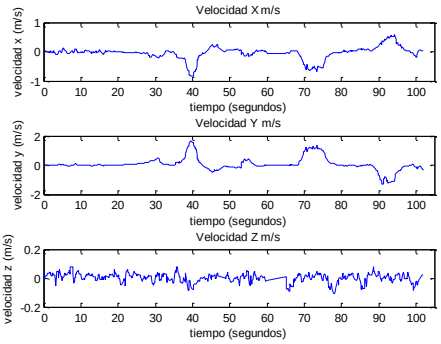

Figura 109. Velocidad estimada

Figura 110. Tracker vuelve a encontrar al sujeto

Tracking exterior cruce de personas FAST

D:\pruebas\outdoor\septima cruce fast

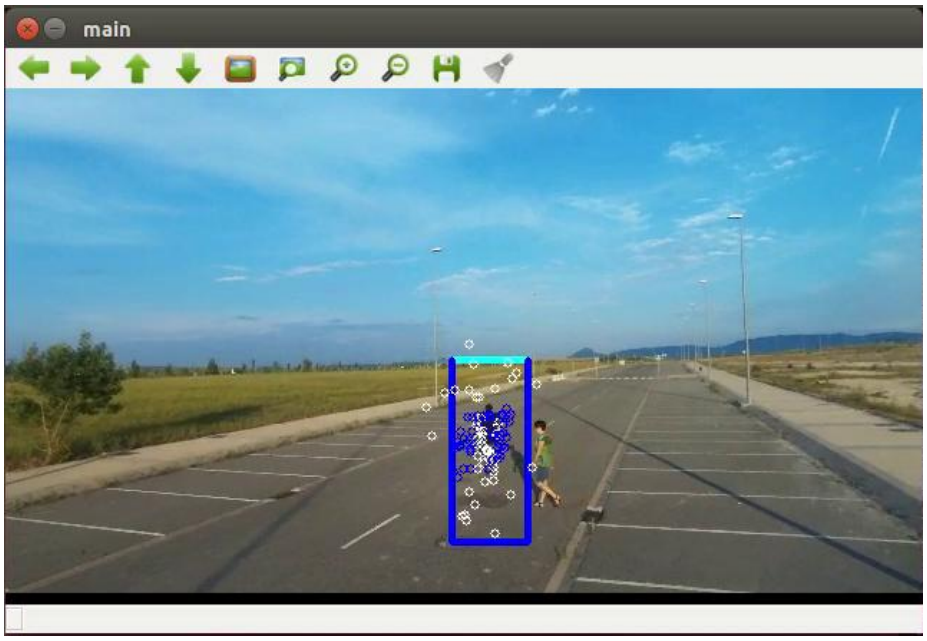

Figura 111. Cruce de personas con detector FAST

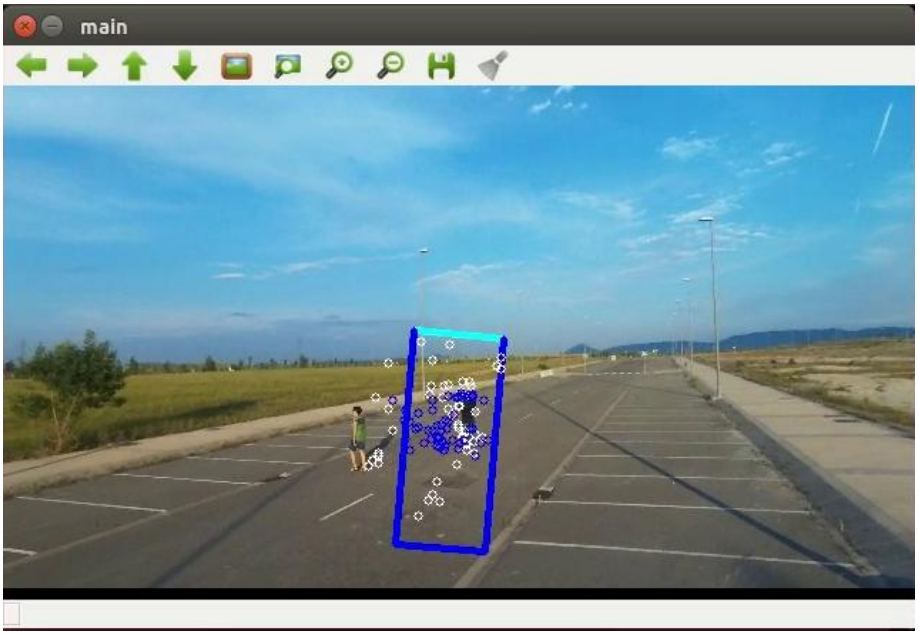

Figura 112. Los puntos característicos se diluyen

Figura 113. El tracker ha perdido al sujeto a seguir

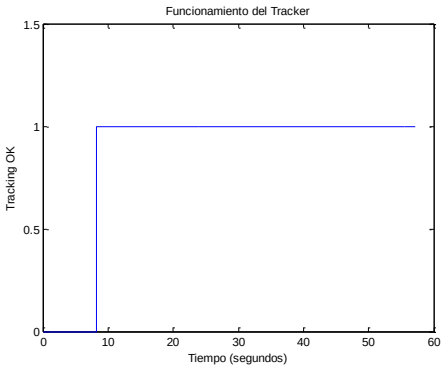

Figura 114. Funcionamiento del tracker

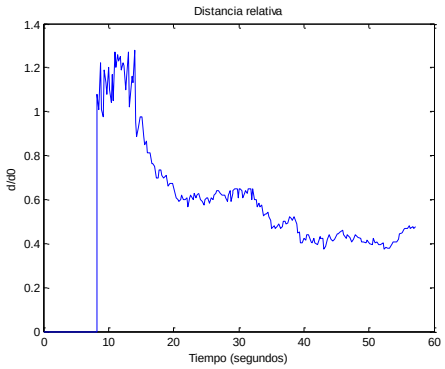

Figura 115. Roll

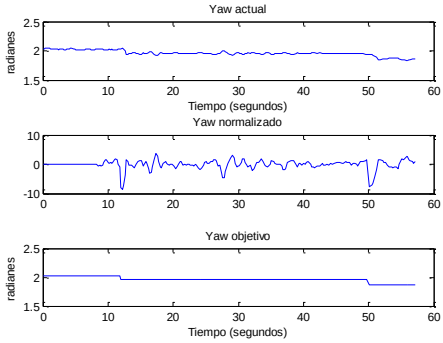

Figura 116. Yaw

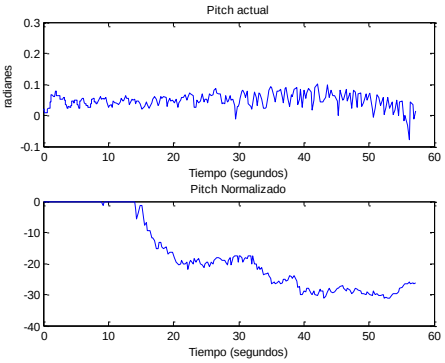

Figura 117. Pitch

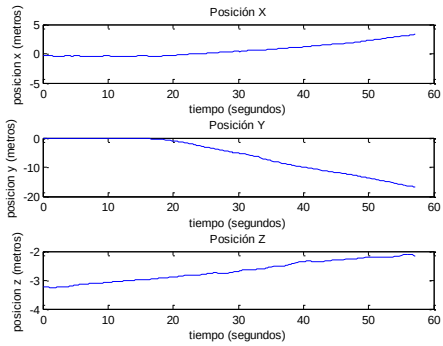

Figura 118. Posición estimada

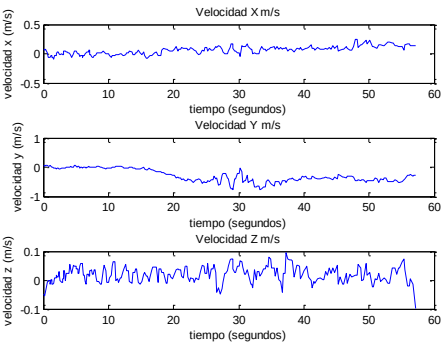

Figura 119. Velocidad estimada

El detector FAST no ha funcionado tan bien como cabría esperar ante el cruce entre personas. Los puntos característicos que ha detectado en el tracking se dispersan en elementos de la escena haciendo imposible una buena estimación de la distancia al objetivo. Para solventar este tipo de situaciones sería necesario implementar un método de contraste que permitiese acotar en la zona dada por el detector la anchura en píxeles de la persona. Una posibilidad sería utilizar el detector de personas de OpenCV (Detection s.f.). En el capítulo de conclusiones se establecerán una serie de líneas de futuro trabajo ofreciendo distintas soluciones a los problemas que se han encontrado en este proyecto.

6.5 Tracking exterior a un objeto voluminoso

Ya que el tracker que estamos utilizando es genérico y puede seguir a un número muy diferente de objetos, vamos a probar a seguir un objeto voluminoso como es un coche. Como en todas las pruebas lo haremos con los dos detectores de características que estamos utilizando.

Tracking exterior a coche con detector BRISK

D:\pruebas\outdoor\octaba coche brisk

Las condiciones de iluminación en las que se realiza la prueba son adversas, atardeciendo con el sol de frente.

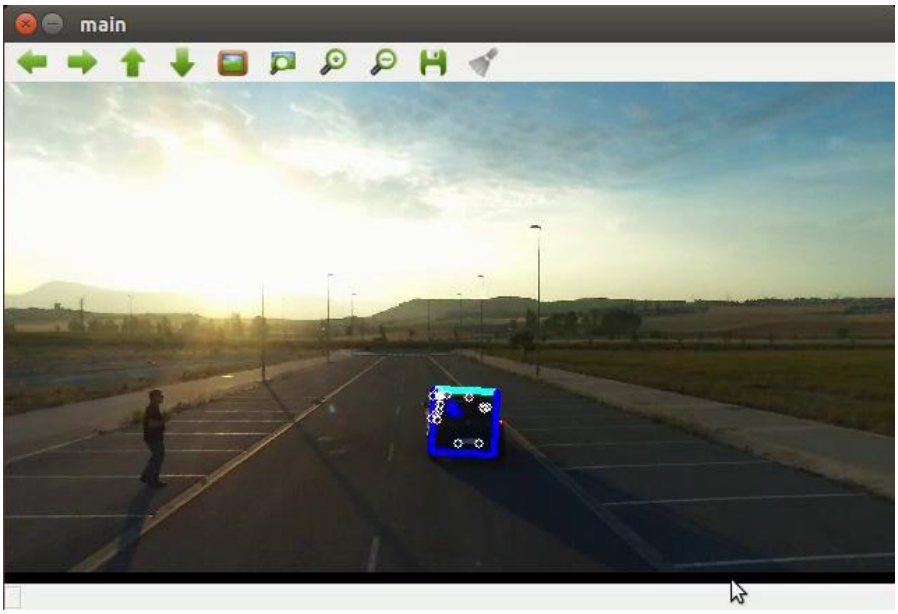

Figura 120. Tracking a un coche, detector BRISK

A pesar de que el tracker funciona correctamente los primeros metros de la prueba, el drone adelanta al coche en un momento de la prueba, por lo que pierde el objeto a seguir tal y como vemos en la Figura 121 y Figura 122.

Existen varios motivos por los que ha sucedido este fallo en la prueba, en primer lugar debido al contraste de iluminación, (el drone miraba hacia el sol) la estimación de la distancia no es buena ya que el vehículo se encontraba oscurecido y los puntos característicos detectados eran muy inestables. En segundo lugar, la velocidad máxima del drone no estaba ajustada para

seguir a un vehículo y la reacción del drone ha sido lenta en comparación con el vehículo. Por último, el recorrido realizado era cuesta abajo, el Bebop drone tiene un comando para establecer la altura de vuelo al despegue, pero esta no se mantiene relativa al suelo si existe un desnivel, siendo el recorrido cuesta abajo el drone aumenta de altura respecto al suelo. Suponiendo que la distancia absoluta al objetivo es constante en todo momento al aumentar la altura del drone respecto al suelo este está más cerca del objetivo en el eje x, esto ocurre hasta que el objetivo desaparece en parte de la escena y el tracker comienza a dar falsos positivos.

Si observamos la gráfica de la Figura 127 podemos observar que según los sensores de altitud el drone pierde altura a medida que se realiza la prueba, de los 3 metros iniciales hasta cerca de los dos metros al finalizar la prueba. Sin embargo, se puede apreciar claramente en las imágenes que la distancia al suelo es mayor al finalizar la prueba. Según esa gráfica podemos suponer que en realidad el Bebop drone no tiene un control establecido para la altura de vuelo. Por otro lado, el altímetro del drone parece establecerse a cero en la posición de despegue, si el recorrido es cuesta abajo sus medidas en todo momento serán sobre la altitud de despegue, por lo que sería imposible realizar un control de la altitud sobre el suelo.

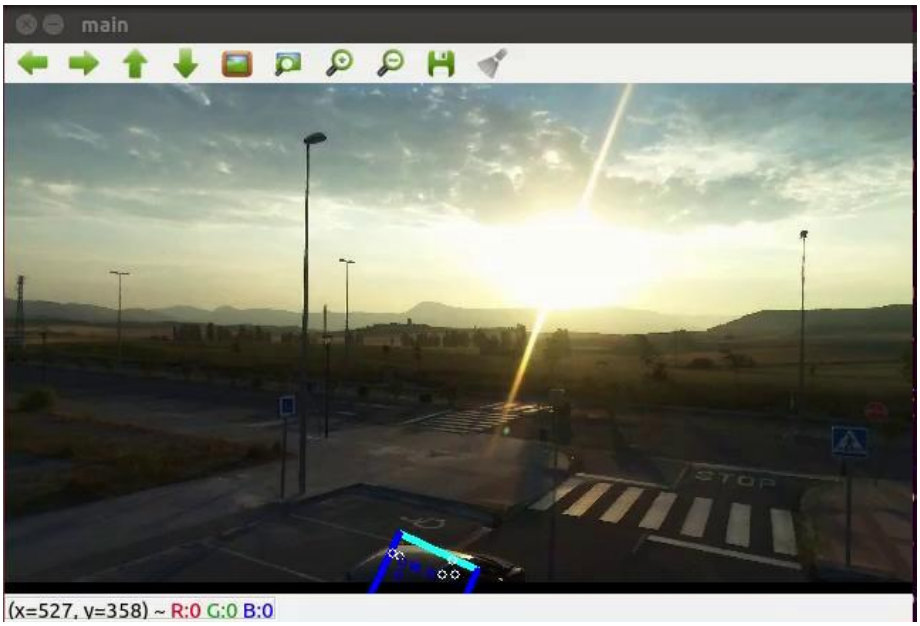

Figura 121. El drone adelanta al coche

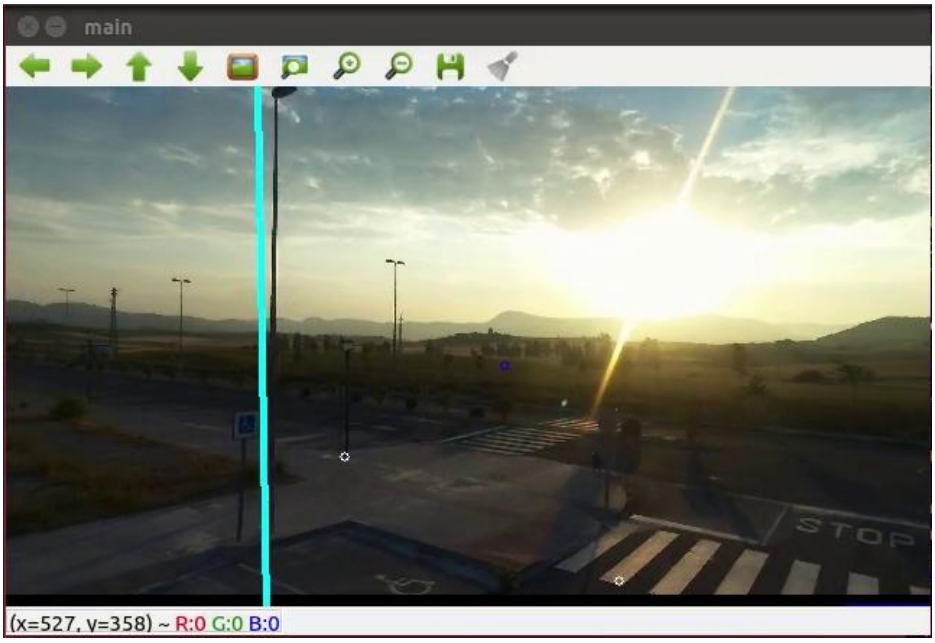

Figura 122. El tracker se pierde

Una vez está adelantando al coche se envía el comando de emergencia de aterrizar, por este motivo en la gráfica de la Figura 123 vemos que el tracker está funcionando en todo momento ya que este nos ha detectado el objeto aunque en los últimos frames se trate de un falso positivo.

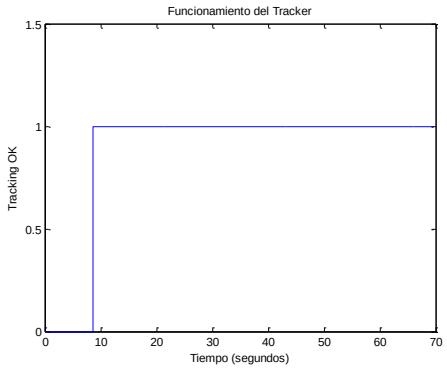

Figura 123. Funcionamiento del tracker

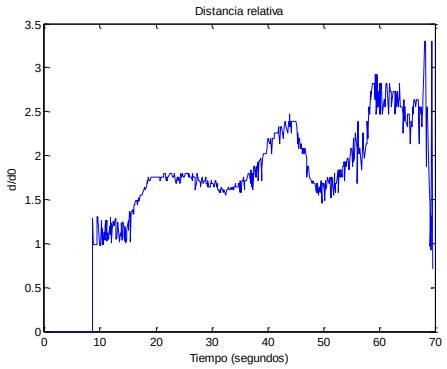

Figura 124. Distancia relativa

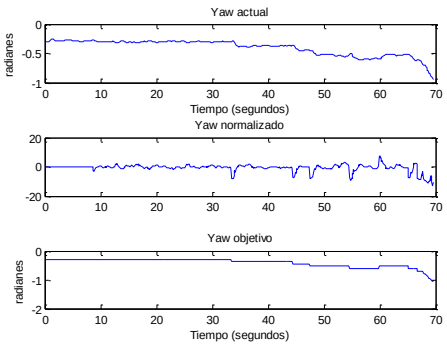

Figura 125. Yaw

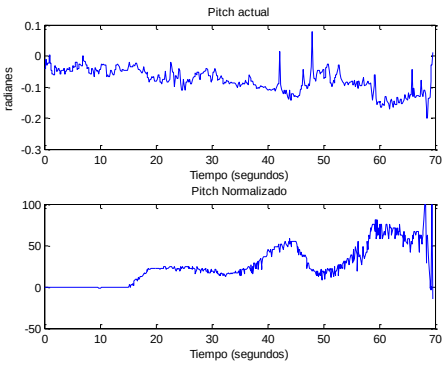

Figura 126. Pich

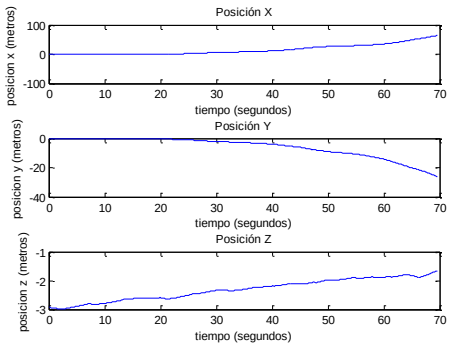

Figura 127. Posición estimada

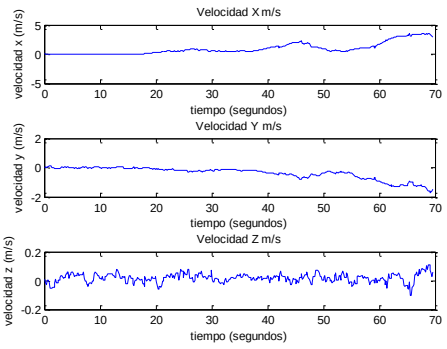

Figura 128 Velocidad estimada

Tracking exterior a coche FAST

D:\pruebas\outdoor\novena coche fast

Al contrario que en la anterior, esta prueba se realiza con unas condiciones de iluminación más favorables.

Figura 129. Tracking a coche con detector FAST

A medida que el vehículo se aleja, los puntos característicos del tracker se van expandiendo en la imagen como podemos ver en la Figura 130. Esto hace que la zona a seguir en la imagen aumente, en consecuencia, el drone al estimar que el objeto está cerca se va hacia atrás.

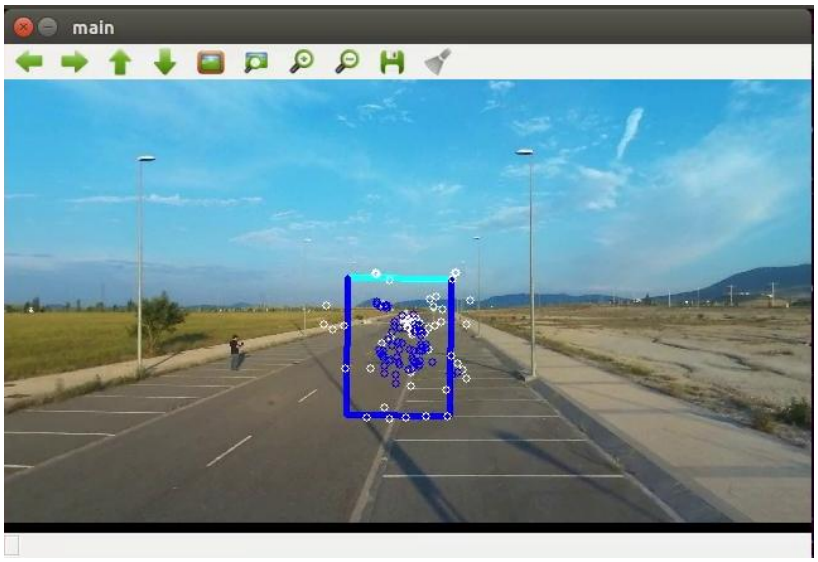

Figura 130. Puntos característicos expandiendose en la imagen

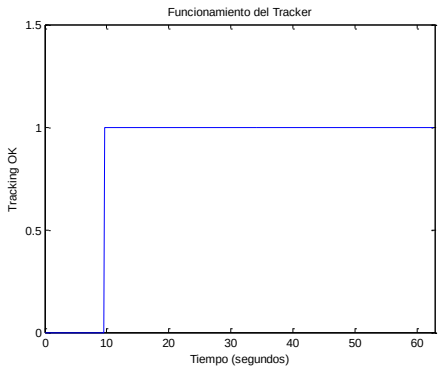

Figura 131. Funcionamiento del tracker

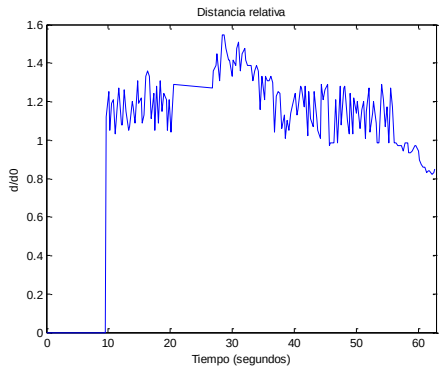

Figura 132. Distancia relativa

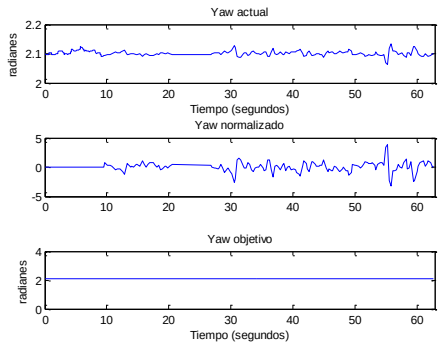

Figura 133. Yaw

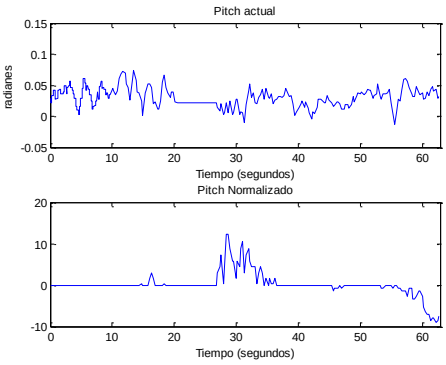

Figura 134. Pitch

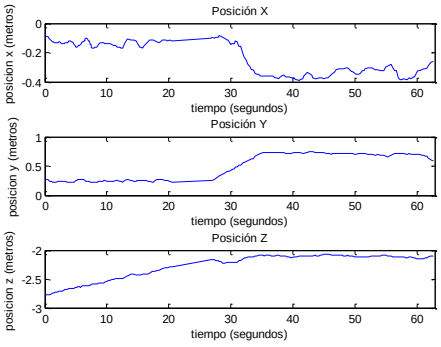

Figura 135. Posición estimada

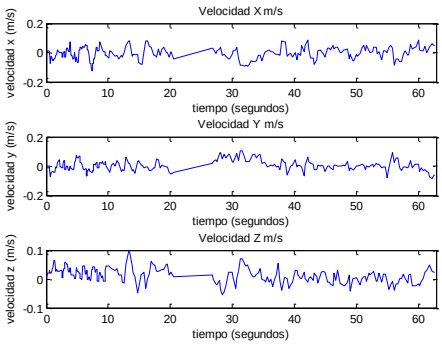

Figura 136. Velocidad estimada

7 Conclusiones

En este proyecto se ha integrado un algoritmo de tracking genérico para controlar un cuadricóptero ordenándole que siga un objeto o persona. A pesar de que se han hecho pruebas de campo en unas condiciones totalmente descontroladas, se han tenido muy buenos resultados en cuanto al control del alineamiento del robot respecto al objeto a seguir, quedando el objeto siempre centrado en la imagen que nos ofrece el robot. No podemos decir lo mismo del control de la distancia, ya que la estimación de la distancia a la que se encuentra un objeto a través del algoritmo de tracking utilizado no ha sido tan buena.

El algoritmo de tracking utilizado se basa en la búsqueda de puntos característicos en toda la imagen de puntos característicos que previamente había encontrado en una zona de la imagen delimitada por el usuario, donde se encuentra el objeto a seguir. Como los puntos característicos se pueden dispersar al tratarse de imágenes en movimiento en un entorno no controlado, la zona que nos devuelve el tracker utilizado no está siempre limitada a la zona de tracking inicial. Por este motivo, la estimación de la distancia entre el cuadricóptero y el objeto a seguir no ha sido buena en las pruebas de exterior realizadas.

En el algoritmo de tracking utilizado, se han probado dos detectores de puntos característicos el detector BRISK y el FAST. El detector FAST que encuentra un número mayor de puntos característicos tiene dos inconvenientes, por un lado al tener que buscar más puntos característicos en toda la imagen el tiempo de procesamiento es mayor, por lo que el movimiento del robot es más escalonado, ya que hay instantes de tiempo donde el robot no “sabe” dónde se encuentra el objeto a seguir. El otro inconveniente tiene relación con lo ya mencionado en el párrafo anterior, al dispersarse los puntos característicos en toda la imagen la estimación de la distancia relativa entre el drone y el objeto ha sido peor en las pruebas en exterior. A pesar de la mala estimación de la distancia, este detector es más fiable a la hora de volver a encontrar el objeto en la escena si este desaparece.

El detector BRISK por el contrario es mucho más rápido al usar menos puntos característicos, este tiene el inconveniente de que estos puntos se pierdan siguiendo objetos relativamente pequeños en la imagen. También con este detector es más sencillo que el algoritmo de tracking pierda a la persona ante deformaciones, por ejemplo que se gire. En cambio, a la hora de seguir objetos grandes como un coche, este detector ha tenido mejores resultados que el detector FAST a pesar de que en la prueba realizada las condiciones de iluminación sean peores.

En general, el algoritmo de tracking con el detector FAST es más fiable que el BRISK, ya que al apoyarse en más puntos característicos, es más fácil que posicione correctamente el objeto a seguir en la imagen y es menos propenso a falsos positivos que el detector BRISK.

Podemos decir que el algoritmo de tracking CMT es bastante robusto siendo inmune a cambios de iluminación y sombras. También tenemos que responde adecuadamente ante la pérdida del objeto a seguir volviéndolo a encontrar en la imagen rápidamente.

Como conclusión general podemos decir que el algoritmo de tracking utilizado podría ser válido para controlar un cuadricóptero a través de análisis de imagen. Es necesario mejorar la estimación de la distancia a la que se encuentra el objeto a seguir para tener un control fiable del robot.

Por otro lado, es imposible determinar cuando el algoritmo de tracking utilizado nos da falsos positivos sin aplicar ningún otro método de contraste. Este hecho hace que en ocasiones el movimiento del drone sea errático y aumente progresivamente el error puesto que el objetivo a seguir se puede alejar.

Si observamos los vídeos de demostración de los investigadores que han desarrollado el tracker CMT podemos ver que obtienen muy buenos resultados, sin embargo esos experimentos no se han realizado en aplicaciones de tiempo real y podemos ver que como máximo estos investigadores procesan 4 frames por segundo en los vídeos que han publicado. En nuestro caso, podemos procesar dos o tres frames por segundo y descartamos los frames entre medias por lo que el algoritmo pierde la memoria respecto al último frame procesado ya que el objetivo se ha podido mover mucho en ese tiempo.

7.1 Problemas encontrados en el desarrollo

El mayor de los problemas encontrados para desarrollar este proyecto se ha centrado en la escasa documentación existente del SDK del Bebop Drone. De hecho, como se ha comentado, ha sido necesario el uso de técnicas de Ingeniería inversa para descubrir el protocolo de comunicaciones correcto del drone.

Uno de los principales motivos de seleccionar este drone para el desarrollo fue la existencia de un SDK público. Pensando que la implementación de este SDK sería como el del AR Drone 2.0 (donde está perfectamente documentado y encontramos numerosos ejemplos de las posibilidades existentes), se decidió utilizar este drone como elemento fundamental del desarrollo del proyecto.

El siguiente problema a resolver con este drone ha sido disponer de una correcta recepción del Streaming de vídeo. Sin esto sería imposible realizar un control autónomo a través de análisis de imagen. Si bien se ha conseguido tener un streaming de vídeo con una calidad de imagen y retardo aceptables, ha sido imposible evitar la pérdida del streaming en algunas pruebas. Este drone estabiliza la imagen por software, cuando el drone realiza movimientos violentos, por ejemplo una pequeña ráfaga de viento, la CPU del drone se satura estabilizando la imagen y

deja de enviar tanto el streaming como la telemetría del drone. Este problema no controlable hace que la aplicación de control quede en standby hasta que reciba una cadena de frames decodificables (típicamente 1 segundo) con lo cual el tiempo que el drone está sin comunicación con la aplicación de control es muy alto (unos 4 o 5 segundos) para nuestros propósitos.

Por último la realización de pruebas ha sido una labor ardua ya que la batería de este drone únicamente nos ofrece un tiempo de vuelo de unos 10 minutos. Por este motivo sólo se han podido realizar dos o tres pruebas por cada carga de batería.

7.2 Posibles líneas de investigación

El futuro trabajo dentro de este proyecto pasa por mejorar la estimación de la distancia entre el objeto a seguir y el drone. Para ello sería necesario implementar o integrar métodos adicionales al tracker CMT que nos permitan estimar mejor las distancias.

La solución ideal pasaría por tener otro sensor que, o bien nos ofreciese la distancia correcta al objetivo, o bien tener otro sistema de captura de datos con el que poder contrastar los resultados obtenidos con el tracker CMT.

Al finalizar el desarrollo de este proyecto, el fabricante DJI ha introducido en el mercado una plataforma de desarrollo para drones llamada Matrice 100 (DJI s.f.), se trata de un cuadricóptero programable al que se le pueden añadir diferentes sensores con los que poder conseguir vuelos completamente autónomos. Junto a esta plataforma, DJI también ha desarrollado un sensor llamado Guidance (Guidance s.f.), este sensor constan de 5 cámaras estéreo y cinco sensores ultrasónicos de proximidad. Cuatro de las cámaras se posicionan a los cuatro laterales del cuadricóptero y una mirando hacia abajo. De esta forma es posible que el drone tenga una visión de 360 grados.

La plataforma desarrollada por DJI junto con el sensor Guidance parece ser la solución ideal para el desarrollo de un proyecto con los mismos objetivos que este. Al disponer de cámara estéreo es posible medir la distancia a un objetivo de una forma bastante precisa (Jernej Mrovlje 2008).

Volviendo a nuestro caso donde disponemos de una única cámara y el tracker CMT, sería necesario mejorar la velocidad de procesamiento en el caso del uso del algoritmo con el detector FAST. Esto se podría realizar implementando este algoritmo en GPU por ejemplo, con ello conseguiríamos procesar más frames por segundo y evitaríamos que el objeto a seguir se vaya haciendo cada vez más pequeño en la imagen si el objeto se está alejando del drone y una mejor estimación de la distancia. En las pruebas realizadas, utilizando el detector FAST se pueden procesar unos dos frames por segundo aproximadamente. Si el objeto a seguir tiene

una velocidad v, entre frame y frame se habrá movido v/2. Si en t=0 que coincida con la primera imagen a procesar el objeto empieza a moverse en la segunda imagen a procesar se habrá alejado v/2 por lo que será más pequeño en la imagen y más difícil de localizar para el algoritmo.

Una posible solución para acotar la zona donde se encuentra el objetivo y así estimar mejor la distancia sería utilizar otro método de detección junto con el tracker CMT. En el caso de que el objetivo a seguir sea una persona se podría utilizar el detector de personas de OpenCV junto con el tracker CMT. Durante la fase de análisis del estado del arte de este proyecto se han realizado pruebas con el detector de personas de OpenCV teniendo como entrada vídeos capturados con el Bebop Drone. Los resultados obtenidos no han sido nada buenos, por un lado el tiempo de procesamiento es muy elevado con lo que una combinación de ambos algoritmos sería imposible para una aplicación en tiempo real. Por otro lado, los resultados obtenidos con el detector de personas han sido poco fiables, detectando a la persona en muy pocos frames.

En el caso de poder reducir el tiempo de procesamiento del detector Fast, una posibilidad sería poder utilizar el algoritmo aplicando los dos detectores y comprobar la coherencia entre ellos, desarrollando el sistema que nos ofrezca la mejor distancia estimada.

Otra solución posible para obtener mejores resultados, sería el uso de tecnologías de inteligencia artificial. Esta consistiría en entrenar al tracker (Deva Ramanan 2007) con el objetivo a seguir desde diferentes perspectivas de forma que la aplicación “sepa” como es el objetivo a seguir y lo detecte en todo momento. Otra línea en la que trabajar sería la detección de obstáculos en la trayectoria que el robot. Aunque en las pruebas de exterior realizadas se haya tenido la precaución de realizarlas en un espacio más o menos abierto, podría darse el caso de que el robot siguiendo un objeto se encuentre en su camino con un obstáculo como puede ser un árbol o una farola.

8 Referencias

ARDrone-PCAP. *Github.* https://github.com/Zepheus/ardrone3-pcap.

Detection, OpenCV Object. http://docs.opencv.org/modules/gpu/doc/object_detection.html#gpu-hogdescriptor-getdefaultpeopledetector.

Deva Ramanan, David A. Forsyth, Andrew Zisserman. «Tracking People by Learning Their Appearance.» *IEEE Transactions on pattern analysis and machine intelligence*, 2007: Vol 29 .

Developers, Github Parrot. *ARDrone3 commands.* https://github.com/Parrot-Developers/libARCommands/blob/master/Xml/ARDrone3_commands.xml.

DJI. *Matrice 100.* http://store.dji.com/product/matrice-100.

Dlouhý, Martin. *Katarina Robot.* http://robotika.cz/robots/katarina/en.

Georg Nebehay, Roman Pflugfelder. «Clustering of Static-Adaptative Correspondences for Deformable Object Tracking.» *Computer Vision and Pattern Recognition*, 2015.

Guidance, DJI. *Guidance.* https://developer.dji.com/guidance/.

Jernej Mrovlje, Damir Vrančić. «Distance measuring based on stereoscopic pictures.» *9th International PhD Workshop on Systems and Control: Young Generation Viewpoint.* Izola, Slovenia, 2008.

Leutenegger, M. Chli and R. Y. Siegwart. «"BRISK: Binary Robust Invariant Scalable Keypoints ."» *Autonomous Systems Lab.* ETH Zürich, 2011.

Luukkonen, Teppo. «Modelling and control of quadcopter.» De Aalto University. Espoo, 2011.

Matlab. *Matlab Vision System Toolbox.* http://es.mathworks.com/products/computer-vision/.

Microsoft. *Kinect for windows.* https://www.microsoft.com/en-us/kinectforwindows/.

OpenCV 2.4.11, Documentation. *Introduction to Support Vector Machines.* http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html (último acceso: 06 de 2015).

OpenCV. *FAST Algorithm for Corner Detection.* http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html (último acceso: 04 de 2015).

Parrot. *Parrot Developers.* https://github.com/Parrot-Developers/ARSDKBuildUtils.

Quadcopter Dynamics. http://andrew.gibiansky.com/downloads/pdf/Quadcopter%20Dynamics,%20Simulation,%20and%20Control.pdf.

Sergei Lupashin, Markus Hehn, Mark W. Mueller, Angela P. Schoelling, Michael Sherback, Raffaello D´Andrea. «A platform for aerial robotics research and demostration: The Flying Machine Arena.» *Mechatronics Volume 24*, 2014: 41-54.

Slavomir Matuska, Robert Hudec, Miroslav Benco. «The Comparison of CPU Time Consumption for Image Processing Algorithm in Matlab and OpenCV.» *IEEE*, 2012: 75-78.

Spectrum, IEEE. *Top 10 Robotic Kinect Hacks.* http://spectrum.ieee.org/automaton/robotics/diy/top-10-robotic-kinect-hacks.

Triggs, Navneet Dalal and Bill. «Histograms of Oriented Gradients for Human Detection.» *INRIA Rhone-Alps*, 2005.

ANEXO 1 Instalación de las dependencias para la ejecución del software desarrollado

Ya que el software desarrollado utiliza distintas librerías, en este anexo se incluye un pequeño manual de instalación y compilación de las librerías necesarias para el correcto funcionamiento de la plataforma.

Las grandes dependencias del software desarrollado son:

- **Python 2.7.6**: es la versión por defecto instalada en Ubuntu 14.04, aún así es necesario instalar algunas librerías para Python:
  - Numpy: librería para computación científica. Su instalación desde terminal:
    - sudo apt-get install python-numpy
- **OpenCV 2.4.10**: librería de visión por computador, es necesario compilar la librería para su uso en el proyecto:
  - Descargamos el código fuente desde: http://opencv.org/downloads.html
  - Descomprimimos en la carpeta que queramos
  - Instalamos librerías necesarias para su compilación desde el terminal:
    - sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev libjasper-dev libopenexr-dev cmake python-dev python-numpy python-tk libtbb-dev libeigen3-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev libqt4-opengl-dev sphinx-common texlive-latex-extra libv4l-dev libdc1394-22-dev libavcodec-dev libavformat-dev libswscale-dev default-jdk ant libvtk5-qt4-dev
  - Creamos una carpeta donde generaremos el ejecutable y configuramos el proyecto con Cmake

```
mkdir build
cd build
cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D
WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D
INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D
WITH_OPENGL=ON -D WITH_VTK=ON ..
```

  - Compilamos la librería

```
make
sudo make install
```

  - Editamos el archivo opencv.conf añadiendo /usr/local/lib:

```
sudo gedit /etc/ld.so.conf.d/opencv.conf
```

- Configuramos la librería

```
sudo ldconfig
```

- Añadimos las variables de entorno necesarias

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

- Reiniciamos el equipo

- **CVideo :** es un componente para decodificar los frames H.264
  - Descargamos el proyecto Heidi
    - https://github.com/robotika/heidi
    - Entramos en la carpeta cvideo del proyecto
    - Ejecutamos en la línea de comandos:
    - sudo python setup.py build
    - sudo python setup.py install
- Ffmpeg: librería multimedia que utiliza el componente cvideo, las librerías de ffmpeg ya están instaladas en la instalación de OpenCV