uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática
2019-2020

Trabajo Fin de Grado

Gestión Autónoma de Misiones en UAV Basada en Visión Artificial

---

Álvaro Bustillo de La Cruz

Tutor

Jesús García Herrero

Colmenarejo, 2020

AGRADECIMIENTOS

Quisiera agradecer a todas aquellas personas que han contribuido de una forma u otra y me han apoyado en la realización de este trabajo. Gracias a su desarrollo he podido investigar mucho más sobre dos tecnologías que me encantan, los drones y la inteligencia artificial.

Primero me gustaría agradecer a mis familiares y mi novia, los cuales me han apoyado incondicionalmente durante el transcurso de mi carrera.

También quisiera agradecer a Jesús García Herrero por toda la confianza puesta en mi desde hace tantos años, y por apoyarme en cada decisión que tomase, sin su ayuda y su pasión, este proyecto no hubiera llegado tan lejos.

Por último, agradecer a todos mis compañeros de universidad y a todo el personal de laboratorio que me ha ayudado durante estos últimos años y han contribuido en este proyecto prestándome un momento de su tiempo para realizar las pruebas y la recogida de datos.

RESUMEN

En el presente documento se tratan los conceptos teórico-prácticos para el diseño de un sistema de visión artificial capaz de detectar personas por medio de un UAV y tomar las acciones pertinentes en cada caso.

El sistema se basa en la tecnología de vuelo Pixhawk, sobre la que se embarcará la controladora de vuelo Pixhawk, con el software de vuelo PX4. Además, se creará una misión preprogramada y elaborada con el software QGround, del proyecto Pixhawk.

Durante la misión, el sistema de visión artificial para la detección de personas se ejecutará en una Raspberry Pi (ordenador de pequeñas dimensiones) y se embarcará al chasis del UAV. Además, se implementará un algoritmo con una lógica de vuelo que permita detectar elementos de interés que se encuentren dentro del ángulo de visión del UAV. Cuando un cuerpo humano se encuentre en el ángulo de visión del UAV, se deberá enviar una interrupción a la controladora de vuelo para aplicar las acciones pertinentes. Para comunicar la controladora de vuelo con los algoritmos de visión artificial y la lógica de vuelo, se implementará la comunicación entre los ordenadores Pixhawk y Raspberry Pi.

En conclusión, el UAV será capaz de suspender la misión actual a partir de un evento de interés, después realizará acciones sobre la controladora de vuelo y por último será capaz de recuperar el estado de la misión inicial una vez terminada la interrupción por el elemento de interés.

Palabras clave:

UAV, visión artificial, Raspberry Pi, controladora de vuelo, misión

ABSTRACT

This paper covers all theoretical and practical concepts about the design of an artificial vision system. The main reason of this artificial vision system is to detect people by a UAV and taking the pertinent actions in each case.

The system is based on Pixhawk flight technology, with the onboard computer Pixhawk and the PX4 flight software controller. Also, the software to launch pre planned mission has been QGround, which is part of the Project Pixhawk.

During the pre-planned mission, the artificial vision system will be executed over the component Raspberry Pi. This component is a small external onboard computer to compute all the data collected while the mission lasts. Besides, inside this computer an algorithm will be implemented by following a flight logic that allows detecting people who are within the UAV´s angle of vision. And due to this detection, the flight logic will determine which action the flight controller should take. To achieve this system works, will be necessary to implement the communication between the Raspberry Pi, which contains all the flight logic system and the artificial vision system, and the Pixhawk, which contains the PX4 flight software controller. This communication will give the Raspberry Pi the chance to interrupt the flight if needed.

To sum up, the UAV will be able to suspend the current mission due to an interest event, execute an action over the flight controller and then recover the status of the initial mission once the interruption is over.

Keywords:

UAV, artificial vision, Raspberry Pi, flight controller, mission

ÍNDICE

INDICE DE ILUSTRACIONES

INDICE DE TABLAS

1. INTRODUCCIÓN

En este primer capítulo se introducirá el proyecto. Primero se debe analizar la motivación que ha impulsado el desarrollo de este proyecto. También se mostrará cómo interviene la legislación en el desarrollo práctico. Además, se mostrarán los principales objetivos que se han alcanzado durante su desarrollo. Y por último se explicará la estructura del presente documento.

1.1 Motivación

Hoy en día, el ámbito de los vehículos aéreos no tripulados está poniéndose de moda, una de las principales causas es por la popularidad de drones profesionales en la fotografía, también por la ludificación de drones recreativos y por la inclusión de esta tecnología en el transporte de mercancías.

Hacer despegar esta nueva tecnología está siendo muy agradecido por muchos sectores profesionales y es un gran honor intentar facilitar el día a día de muchas personas.

El foco de este proyecto se ha centrado en esta parte más profesional y en cómo desarrollar un producto con una funcionalidad capaz de facilitar y mejorar la tarea de implementar planes de vuelo con una capacidad de reconocimiento de elementos pertinentes.

Actualmente la automatización de procesos con drones se utiliza para entrega de paquetes, tareas de vigilancia, salvamento, ayuda al sector de la agricultura y ganadería y muchas más utilidades. Es por ello por lo que tener un drone capaz de adaptarse a muchas de estas circunstancias y además ser un producto robusto y con tolerancia a fallos es fundamental.

1.2 Marco Regulador

Actualmente existen varios aspectos limitantes y restrictivos a la hora de comenzar a utilizar estos productos. Esto se traduce en implica complicaciones a la hora de desarrollar servicios que involucren su uso.

El primer aspecto que conocer, con el que se encontraría cualquier persona que quiere comenzar a volar un drone es el de conocer su entorno, la finalidad del vuelo y como va a ser.

Un factor que juega a favor de los usuarios, cuyo uso es meramente recreativo o de investigación, es que ya no será necesario una licencia certificada y ser operador en la Agencia Estatal de Seguridad Aérea (AESA). Por el contrario, si el uso de estos vehículos va a ser profesional, el piloto deber cumplir una serie de requisitos como poseer el título, seguro de responsabilidad civil, certificado médico.

Además de estas restricciones, existen varios aspectos limitantes a la hora de realizar un vuelo como la distancia máxima permitida de vuelo, la cercanía a un aeropuerto y otros aspectos también muy relevantes a tener en cuenta.

AESA diferencia entre dos tipos de aeronaves, los que pesan menos de 250 gramos y los que pesan más.

La normativa que impone AESA para los menos pesados es: [1]

- Se debe volar el drone sin superar una altura máxima de 20 metros.
- Únicamente se restringe el vuelo en zonas naturales protegidas. También impide el vuelo dentro de un radio de 8km de un aeropuerto o aeródromo.
- Si se pretende realizar grabaciones con el drone, es importante no incumplir la Ley de Protección de Datos.

La normativa que impone AESA para los más pesados es:

- Es necesario tener identificado el drone con una placa que muestre el fabricante, el tipo de drone, el modelo, el número de serie y el nombre del piloto. También la emisora se deberá identificar con el nombre del propietario.
- Si se pretende volar sobre zonas urbanas o durante la noche en cualquier zona, el drone, no debe superar los 10 kg de peso.
- Para realizar vuelos más allá del alcance visual AESA impone que deben existir mecanismos para poder controlar el espacio aéreo en el que se encuentra el drone.
- Si se pretende realizar vuelos en zonas urbanas, se debe respetar la altura máxima de 120 metros (sobre el edificio más alto) y una distancia desde el piloto de 100 metros.

El drone que se utilizará para este proyecto se encuentra en el rango de 250 gr - 10kg , por ello deberá cumplir la normativa AESA para los drones de mayor peso. Para este proyecto, dado que se implementa un servicio pilotaje autónomo, se implementan sensores que permiten controlar el espacio aéreo en el que se encuentra el drone. Además, se incluyen mecanismos de seguridad para controlar aterrizajes de emergencia y posibles fallos en la controladora de vuelo.

La realización de las pruebas siempre se realizará de forma controlada en zonas habilitadas para el vuelo.

1.3 Objetivo

El objetivo de este proyecto es conseguir implementar en un drone, un sistema capaz de realizar misiones, embarcar una estación de cómputo y realizar todas las modificaciones necesarias para adaptarlo creando así un sistema de reconocimiento de objetos/personas

basadas en visión artificial. Y finalmente, unificar todos los sistemas en un sistema autónomo capaz de realizar y crear interrupciones a lo largo de un plan si las condiciones lo exigen.

Para el desarrollo de un sistema capaz de reproducir misiones previamente planeadas, la configuración de la misión se hará a través de una herramienta llamada QGround, la cual permite establecer todos los parámetros necesarios para realizar una misión (puntos de interés, velocidad, altitud, tiempo de espera entre puntos de interés …). La herramienta QGround se comunicará con el drone a través del puerto MavLink y el ordenador de abordo PX4.

El ordenador escogido para realizar el computo de la información, ha sido la Raspberry Pi(RPI) 3 B+. La cual para funcionar requiere de una alimentación externa.

El sistema de visión artificial deberá ser capaz de detectar objetos/personas a partir de una secuencia de imágenes recogidas por la RPI.

Y por último se quiere implementar un sistema capaz de suspender la misión inicial si se detectan eventos de interés, a partir de los cuales se tomarían las medidas oportunas y posteriormente se continuaría con la ejecución automática de la misión.

1.4 Estructura

El presente documento trata los aspectos teórico-prácticos relacionados con el diseño de un sistema de visión artificial aplicado a UAVs. Inicialmente se introducirán los conceptos teóricos sobre los que se va a tratar posteriormente en el documento. Después, se evaluará cada parte del diseño, se distinguen 3 partes principales.

La primera de ellas consiste en valorar los algoritmos de visión para ser capaz de detectar personas a partir de imágenes capturadas desde un UAV. Por ello es necesario evaluar y analizar las capacidades de este sistema de visión. El principal problema que se puede presentar son la altura desde la que se debe realizar las detecciones, por eso evaluará los falsos positivos y falsos negativos que obtiene el clasificador.

En el siguiente aspecto del proyecto se debe realizar una lógica de vuelo, a partir de la cual se implementarán posibles misiones adicionales que debe ejecutar el UAV cuando ocurran las interrupciones producidas por elementos de interés.

Y, por último, la interconexión entre dos ordenadores, la Raspberry Pi 3 que se encarga de sistema de visión artificial y el Pixhawk encargado de contener el software de control PX4. Se debe realizar una interconexión que permita la comunicación entre ambos. Esta comunicación permitirá al sistema de visión indicar al ordenador de abordo que instrucciones adicionales al plan de vuelo inicial debe seguir.

Para finalizar se evaluarán distintos casos de uso para determinar a partir de unas circunstancias como debería funcionar el sistema y como funciona actualmente.

2. ESTADO DEL ARTE

2.1 Drone

En el entorno de la aviación a un drone se le conoce como un vehículo aero no tripulado. Pero los drones no son tan actuales como se piensa. Los drones comenzaron a existir en la I Guerra Mundial (1914-1918), donde su principal finalidad era servir de entrenamiento para el ejército de Reino Unido. Visto el avance tecnológico y la fiabilidad de este tipo de "armas", se vio que se podría llegar a utilizar en tareas de espionaje y combate.

A principios del siglo XXI, los drones comenzaron a hacerse hueco en el mercado como vehículo profesional. Esto se tradujo en un reacondicionamiento del drone que empezó a ser un producto pequeño, con menor autonomía y con un fuselaje mucho más ligero.

Desde sus inicios hasta ahora, se pueden distinguir los drones por su utilidad:

- Militar
- Comercial
- Civil
- Aficionado

Actualmente existen varias formas de vehículos aéreos no tripulados: [2]

- Drones de ala fija:

  Este tipo de drones se caracteriza por poseer un gran perfil alar que provoca que el drone sea capaz de planear y utilizar las corrientes de aire para elevarse y descender. Posee una gran autonomía para realizar largos vuelos, esto se debe a que no emplea tanta energía como otros drones convencionales y por ello es muy útil en aplicaciones que requieran largos vuelos y con mucha precisión.

  A diferencia de los drones de ala rotatoria, no poseen la cualidad de despegar y aterrizar en vertical, por ello para su manejo se requiere de una pequeña pista de despegue/aterrizaje. Por consiguiente, este tipo de drones no cuenta con la capacidad de realizar vuelos estacionarios.

1.Drone de ala fija

- Drones de ala rotatoria:

  Los drones de ala rotatoria se caracterizan por disponer en su chasis de varios motores que crean una corriente de aire direccional hacia el suelo. De esta forma, a diferencia del dron de ala fija que planeaba, este drone crea las corrientes de aire para volar.

  Este tipo de vehículos aéreos presenta más utilidad en aplicaciones que requieran un vuelo estático. Además, tiene la cualidad de poder despegar y aterrizar de una manera mucho menos sofisticada.

  Existen una gran cantidad de drones de ala rotatoria, y se diferencian en el número de motores que utilizan.

  Para hacer las hélices rotar, se pueden emplear distintos tipos de motores. Dentro del mercado los dos tipos más conocidos son, motores con escobillas y sin escobillas(brushless). Las escobillas le sirven al motor para cambiar la polaridad de las bobinas del motor de forma mecánica. Los motores sin escobillas utilizan electrónica para ese cambio de polaridad del motor. Por lo tanto, los motores con escobillas desperdician una gran cantidad de calor que se traduce en una pérdida de eficiencia. Por otra parte, los motores con escobillas son más baratos de fabricar y su uso se orienta más a ámbitos recreativos.

2. Drone de ala fija (hexacóptero)

Existen distintos tipos de sensores que se pueden embarcar en un drone:

- Algunos de estos sensores sirven a la controladora de vuelo para fusionar los datos y conseguir un vuelo más estable. Algunos son, giroscopio, acelerómetro, barómetro, sonar, GPS, compas, etc.
- Otros sensores que pueden utilizarse de forma ajena al control del vuelo son , LiDAR, sensores de imagen, termográficos, sensores de presión estática,…

Aunque se les llame vehículos aéreos no tripulados, tenemos varias opciones a la hora de manejarlos:

- Las aeronaves no tripuladas que se manejan mediante control remoto desde una estación en tierra denominadas RPA.
- Pueden ser controlados por un software de automatización de misiones. El cual puede reproducir misiones a través de un conjunto de instrucciones previamente establecidas.

En el caso de este proyecto, el drone utilizado será un hexacóptero, con una disposición hexagonal y seis rotores en su chasis. Y es alimentado con baterías tipo lipo con 4 celdas de almacenamiento, una capacidad de 5000mAh y un valor de descarga máxima de 40C que establece la capacidad de descarga para alimentar los motores.

2.2 OpenCv

OpenCv, sus siglas significan open-Abierto, c-Computer (Ordenador), v-Vision(visión). Es una biblioteca open source gratuita para uso comercial y está enfocada a la visión artificial y procesamiento de imágenes. Apareció por primera vez el año 1999 por parte de Intel. [3]

La librería cuenta con una gran cantidad de algoritmos para el tratamiento de imágenes, en total 2500 algoritmos. Y cuentan con numerosas utilidades como, captura de imagen en tiempo real, tratamiento de archivos de video, modificación de las propiedades de las imágenes, detección de caras, objetos, acciones, y muchas más utilidades.

Inicialmente esta librería comenzó siendo un proyecto orientado para el lenguaje de programación C++. Con el paso del tiempo se actualizo la compatibilidad de estos lenguajes ofreciendo interfaces para C++, Python Java y Matlab. [4]

2.3 Raspberry Pi

Raspberry Pi(RPI) es un ordenador de tamaño reducido, orientado inicialmente a estudiantes e investigadores. Apareció por primera vez en el año 2012 ofreciendo su primer modelo Raspberry Pi Modelo B. A partir de ese momento han ido evolucionando sus versiones hasta tener una cantidad de 7 modelos en su gama Raspberry Pi.

En el año 2015 introdujeron en el mercado los modelos Raspberry Pi Zero. Cuya principal característica era su reducido tamaño con capacidad para introducirse en cualquier sistema. Hasta el momento han aparecido un total de 3 versiones.

Hoy en día, las Raspberry Pi, destacan en el uso de proyectos para I.O.T. y Visión Artificial. Esto se debe a su capacidad de embarcar en cualquier sistema sin muchos problemas de espacio.

En este proyecto se está usando la versión Raspberry Pi 3 modelo B+, cuenta con las siguientes especificaciones: [5]

- Chipset Broadcom BCM2837 a 1,2 GHz
- ARM Cortex-A53 de 64 bits y cuatro núcleos
- Memoria LPDDR2 de 1 GB
- 40 pines GPIO
- Dimensiones: 85 x 56 x 17 mm

Para alimentar la Raspberry Pi 3 modelo B requiere un conector micro USB de 2,5 A. En este proyecto se ha adaptado un regulador de tensión para reducir la tensión de 14.8 V a 5V de la batería lipo principal del drone.

Raspberry Pi es la mejor opción entre sus alternativas (OrangePi , RockPi, Nvidia Jetson Nano…) por ser una de las primeras "computadoras de mano" que salieron al mercado y eso se traduce en una extensa comunidad y una fiabilidad que no ofrece ninguna otra.

2.4 Pixhawk

Pixhawk es un proyecto de open hardware. Principalmente diseñado para uso académico, para hobby y para la comunidad industrial.

Pixhawk destaca por su gran compatibilidad con numerosos sensores. Esto lo hace tener un carácter diferenciador y ser la mejor alternativa para crear un sistema capaz de tener fiabilidad y robustez. Además, el proyecto Pixhawk destaca por tener más productos como QGround, y es con estos productos con los que genera una experiencia de usuario muchísimo mejor que la competencia. Gracias a esta fusión de servicios y productos, el Pixhawk es capaz de recibir actualizaciones constantemente y corregir cualquier error de hardware o software que pueda surgir.

En este proyecto en concreto se ha usado la versión 2.4.8 de Pixhawk que tiene las siguientes especificaciones:

- 32 bits STM32F427 Cortex M4 core con FPU

- 168 MHz/256 KB RAM/2 MB Flash
- Coprocesador de 32-bit STM32F103 a prueba de fallos

Además, cuenta con los siguientes sensores incorporados:

- ST Micro L3GD20 3-axis giroscopio de 16 bits
- ST Micro LSM303D 3-axis de 14 bits acelerómetro y magnetómetro
- Invensense MPU 6000 3-axis acelerómetro y giroscopio
- MEAS MS5611 barómetro

Pero también cuenta con la posibilidad de incluir sensores externos partir de interfaces de comunicación que ofrece el Pixhawk.

Pixhawk destaca ante sus alternativas (NAVIO2, Naza, ArduPilot) por ser uno de los ordenadores abordo con más tiempo en el mercado y ello conlleva una gran comunidad con muchísima documentación en internet, además de una gran compatibilidad con los componentes de vuelo.

2.5 MavLink

MavLink es un protocolo de comunicación. Principalmente se creó para comunicación con drones, aunque también puede ser utilizado en numerosos proyectos diferentes.

MavLink es un tipo de protocolo punto a punto, a partir del cual se puede enviar información al drone o recibir el estado de este mismo. Internamente maneja una serie de subprotocolos, como por ejemplo el protocolo de misión y parámetros. La información que envía y recibe debe tener el formato XML en el puerto de salida y, de entrada.

Para que la conexión sea posible, y la transmisión de paquetes sea efectiva, el sistema debe ser muy eficiente, y esto es porque cada paquete que envía MavLink cuenta con 8bytes. También incluye para volverlo un sistema robusto un sistema de perdida de paquetes, corrupción de paquetes y un sistema de autentificación.

Actualmente en este proyecto se utiliza este protocolo a través del lenguaje de programación Python, pero destaca por soportar numerosos lenguajes de programación como, C, c++11, Python (2.7+,3.3+) c#, Objective C, Java, JavaScript, TypeScript, Lua, Swift, Clojure, Go, Haskell. [6]

2.6 Python

Python surge a finales de los años 80, en el año 1989, por Guido van Rossum cuya principal intención fue dar continuidad a un proyecto en el que había participado con la finalidad de ofrecer un lenguaje de programación fácil de usar.

Python es un lenguaje de programación multiparadigma, el cual soporta orientación a objetos, programación imperativa. Una característica muy representativa de este lenguaje es su capacidad de definir sus variables y sus estructuras en tiempo de ejecución. Gracias a esto no es necesario definir los tipos de datos y todas las estructuras antes de utilizarlas como en otros lenguajes, esto genera una experiencia de programación bastante más rápida que sus competidores.

En la actualidad, los principales usos de Python son:

- Inteligencia artificial
- Big Data
- Data Science
- Framework de Pruebas
- Desarrollo Web

A día de hoy se premia mucho los lenguajes de programación rápidos y con una curva de aprendizaje muy alta. En este punto es donde Python destaca antes sus alternativas por su simplicidad, su extensa biblioteca, su rápido desarrollo y su tipado dinámico.

2.7 QGround

QGround forma parte del proyecto Pixhawk. Es un software dedicado al control y diseño del vuelo. A día de hoy es una de las herramientas más utilizadas en este ámbito debido al gran poderío de los servicios Pixhawk y a sus productos.

QGround ofrece una amplia parametrización del Pixhawk. Pudiendo así ofrecer una interfaz gráfica sencilla para la comunicación (vía MavLink) del drone con el piloto, sin necesidad de utilizar un lenguaje de programación. Además, ofrece la posibilidad de observar en tiempo real el estado del drone.

QGround tiene compatibilidad con la mayoría de los sistemas operativos del mercado:

- Windows
- OS X
- Linux
- iOS
- Android

Gracias a que QGround pertenece al proyecto Pixhawk, representa una de las herramientas más adecuadas y mejor optimizadas para el control y diseño de una misión. Existen otras alternativas como Mission-Planner y ARMPlanner que también son muy

buenas alternativas, pero presentan más incompatibilidades con el ordenador de abordo Pixhawk.

2.8 DroneKit

DroneKit es un proyecto de código libre dispuesto para todo el mundo. Su principal función es ofrecer una comunicación rápida y fiable entre la controladora de vuelo y aplicaciones. Gracias a su tecnología de baja latencia es posible construir un dispositivo UAV mucho más completo y con muchas más funcionalidades.

La librería DroneKit da soporte a vehículos que sean compatibles con el protocolo MavLink. Y por esto la librería aporta una mayor funcionalidad a cualquier tipo de vehículos (aéreo, terrestre, marítimo, etc.).

DroneKit es una librería para implementar en el lenguaje de programación Python. Gracias a esto, se tiene un gran abanico de posibles aplicaciones y crea un entorno mucho más fácil de utilizar y más versátil.

DroneKit se caracteriza por tener muchas cualidades. Se pueden destacar las siguientes más importantes:

- Es capaz de conectarse a uno o múltiples vehículos desde un simple script.
- Si se quiere tratar la telemetría del vehículo para posibles aplicaciones, es capaz de extraer esta información en tiempo real sin la necesidad de usar interfaces graficas que imposibilitan el tratamiento de los datos por algoritmos externos en tiempo real.
- Puede tomar el control del vehículo y conducirlo con el modo “GUIDED mode” a donde el usuario necesite.
- Es capaz de crear misiones, enviarlas al vehículo e inicializarlas.
- Tiene la capacidad de sobrescribir los ajustes del canal RC.

DroneKit es una de las opciones más consideradas a la hora de implementar software para el control de vehículos no tripulados. Esto se debe a su gran documentación y el soporte que ofrece la gran comunidad que lo utiliza. Gracias a ser un software libre, los desarrolladores han hecho posible que DroneKit sea compatible con la mayoría de las controladoras del mercado.

```
from dronekit import connect
# Connect to UDP endpoint.
vehicle = connect('127.0.0.1:14550', wait_ready=True)
# Use returned Vehicle object to query device state - e.g. to get the mode:
print("Mode: %s" % vehicle.mode.name)
```

3. DISEÑO DEL SISTEMA DE VISION ARTIFICIAL

La visión artificial se basa en poder transformar las imágenes que una persona podría captar con sus ojos, a datos que un ordenador sea capaz de interpretar. Para ellos se extrae de las imágenes la información relevante a partir de números y símbolos, con los cuales los ordenadores son capaces de percibir y comprender que es lo que se está viendo. [7]

Una vez se ha extraído los datos de las imágenes, en forma de números y símbolos, se busca que el ordenador sea capaz de identificar una situación concreta a partir de estos datos. Dicha situación concreta, es con la cual el usuario podrá elegir qué comportamiento debe de tomar el sistema. En el caso de este proyecto se buscará que el UAV sea capaz de identificar un conjunto de objetos, con los que posteriormente se utilizarán para planificar el movimiento del UAV.

A continuación, se va a explicar cómo se ha instalado este sistema de visión artificial en la Raspberry Pi. También será necesario determinar qué cámara se va a utilizar para posteriormente crear un conjunto de imágenes. Esta recopilación de imágenes servirá al algoritmo de visión artificial, "Haar Cascade", para generar un clasificador. Para ello se utilizarán las herramientas de OpenCV que facilitarán la creación de este sistema de visión artificial y permitirán crear un algoritmo más robusto y con mayor tolerancia a fallos. Y por último se analizarán los parámetros más adecuados de este algoritmo que permitirán reducir la posibilidad de fallo.

Para el desarrollo de este sistema de visión artificial embarcado en un UAV, tan solo se pretende implementar la identificación de personas. Se extraerá un fotograma de la cámara y se utilizará para identificar a las personas. Para hacer el sistema más robusto se utilizarán técnicas que ayudarán a mejorar la fiabilidad y precisión de las detecciones. Es por eso que únicamente será relevante tratar algoritmos de clasificación de objetos.

La clasificación de objetos consiste en identificar un elemento asociado a una clase. Cada clase posee unas características específicas que determinan en distintas situaciones cómo se reconoce ese objeto en la imagen. Para el caso de este proyecto, se determinarán dos clases, una persona con todos sus atributos (cuerpo, cabeza, brazos, piernas…) y lo que no es una persona (cualquier otra cosa). Como objetivo se busca clasificar los objetos que sean persona como clase_persona, y aquellos que no lo sean tratarlos como clase_no_persona.

3.1 Unidad de computación

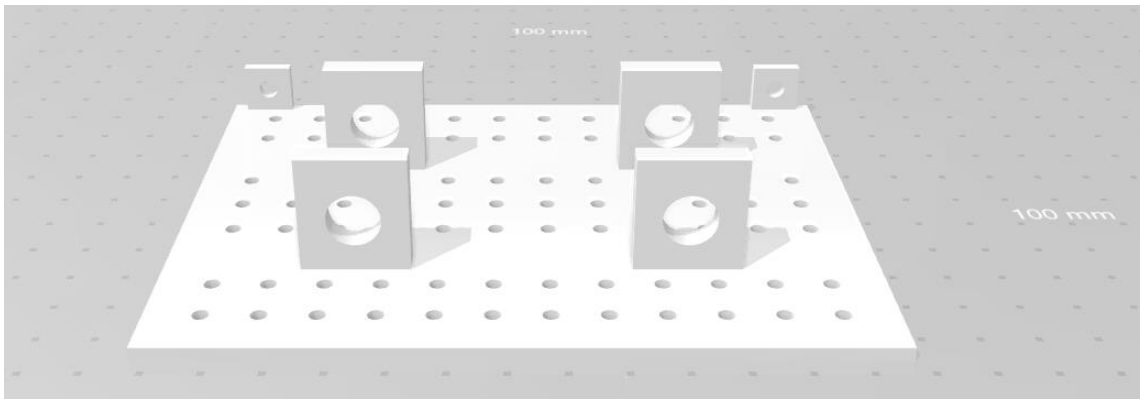
Para embarcar un sistema de visión artificial en un UAV, se deben tener en consideración los siguientes aspectos:

- Peso
- Potencia

- Autonomía
- Precio

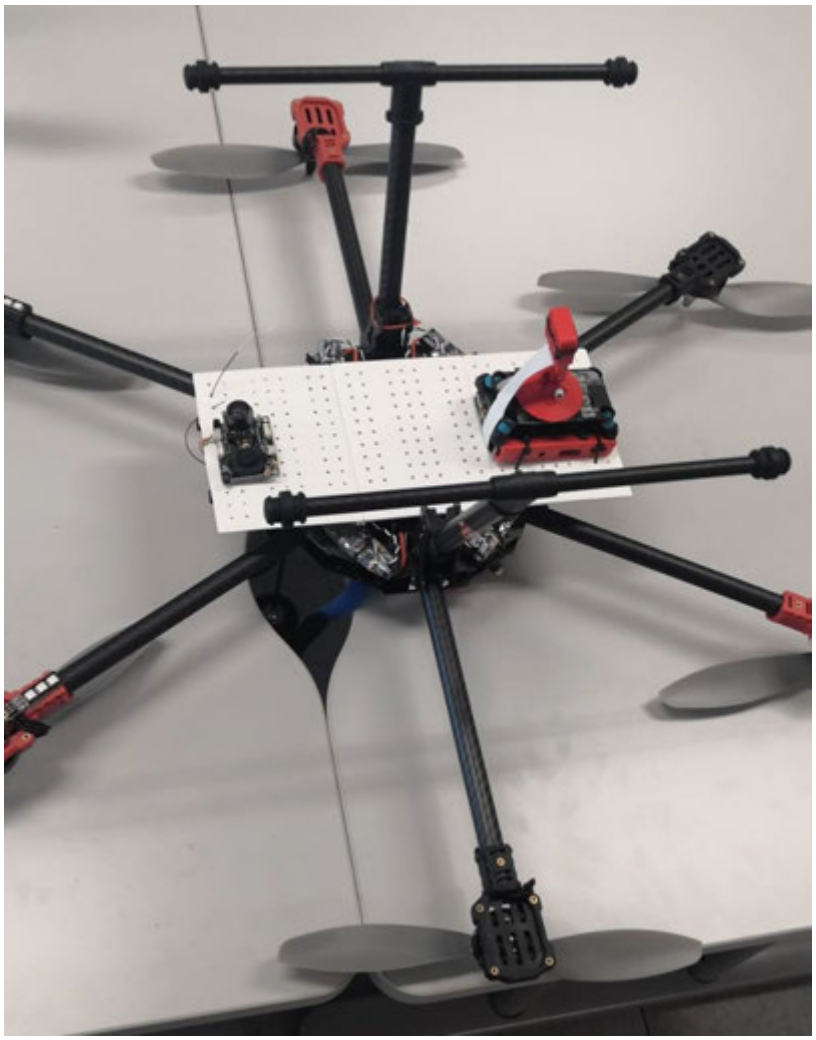
Uno de los ordenadores que pueden proporcionar todos estos requisitos es la Raspberry Pi. En concreto se va a utilizar la Raspberry Pi 3 B+ ya que hasta el momento de realización de este trabajo era el computador más adecuado (a lo largo del desarrollo la empresa Raspberry Pi, introdujo en el mercado su nuevo modelo mucho más potente Raspberry Pi 4, pero dado que el proyecto ya había empezado no se ha podido adquirir el producto).

La Raspberry Pi 3 B+, tan solo tiene una masa de 50 gramos, con la cual el UAV sobre el que se a embarcar este componente apenas se verá afectado. Aun así, se ha diseñado una plataforma para acoplar eficientemente la Raspberry Pi al chasis del drone.



3. Base para acoplar RPI a chasis del drone

4. Drone Vista superior



5. Drone vista inferior

Para utilizar este ordenador, se va a necesitar instalar un sistema operativo que sea compatible con los componentes de este ordenador. En este caso se ha optado por un sistema Raspbian, que ha sido diseñado 100% para este tipo de productos, por lo tanto, ofrece un alta robustez y fiabilidad en comparación con otros. Raspbian es una distribución del sistema operativo GNU/Linux que para su desarrollo fue basado en Debian. Su lanzamiento se produjo en junio de 2012.

Una vez que se ha instalado Raspbian en la Raspberry Pi 3 B+, se necesita una cámara a partir de la cual tomar imágenes de lo que se desee. Para ello se ha utilizado la Raspberry Pi Camera V2.

	Camera Module v2
Precio	$25
Peso	3g

Resolución	8 Megapixels
Modos de vídeo	1080p30, 720p60 y 640 × 480p60/90
Integración con Linux	V4L2 controlador disponible
C programming API	OpenMAX IL y otros disponibles
Sensor	Sony IMX219
Resolución del sensor	3280 × 2464 píxeles
Zona de imagen del sensor	3.68 x 2.76 mm (4.6 mm diagonal)
Tamaño de píxel	1.12 µm x 1.12 µm
Tamaño óptico	1/4"
Longitud focal	3.04 mm
Campo de visión horizontal	62.2 grados
Campo de visión vertical	48.8 grados
Radio Focal (F-Stop)	2.0

1.Especificaciones cámara v2 Raspberry Pi

Para poder acoplar la Raspberry Pi Camera v2 en la placa Raspberry Pi se introducirá el cable dentro del puerto especifico de la cámara.

6.Raspberry Pi con cámara conectada

Una vez haya sido instalada en la placa, quedará activar el puerto en la configuración de software de la Raspberry:

1. En la consola de comandos “sudo raspi-config”
2. Entrar en el menú “Interfacing Options” → “Camera”
3. Habilitar la cámara con la opción “Enable”

En el caso de tener errores a la hora de utilizar la cámara con OpenCV, se recomienda cargar el módulo de la cámara:

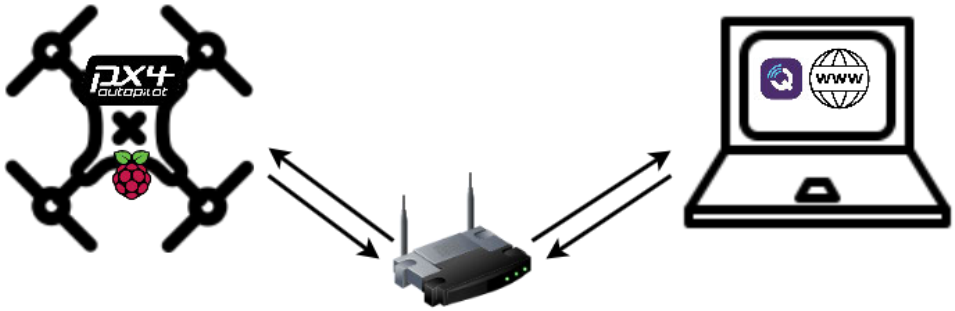
- Sudo modprobe bcm2835-v4l2

3.2 DataSet

Para obtener los datos necesarios que permitan la realización de este proyecto, se ha propuesto obtener los datos a partir de un streaming creado a través de la Raspberry Pi instalada en el UAV.

El proceso de comunicación con la Raspberry Pi es el siguiente:

En la imagen 7 se puede ver a la izquierda el UAV con los componentes Pixhawk y Raspberry Pi. A la derecha se encuentra la estación de tierra, encargada de reproducir las imágenes capturadas en el UAV y también contiene el software QGround para enviar las misiones correspondientes. En medio del diagrama se encuentra el enrutador, que comunica la estación de tierra con el drone, para este proyecto se ha optado por usar un móvil con la función “Zona wifi portátil”.



7.Diagrama de comunicación UAV-Estación tierra

Una funcionalidad extra que se introduce en este programa es permitir al piloto visualizar el streaming para ver lo que se está capturando.

De esta forma se debe paralelizar el uso de la cámara, porque de forma aislada no se puede stremear la imagen de la cámara y a la vez tomar fotografías/video. Por ello, se redirige la salida del streaming a una dirección web, a partir de la cual un proceso ejecutado paralelamente accede a esa dirección y captura la imagen stremeada. Por esta razón, la

resolución que se utilice para stremear será la resolución que se obtenga para el dataset. Al estar haciendo peticiones a un servidor externo, no se obtiene perdida de fotogramas ya que es muy improbable la pérdida de paquetes a través de una red local. El único error que se puede contemplar y que suponga una pérdida de frames es que la Raspberry sufra de hiperpaginación y como consecuencia no sea capaz de capturar los frames. Esto no se da con este hardware de Raspberry Pi 3 B+, pero en versiones inferiores con menores especificaciones no se asegura que sea el método más eficiente.

3.3 OpenCV

Una vez que el hardware está completamente montado y disponible para su uso, se debe instalar el software necesario para poder implementar visión artificial al sistema.

Para poder crear un sistema de visión artificial se necesitará instalar las librerías para Linux llamadas OpenCv (Open Computer Vision).

OpenCv está disponible en otros sistemas operativos y tiene compatibilidad con numerosos lenguajes de programación.

Para poder instalar OpenCv en la Raspberry Pi se deben seguir los siguientes pasos:

```
sudo apt -y update

sudo apt -y upgrade

git clone https://github.com/opencv/opencv.git

cd opencv

git checkout $cvVersion

cd ..

 git clone https://github.com/opencv/opencv_contrib.git

cd opencv_contrib

git checkout $cvVersion

cd ..

cd opencv

mkdir build

cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE \

            -D CMAKE_INSTALL_PREFIX=$cwd/installation/OpenCV-"$cvVersion" \

            -D INSTALL_C_EXAMPLES=ON \

            -D INSTALL_PYTHON_EXAMPLES=ON \
```

```
            -D WITH_TBB=ON \

            -D WITH_V4L=ON \

            -D OPENCV_PYTHON3_INSTALL_PATH=$cwd/OpenCV-$cvVersion-
py3/lib/python3.5/site-packages \

        -D WITH_QT=ON \

        -D WITH_OPENGL=ON \

        -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules \

        -D BUILD_EXAMPLES=ON ..

make -j$(nproc)

make install
```

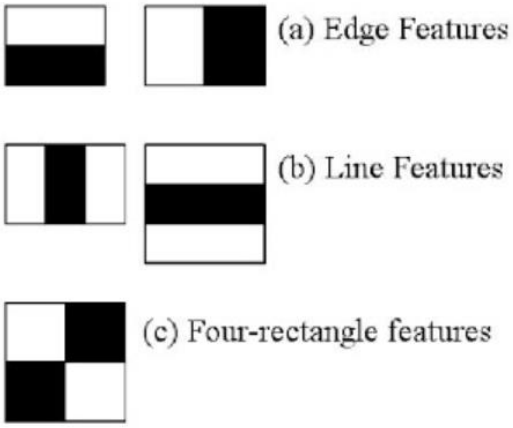
En el caso de este proyecto se ha decidido utilizar las librerías OpenCv con el lenguaje de programación Python. Esta decisión ha sido tomada por ser un lenguaje altamente compatible con las numerosas librerías necesarias para desarrollar un sistema de visión artificial, además de su cómodo desarrollo.

Una alternativa muy válida para el desarrollo de este sistema hubiera sido utilizar los lenguajes de programación C/C++, con los cuales se obtendría mucho más rendimiento en comparación con Python. En este caso al haber probado el sistema con Python y conocer las limitaciones que tiene, se sabe que es suficiente para este proyecto, pero en el caso de necesitar más velocidad de respuesta a la hora de identificar los elementos, se debería optar por estos otros lenguajes de programación.

3.4 Haar Cascade

Este metodo de clasificación en cascada fue porpuesto por P.Viola y M. Jones en 2001. [8] Este algoritmo genera un clasificador a partir de un proceso de entrenamiento. Para este proceso de entrenamiento, el algoritmo, emplea casos positivos y casos negativos, para aprender a discriminar los objetos de interés. Este problema es comunmente utilizado para la clasificacion de caras. En este proyecto se va a evaluar la precision y la utilidad que tienen los clasificadores Haar Cascade para la identificacion de cuerpos humanos. Esto manitene mas dificultades que una cara debido a que un cuerpo se define por muchas mas caracterisitcas(manos, piernas, tronco, cabeza…).

Para poder identificar como es el objeto que se quiere detectar, se debe extraer todas las características que lo identifiquen. Sobre una imagen en escala de grises, se deben realizar una suma de los píxeles que se encuentran en la parte negra menos los píxeles que se encuentran en la parte blanca, siguiendo estos filtros.



8.Filtros Haar Cascade

Para resolver este complejo problema de cálculo de todos los pixeles, se utilizó la solución de una imagen integral. Se usarán las siguientes secuencias recurrentes para calcular la suma de los pixeles.

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

s(x, y) representan la suma acumulada de la fila x con s(x -1) = 0 y ii(-1, y) = 0

El problema se encuentra en la tesitura de identificar qué partes de la imagen representan características relevantes y cuáles no. Por ello, se utiliza el algoritmo de aprendizaje AdaBoost para combinar varios clasificadores en cascada.

AdaBoost seleccionará unicamente aquellas características que mejoran el proceso de entrenamiento del clasificador, y para ello reducen considerablemente la gran cantidad de las que no son útiles y que representan ruido para el clasificador. Esto se traduce en un disminución exponencial del tiempo de ejecución del entrenamiento de un algoritmo Haar Cascade.

Gracias a este concepto, los creadores de este tipo de clasificador introdujeron al mundo del Machine Learning el concepto de clasificadores en cascada. En los que en el proceso de entrenamiento de clasificación de un objeto no se trataba de analizar el gran número de características que lo diferenciaban. Lo que buscaban era a partir de diferentes fases del entrenamiento identificar subconjuntos de características (como por ejemplo, dentro de una cara se podrían encontrar los ojos, la boca y otros rasgos faciales). Y con ello se reducía considerablemente el tiempo de identificación del objeto, ya que si las primeras fases de clasificación las primeras características fallaban. Es decir, si una cara no tiene ojos, no se puede considerar una cara. Esto crea un sistema de clasificación mucho mas rápido que otro convencional.

3.5 Proceso de entrenamiento del modelo de clasificación

El proceso de entrenamiento del modelo de clasificación sirve para ajustar el algoritmo para que "aprenda" a clasificar en imágenes las aéreas de interés. De esta forma el algoritmo será capaz de identificar en la imagen las personas y así poder realizar las acciones correspondientes.

Antes de comenzar con el entrenamiento se deben preparar todos las imágenes y datos que van a ser necesarios para que el entrenamiento sea efectivo. En esta ocasión se buscará que las imágenes que van a usarse en el entrenamiento y posteriormente en su uso, provengan del mismo sensor.

Un factor importante a tener en cuenta es valorar cuál es la mejor distribución de imágenes positivas e imágenes negativas. Para resolver este problema, se ha optado por obtener la mejor solución de forma empírica.

Existen múltiples formas de realizar el entrenamiento del modelo, para este caso se va a utilizar un conjunto de scripts que facilitan la tarea de utilizar una gran variedad de ejemplos positivos. En el caso de querer seguir las instrucciones oficiales de la librería OPENCV para realizar el entrenamiento [9], únicamente se podrá entrenar el modelo a partir de un solo ejemplo positivo, y sus derivadas transmutaciones (imágenes generadas a partir de la mezcla de una o varias imágenes positivas y una o varias imágenes negativas). Como para este caso se quiere añadir un dataset propio de ejemplos positivos y generar el mínimo de transmutaciones, se ha optado por una versión más adaptada a estas necesidades.

Es importante destacar que todo el dataset debe estar en un mismo formato, en este caso se ha optado por el formato de imagen ".png". Y el tamaño para el dataset de imágenes positivas es 50x50 (manteniendo es aspecto original de la foto,85x50) y para las imágenes negativas es 50x50 (manteniendo es aspecto original de la foto, 85x50). El motivo de elegir este tamaño se debe a que cuanto más grande sean los resultados más precisión en la evaluación se obtiene. La forma de análisis para obtener una vista parcial del coste que podría suponer el entrenamiento con distintos tamaños de imagen se tiene que para una imagen de 25x25 pueden existir un total de 160.000 características distintas a detectar, y teniendo en cuenta que esto aumenta exponencialmente a medida que aumentamos el tamaño de la imagen, resulta inviable utilizar imágenes de un tamaño superior. Además, para el entrenamiento de este modelo el ordenador ha consumido el 100% de los recursos durante un total de 18 horas. Esto suponiendo que se debe realizar más de un entrenamiento probando distintas configuraciones de parámetros siendo imposible utilizar imágenes de un tamaño superior.

Para el entrenamiento del modelo de clasificación en cascada se deben seguir los siguientes pasos:

1. El primer paso es tener las dependencias instaladas, para este ejemplo se van a utilizar una versión concreta de OpenCV para hacer funcionar los scripts proporcionado por el repositorio de GitHub.

```
Brew tap homebrew/science
brew install --with-tbb opencv
wget http://downloads.sourceforge.net/project/opencvlibrary/opencv-
unix/2.4.9/opencv-2.4.9.zip
unzip opencv-2.4.9.zip
```

2. El siguiente paso es descargar el repositorio que contiene la estructura para realizar y los scipts para realizar el entrenamiento.

```
git clone https://github.com/mrnugget/opencv-haar-classifier-training
```

3. A continuación, se deben colocar las imágenes positivas en el directorio llamado "positive_images" y posteriormente se deben listar en un fichero de texto todas las fotografías añadidas llamado "positives.txt"

```
find ./positive_images -iname "*.png" > positives.txt
```

4. De la misma forma que con los ejemplos positivos, se deben introducir todas las imágenes negativas en el directorio "negative_images" y listarlas en un fichero de texto llamado "negatives.txt"

```
find ./negative_images -iname "*. png " > negatives.txt
```

5. Una vez estén preparadas todas las imágenes positivas y negativas, se debe comenzar con el proceso de crear el dataset de estas imágenes que van a ser utilizadas para entrenar. Y finalmente se realizarán transmutaciones de las imágenes hasta conseguir el número total de ejemplos positivos necesarios para el entrenamiento.

```
perl bin/createsamples.pl positives.txt negatives.txt samples 1500\
  "opencv_createsamples -bgcolor 0 -bgthresh 0 -maxxangle 1.1\
  -maxyangle 1.1 maxzangle 0.5 -maxidev 40 -w 80 -h 40"
```

Parámetros:

- Simples → indica el número de imágenes positivas que se quiere para realizar el entrenamiento
- Bgcolor → determina la transparencia del color de fondo
- Bgthres → determina el límite de color de fondo

- Maxxange, maxyangle, maxzangle→ es el ángulo máximo que se van a rotar las imágenes positivas en los ejemplos negativos.
- Maxidev→Desviación de la intensidad máxima de los pixeles
- W y h → representan el tamaño final de las imágenes creadas.

6. Se deben unir todas las imágenes en un único fichero representado como un vector de las imágenes.

```
python ./tools/mergevec.py -v samples/ -o samples.vec
```

7. Y por último se debe empezar el entrenamiento [10]

```
opencv_traincascade -data classifier -vec samples.vec -bg negatives.txt\
        -numStages 20 -minHitRate 0.999 -maxFalseAlarmRate 0.5  numPos 1000\
        -numNeg 600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024\
        -precalcIdxBufSize 1024
```

- numStages, número de etapas
- minHitRate, tasa de éxito mínima en cada etapa.
- maxFalseAlarmRate, tasa máxima de falsas alarmas
- numPos, número de ejemplos positivos que van a usarse para cada etapa del entrenamiento.
- numNeg, número de ejemplos negativos que van a usarse para cada etapa del entrenamiento.
- precalcBufSize, tamaño del buffer para valores de característica que ya han sido calculadas
- precalcIdxBufSize, tamaño del buffer para índices de característica que ya han sido calculadas

Si se quisiera realizar una pequeña prueba que muestre la efectividad de las imágenes y no se quiera realizar un entrenamiento más rápido se deben usar las siguientes instrucciones:

```
opencv_traincascade -data classifier -vec samples.vec -bg negatives.txt\
        numStages 20 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -numPos 1000\
        numNeg 600 -w 80 -h 40 -mode ALL -precalcValBufSize 1024\
        precalcIdxBufSize 1024 -featureType LBP
```

El entrenamiento se ha realizado bajo una máquina virtual a las que se le han asignado las siguientes especificaciones:

- Procesador: Intel i7 7700HQ
- Hilos asignados: 6
- RAM: 16GB
- SO: Ubuntu x64
- Grafica GTX 1050

3.6 Ajuste de los parametros ROI

Una vez el modelo de clasificación en cascada haya sido creado, se debe continuar con su implementación en el algoritmo de visión para poder detectar cuerpos humanos.

Y es aquí donde se debe ajustar bien el funcionamiento del modelo, ya que, sin unos buenos parámetros (por mucho que el modelo contenga la información de las características propias de los cuerpos humanos), si el clasificador no determina bien cómo tiene que actuar el modelo y dónde, probablemente falle.

Y es por esto que se debe parametrizar correctamente la siguiente instrucción:

```
cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, f
lags[, minSize[, maxSize]]]]])
```

A continuación, se van a definir los conceptos teóricos que de cada parámetro de la anterior instrucción:

- Image, contiene el vector de tipo CV_8U de la imagen sobre la que se aplicará el modelo.
- scaleFactor, representa el reescalado que se realiza sobre la imagen, esto se traduce en una mayor posibilidad de detección del objeto. En el caso de que scaleFactor valga 1.03, se reduciría un 3% la imagen.

  Los valores permitidos van desde 1,0 hasta 1,99.

9.ScaleFactor en una imagen

- minNeighbors, indica cuántos vecinos debe contener cada bloque evaluado. Cuanto mayor sea el valor se producirán menos detecciones ya que obliga a encontrar más candidatos alrededor del bloque, por el contrario, un valor más bajo encontrará más detecciones.

Los valores permitidos van desde 1 hasta infinito, pero para acotar el problema a el caso real, se va a determinar que a partir de 7 no va a haber una solución factible. Por lo tanto, el rango será de 1 hasta 7.

- minSize, representa el tamaño mínimo que debe tener el objeto que se detecta.
- maxSize, representa el tamaño máximo que debe tener el objeto que se detecta.

Los valores más importantes que determinarán el éxito del modelo son scaleFactor y minNeighbors. Para encontrar los mejores parámetros se ha realizado de forma empírica un análisis con un conjunto de imágenes que previamente han sido clasificadas especificando en un fichero el número de detecciones y sus áreas. Se ha procedido a analizar todos los posibles valores de cada parámetro y comprobar la matriz de confusión que crea.

Los pasos para utilizar este script son los siguientes:

Selección Región de Interés en la imagen

Para comenzar se debe guardar en una estructura de datos todas las imágenes sobre las que se quiere aplicar una región de interés. Dichas imágenes deben estar contenidas en una carpeta del sistema de ficheros.

Para poder tratar los datos, se creará un fichero separados por comas (“.csv”), en el cual se representarán en distintas líneas los resultados con el siguiente formato:

- “RUTA_DEL_FICHERO/FICHERO”, PARAMETROS

En el caso de que una imagen contenga varias regiones de interés, se representaran en distintas líneas, pero con el mismo nombre y ruta del fichero.

Para el siguiente paso, se presupone que las imágenes tienen un tamaño y formato original no deseado. Por lo tanto, se debe realizar una modificación del formato y tamaño. Dicho formato debe ser el mismo para todas las imágenes y el tamaño resultante debe ser idéntico con el que se realizará luego en el análisis de los parámetros de clasificación. Eso se debe a que las coordenadas de las regiones de interés no pueden cambiar, ya que no serviría para nada este proceso y no se podría utilizar para identificar los falsos positivos y falsos negativos a partir de la posición de los elementos que se están identificando.

Una vez se han realizado las modificaciones previas a la selección, el script mostrará la pregunta en la consola de comandos: "Número Personas?", a partir de la cual se debe escribir el número de personas que se identifican en la foto.

Utilizando la herramienta SelectROI de la librería cv2, se mostrará la imagen con la posibilidad de trazar un recuadro sobre los elementos que se quieren seleccionar. Este método devuelve un vector de coordenadas del recuadro seleccionado.

10.Coordenadas detección persona

Es importante destacar que la representación de las coordenadas se ordena de mayor a menor de forma descendente en el eje de coordenadas, tal y como se representa en la imagen 10. Para el eje 'x' se ordenan los valores de mayor a menor en orden de izquierda a derecha.

El resultado del vector contendrá los siguientes parámetros representados en la imagen anterior:

Imagen [x y w h]

- x → Coordenada ‘x’ de la imagen correspondiente a la esquina superior izquierda del área seleccionada.
- y → Coordenada ‘y’ de la imagen correspondiente a la esquina superior izquierda del área seleccionada.
- w → Ancho del área seleccionada.
- h → Alto del área seleccionada.

De esta forma, es muy simple obtener las coordenadas más informativas del área seleccionado:

- Esquina superior derecha, coordenada X (x+w) y coordenada Y (y)
- Esquina inferior izquierda, coordenada X (x) y coordenada Y (y+h)
- Esquina inferior derecha, coordenada X (x+w) y coordenada Y (y+h)
- Centro, coordenada X (x+w/2) y coordenada Y (y+h/2)

La cuestión que toca decidir es qué coordenadas de la selección de la región de interés se deben guardar, para posteriormente poder identificar si se ha seleccionado correctamente. En el algoritmo de clasificación es muy raro que se identifique exactamente el mismo área que se ha seleccionado manualmente. Por lo tanto, hay que contemplar la mejor opción que valore el mínimo error posible y haga viable identificar si la ROI seleccionada por el clasificador hace referencia a la ROI seleccionada manualmente.

- Coordenadas de esquinas contrarias [superior-izquierda , inferior-derecha] o [superior-derecha , inferior-izquierda]

11.Selección manual de la región de interés

Resulta imposible determinar si la ROI del algoritmo de visión artificial hace referencia al mismo individuo que la ROI de la selección manual. Esto se debe a que las esquinas de la ROI no siempre van a localizarse en las mismas coordenadas.

- Coordenadas del centro de la imagen, de esta forma se puede comprobar si la región seleccionada por el algoritmo contiene a la región seleccionada de forma manual. Para ello se realizarán los siguientes cálculos que verifican si el punto intermedio de la selección manual está contenido en la selección de la región del algoritmo de visión artificial:

```
if x < i[0] and x+w > i[0] and y < i[1] and y+h > i[1]
```

Siendo ‘x’ e ‘y’ las coordenadas de la esquina izquierda superior de la región identificada por el algoritmo de visión artificial. Y siendo ‘h’ y ‘w’ la altura y el ancho de la región identificada por el algoritmo de visión artificial.

El vector i tiene un tamaño de 2. En la primera posición (i[0]) se almacenaría la coordenada ‘x’ del centro de la región identificada de forma manual y en la segunda posición (i[1]) la coordenada y del centro de la región.

Por ello siguiendo la lógica anterior descrita, siempre las coordenadas de la esquina izquierda-superior deben ser menores que las del centro de la región manual y el centro debe ser menor que la esquina derecha-superior. Lo mismo sucede con las coordenadas del eje y.

De esta forma el único error que puede cometer el algoritmo es sobrepasarse en el área seleccionada, y abarque otros puntos de interés. Pero este error será mitigado gracias a la configuración del clasificador determinando el tamaño máximo que puede tener un elemento.

12.Selección ampliada región de interés

Por último, se devolverá para escribir en el fichero de texto las coordenadas del centro de la región, siguiendo la siguiente estructura:

- *"RUTA_DEL_FICHERO/FICHERO",COORDENADA_X_DEL_CENTRO, COORDENADA_Y_DEL_CENTRO*

Analizar los parámetros que obtienen el mejor resultado:

Como se ha comentado anteriormente los parámetros más importantes para hacer que el algoritmo funcione correctamente son scaleFactor y minNeighbors.

Lo primero de todo será abrir la imagen que se quiere clasificar. Además, será necesario abrir el fichero de datos que contiene todas las coordenadas de las regiones identificadas manualmente.

A continuación, se debe realizar la detección sobre la imagen abierta y comprobar de forma escalonada los resultados que ofrece el modelo para los valores de scaleFactor de 1,0 hasta 1,99, con una precisión de 10^{-2}. Y lo mismo ocurre para minNeighbors con unos valores de 1 hasta 7, con una precisión de una unidad.

Si la región se ha detectado correctamente y existe en el fichero de datos, se incrementará a una estructura de datos una unidad para esos dos valores de scaleFactor y minNeighbors.

Una vez analizado todos los parámetros posibles sobre la misma imagen, se continuará analizando otras imágenes diferentes.

Este script no contempla la posibilidad de penalizar los parámetros en función de si no detectan la cantidad exacta de regiones de interés en la imagen. Podría ser una solución interesante, pero se ha decidido, tras analizar los resultados, que la finalidad del experimento no es obtener la cantidad exacta de regiones de interés en una imagen, sino obtener de forma más precisa la existencia de alguna región de interés. Por ello se debe generalizar el parámetro que obtiene más regiones de interés en la totalidad de las imágenes.

Aunque estos dos parámetros son importantes, no se deben olvidar otros parámetros de clasificación como el tamaño mínimo y máximo de la región identificada. Estos valores deberán ser parametrizados de forma experimental en función de la altura sobre la que se haya realizado el vuelo. De esta forma será más fácil determinar de forma aproximada el tamaño que puede tener una persona en la imagen.

Solución alternativa planteada

Otra alternativa que se ha planteado como posible solución es analizar la tonalidad de la imagen identificando los colores primarios. Para esta solución se debería plantear con un solo tipo de cuerpo y vestimenta con un color predominante. Y analizar si ese color se representa en los colores principales de la detección. Se plantea un pequeño problema con una rápida solución, y es que la detección de dicho color dependería de factores externos como la lente, la posición del sol respecto al cuerpo, la cantidad de luz que incide en el cuerpo. Para solucionar esto, se debería realizar en un entorno con unas condiciones ideales, y tratar la detección del color como un rango de valores RGB que cubra todo el espectro de posibles variaciones en la captura.

13.Histograma de colores RGB

3.7 Algoritmos de visión alternativos

Algunos de los algoritmos más útiles en el entorno de clasificación de elementos como cuerpos humanos se encuentran HOG con SVM y algoritmos de clasificación en cascada.

Dado que los principios de este trabajo buscan posteriormente crear un sistema mucho más complejo, tan solo se va a explicar y demostrar cómo funciona el sistema de clasificación con únicamente 1 clase objetivo. También hay que contemplar que cuanto mayor sea el número de clases a detectar más capacidad de computo es necesaria y por lo tanto se debería plantear una solución con un ordenador con mejores especificaciones que la Raspberry Pi y utilizar otros algoritmos de visión más complejos.

3.7.1 HOG y SVM

Parte HOG

En el clasificador HOG, Histogram of oriented gradients, se utiliza la distribución de las direcciones de los gradientes como características necesarias para identificar un objeto.

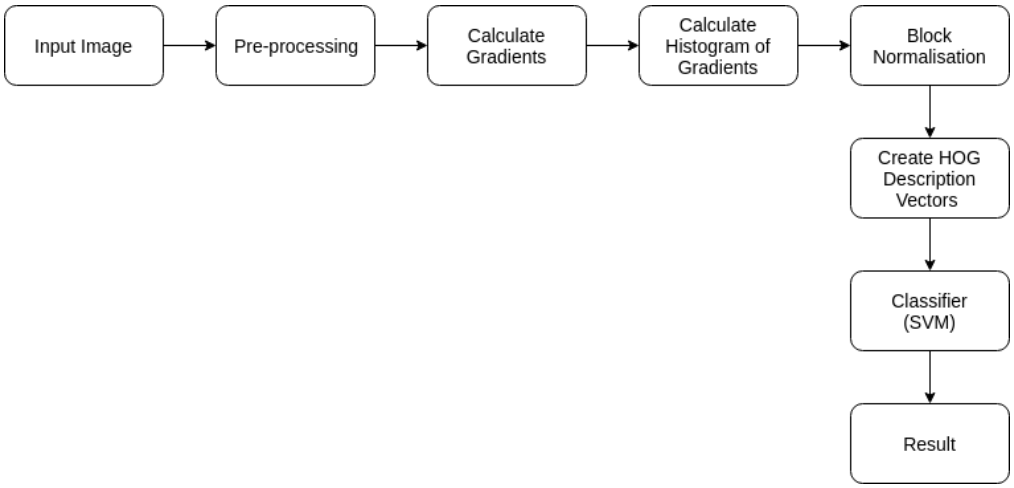
Antes de explicar teóricamente el histograma de gradientes orientados, se debe explicar qué es y cómo funciona el gradiente. El gradiente es un concepto muy utilizado en numerosos temas de Inteligencia artificial. Su definición aplicada al ámbito de la visión artificial es medir en un píxel concreto de la imagen y cuál es la razón que existe entre el cambio de color e intensidad en una dirección determinada.

“Matemáticamente, el gradiente de una función de dos variables (aquí la función de intensidad de la imagen) en cada punto de la imagen es un vector 2D con los componentes dados por las derivadas en las direcciones horizontal y vertical. En cada punto de la imagen, el vector de gradiente señala en la dirección del mayor incremento de intensidad posible, y la longitud del vector de gradiente corresponde a la tasa de cambio en esa dirección.” [11]

Una vez establecido el marco teórico del gradiente aplicado a conceptos de visión artificial, se debe definir en qué consiste la distribución direccional de los gradientes utilizados como características para un clasificador tipo HOG. Dichos gradientes son muy útiles en la identificación de regiones de interés como bordes o esquinas debido a que contienen mucha más información que otras partes planas de la imagen. El valor de la magnitud del gradiente en los bordes de la imagen normalmente presenta valores grandes y esto hace que la identificación con respecto a partes planas de la imagen sea más fácil.

Para calcular el histograma de gradientes orientados se deben seguir los siguientes pasos:

Se va a aplicar el cálculo de HOG sobre un sector de una imagen que contiene la información que se quiere detectar. [12]



14.Proceso del algoritmo HOG y SVM [13]

1 Preprocesado

El preprocesado consistirá en aplicar una reducción de tamaño sobre la imagen que se quiere tratar. La finalidad es crear imágenes menos pesadas y por tanto más rápidas a la hora de computar los gradientes.

$$V_{out} = AV_{in}^{\gamma}$$

- Vm → Vector de números reales no negativos

- Gamma → Valor gamma. Si se tiene un gamma<1 se le llama “gamma compression” y si gamma>1 se le llama gama expansión.
- A, es un valor constante
- Vout → Vector de salida

Después a aplicar una reducción de tamaño, se debe normalizar el gamma de la imagen.

La codificación gamma de imágenes se utiliza para optimizar el uso de bits, mejorar la transferencia de imágenes y aprovecha la manera no lineal en la que los humanos perciben la luz y el color. [14]

$$V_{out} = A \times V_{in}^2$$

2 Cálculo de los gradientes

Para el cálculo de gradientes, primero se necesitan calcular los horizontales y verticales. Una forma muy efectiva es utilizar los siguientes kernels para filtrar la imagen.

15.Kernels para cálculo de gradientes [15]

Teniendo la siguiente distribución del píxel Q, ha de calcularse la magnitud y dirección del gradiente:

	100	
70	Q	120
	50	

2.Cálculo gradientes en un pixel

Primero calcular el gradiente horizontal, que gracias a los kernels comentados anteriormente se consiguen aislar los valores horizontales:

$$G_x = 120 - 70 = 50$$

Y después se calcula el gradiente vertical:

$$G_y = 100 - 50 = 50$$

Una vez calculados el gradiente horizontal y el gradiente vertical, se debe cuantificar la magnitud del gradiente siguiendo esta fórmula:

$$g = \sqrt{g_x^2 + g_x^2}$$

$$G = \sqrt{(G_x)^2 + (G_y)^2} = 70.7$$

Una vez obtenido la magnitud solo queda saber la direccion del gradiente, para ello se utiliza la siguiente formula

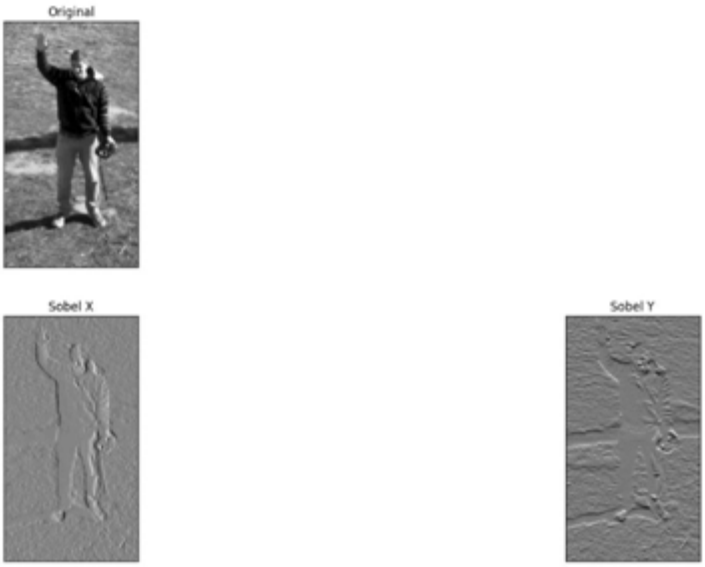
$$\theta = \arctan \frac{g_y}{g_x}$$

$$\theta = \arctan\left(\frac{g_y}{g_x}\right) = 45°$$

Como alternativa para calcular los gradientes positivos y negativos programando, se puede hacer uso de librerias como Sobel, que facilitan esta tarea de cálculo.

```
# Python gradient calculation
# Read image
Im = cv2.imread('test.png')
Im = np.float32(im) / 255.0
# Calculare gradient
Gx = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=1)
Gy = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=1)
```

A continuación, se puede ver el resultado de la transformación de la imagen original en el gradiente-x y gradiente-y. Y se denota una gran cantidad de información perdida. Pero esta información que ha eliminado la fotografía no se considera útil para identificar qué objeto se encuentra, aunque se pueda seguir reconodicendo que hay un cuerpo.



16.Sobel sobre imagen

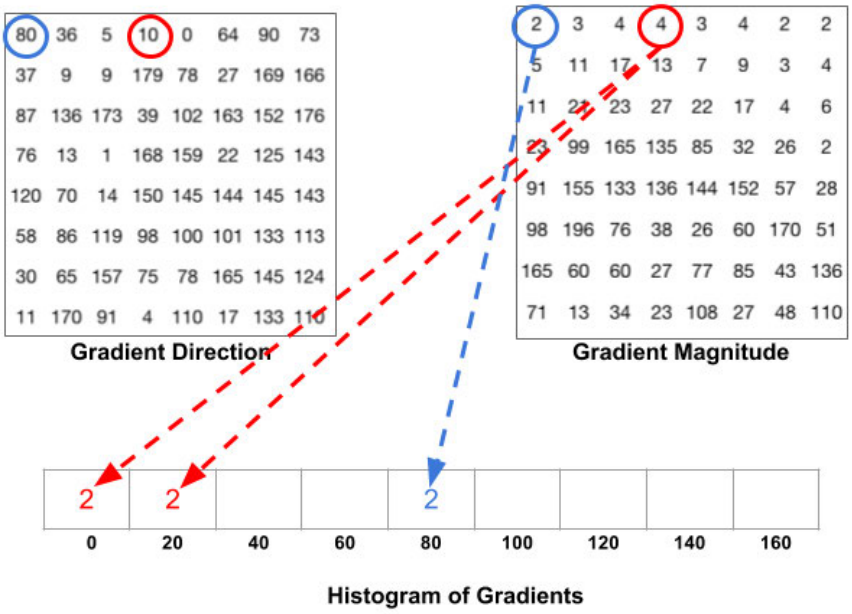
3 Cálculo del histograma del gradiente

Para el cálculo del histograma del gradiente, se debe dividir la imagen bloques proporcionales. Cada sub-bloque debe contener un bloque de 8x8 (la distribución 8x8 es meramente cuestión de diseño, puede tener cualquier otro valor proporcional, pero 8x8 es la medida más eficiente para resolver este problema).

Cada bloque de la división 8x8, conforma dos matrices de dimensión 8x8. Una matriz representa la magnitud del gradiente en cada sub-bloque. Y la otra matriz representa las direcciones del gradiente.

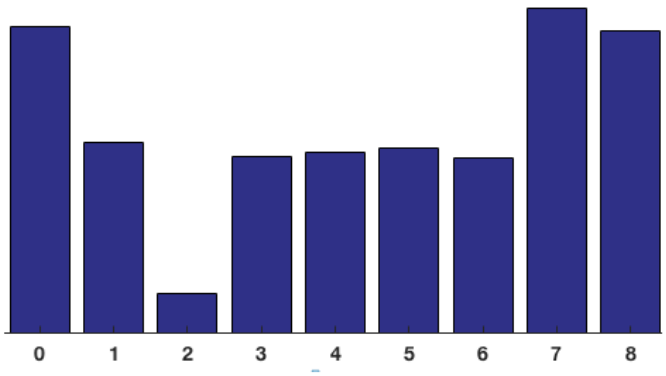
Es importante que las dimensiones del gradiente se representen en ángulos de 0 – 180. Esto tiene como finalidad obtener gradientes sin signo (dicho factor es otra cuestión de diseño que crea un sistema mucho más eficiente para resolver esta tarea).

Una vez realizados los cálculos se representan en un histograma de 9 columnas.



17.Cálculo histograma de gradientes

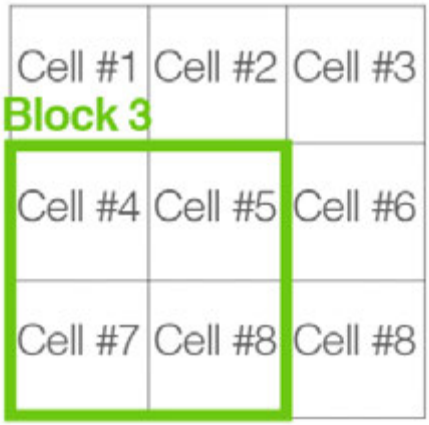
Para este proceso, se creará un histograma de los gradientes con 9 zonas ([0-19][20-39][40-59][60-79][80-99][100-119][120-139][140-159][160-179]).
Como se puede ver en el anterior esquema, se asociarán las dos matrices, escogiendo el valor de la matriz de direccion del gradiente y el valor corresponiente a la misma fila junto con la columna de la matriz de la magnitud del gradiente. El valor de la magnitud hace referencia al valor que se debe añadir a una posicion del histograma de los gradientes. Dicho valor de la direccion representa la posicion en la que se debe sumar ese valor de la magnitud. En caso de haber números intermedios se debe realizar una media entre los valores adyacentes y distribuir asi el valor total. Si el valor de la magnitud supera los 160º, se debe distribuir de forma proporcional entre la columna de 0º y la de 160º.



18.Histograma de gradientes

4 Normalización de bloque

Tras calcular el histograma de los gradientes, se debe tener en cuenta que el gradiente es sensible a cambios de luz. Para evitar este problema y obtener datos más reales, se debe realizar una normalización de bloques. Para dicha normalización se escogerán grupos de celdas de 2x2, 3x3, etc.



19.Normalización de bloque

Para la normalización se podrán utilizar algunas de las siguientes normas:

- L1
- L2
- L2-Hys

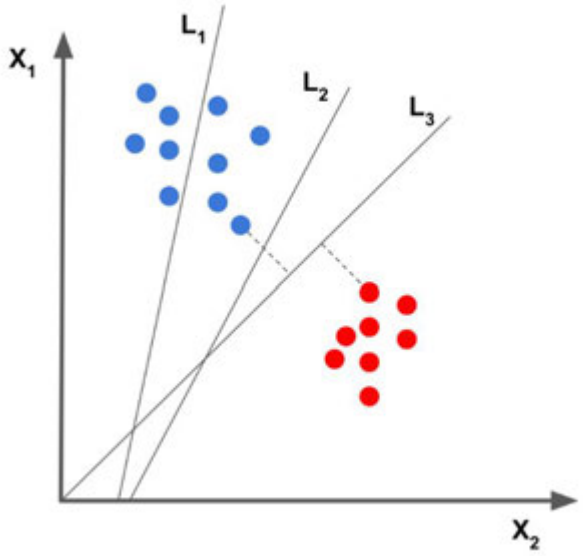
Las normas se aplicarán a los vectores RGB de los grupos de bloques creados anteriormente.

5 Cálculo del vector HOG

Tras calcular los valores de los gradientes normalizados del anterior vector, se debe realizar el mismo cálculo sobre todos los vectores y crear un gran vector con todos los resultados de la imagen.

Parte SVM

Para entender cómo funciona SVM (Support Vector Machines) de forma sencilla, hay que resolver el problema de clasificación como un problema binario. A partir de un problema binario de 2 clases, se quiere separar las clases.

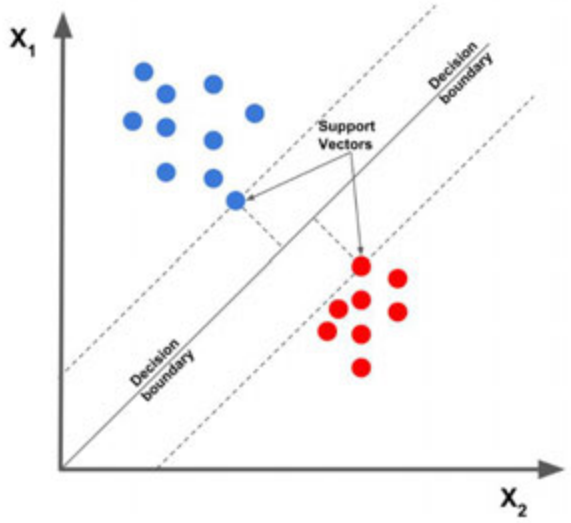


20.Problema de clasificación

Según esta gráfica se ha entrenado el sistema para tratar de separar estas clases, y como se puede ver L1 no obtiene un buen resultado, mientras que L2 y L3 dividen las clases de una manera más precisa. En el caso de introducir un nuevo caso, L3 representaría una solución más correcta, ya que obtiene una distancia entre clústeres más aproximada.

Al tratarse de una solución propuesta para un problema de dos dimensiones, se representa la línea límite de decisión como una recta. Pero si se tratase de un problema con 3 dimensiones es necesario usar un plano, y para sucesivas dimensiones un hiperplano.

Como se puede ver en la siguiente imagen, el problema consiste en encontrar la linea de decision que maximice el margen geométrico entre ambas clases.



21.Problema de clasificación con SVM

3.8 Sistema final

3.8.1 Parametrización de entrenamiento y proceso de validación

Para el entrenamiento del modelo, se aplicarán los siguientes parámetros sobre los que se ha obtenido un mejor rendimiento del clasificador:

- numStages, se ha establecido a 15, realizando pruebas con 10 y con 20. Con respecto a 20 etapas, no han existido diferencias apreciables y que argumentasen la diferencia de ejecución. A diferencia de que con 10 la obtención débilmente mejoró el rendimiento, pero la diferencia de tiempo entre un proceso y otro merece la pena.
- minHitRate, se ha establecido a 0.995 porque es el valor más eficiente que la mayoría de los usuarios e investigadores han reportado.
- maxFalseAlarmRate, sucede lo mismo que con el parámetro anterior, y el más recomendado es 0.5.
- numPos, se han introducido 1500 positivos, los cuales el 30% han sido imágenes capturadas con el mismo UAV a una misma altura y con la misma cámara que se utilizará posteriormente. El 70% restante, son fotografías generadas a partir de imágenes negativas, de las cuales se realizan transposiciones y rotaciones de las positivas dentro de las negativas.
- numNeg, se han introducido 2000 ejemplos negativos. Estos ejemplos han sido capturados únicamente de fotogramas extraídos de videos grabados con la cámara. Han sido capturadas sobre diferentes condiciones (diferentes alturas).

- precalcBufSize, 8196 MB, debido a las restricciones lógicas del computador sobre el que se ha realizado el entrenamiento.
- precalcIdxBufSize, 8196 MB, debido a las restricciones lógicas del computador sobre el que se ha realizado el entrenamiento.

Es importante realizar un tratamiento de los datos para que el modelo de clasificación generado tras el entrenamiento funcione de forma más eficiente y sobrecargue menos la unidad de procesado de la Raspberry Pi. Las técnicas utilizadas son las siguientes:

- Reajustar el tamaño de la imagen

Como ya se ha comentado anteriormente es esencial para la carga del procesador, reducir el número de pixeles sobre los que se va a aplicar el modelo de clasificación tipo "haar-cascade". De esta forma se reduce considerablemente el número de rectángulos sobre los que se tiene que calcular el sumatorio de los píxeles claros y los oscuros. A pesar de que se está intentando mitigar la carga del procesador en la tarea de clasificación aplicándole más carga al procesador con la tarea de reducción de tamaño, no es proporcional, y no afecta de forma significativa.

De esta forma se ha decidido reducir la imagen que se extrae del sensor. Pero una duda que surge podría ser, ¿Por qué no reducir el tamaño de la imagen que se obtiene directamente del sensor de la cámara? Pues bien, esta solución sería muy interesante, pero reduciría las posibilidades de utilizar esa imagen para otras aplicaciones, por ejemplo, poder retransmitir en vivo la imagen recogida por la cámara.

Existen distintos tipos de interpolación aplicables a imágenes:

22.Imagen sin interpolación [16]

Interpolación de vecino más cercano, solo tiene en cuenta el píxel más próximo y se calcula el valor del píxel promedio entre ambos. Es muy rápida, pero pierde mucha información. [17]

23.Interpolación de vecino más cercano [16]

Interpolación lineal, se toma una vecindad de 2x2 vecinos alrededor. Y con ello se calcula el valor el píxel promedio.

24.Interpolación lineal [16]

Interpolación cubica, aumenta el tamaño del vecindario a 4x4.

25.Interpolación cubica [16]

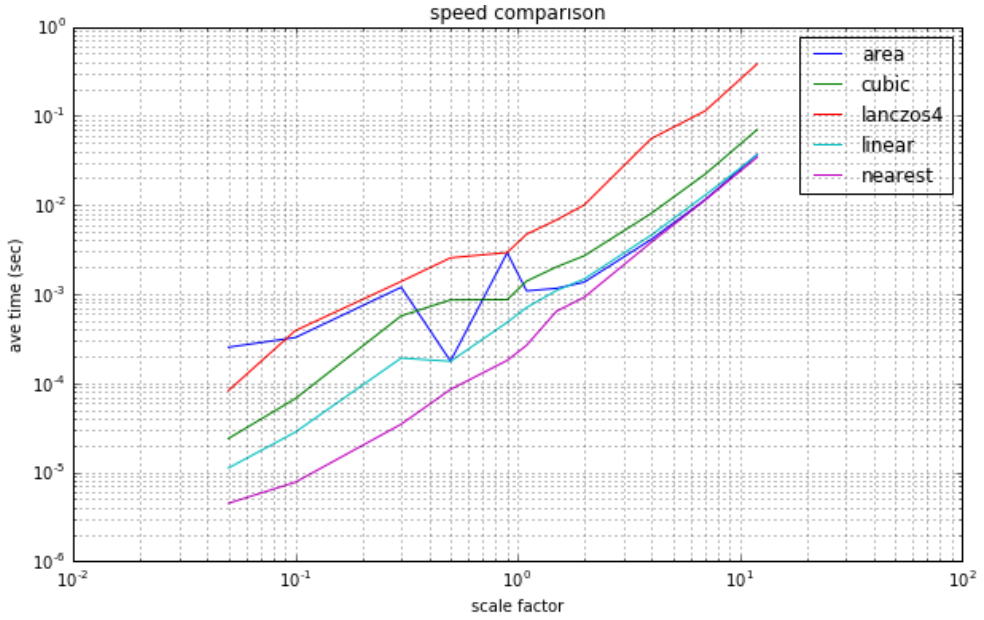
Interpolación LANCZOS4, utilizar un vecindario de 8x8

26.Interpolación LANCZOS4 [16]

Interpolación de área, se produce un re-muestreo utilizando la relación del área de los píxeles.

27.Interpolación de área [16]

A continuación, se va a realizar una comparativa de tiempos aplicando distintas interpolaciones a diferentes tamaños para analizar cuál es la más conveniente para este proyecto.



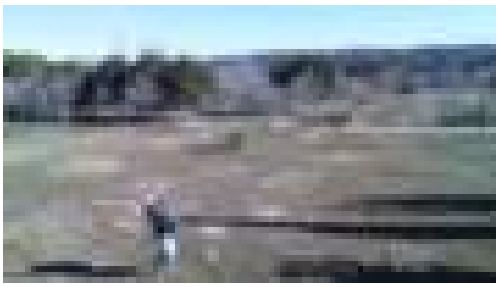
28.Comparativa tiempos con distintos métodos de interpolación [16]

Como se puede ver la mejor alternativa es la interpolación por área, lineal y por vecino más cercano. Se puede ver que la imagen que más información conserva es la interpolación por área. En el caso de querer utilizar la interpolación línea, si se cuenta con un número de canales en la imagen mayor que 2, al final el propio algoritmo realizaría las operaciones siguiendo la interpolación por área. Y, por lo tanto, dado que la primera transformación que se realiza en la imagen es sobre 3 canales RGB (rojo, verde y azul). Se ha llegado a la conclusión de que la opción más rápida y que obtiene mejores resultados es interpolación por área.

- Cambiar a escala de grises

Principalmente se quiere reducir a escala de grises una imagen capturada con los canales RGB ya que reduce computacionalmente el número de cálculos que el algoritmo de clasificación realiza. Esto se debe a que el modo RGB tiene 3 canales ($\mathbb{R}^{n \times m \times 3}$) y la escala de grises tan solo 1 canal ($\mathbb{R}^{n \times m}$). Además de esto, existen múltiples razones por las que realizar esta conversión. En la situación de utilizar un clasificador en cascada, se ha demostrado que la información relevante no es el color de la imagen. Y es por esto, que dejar el color en la fotografía no ayudaría a encontrar características del elemento que se quiere detectar y esto no sucedería ni para el entrenamiento ni para la verificación. Y dado que la finalidad del proyecto no es detectar tipos de personas basadas en su color o en la luminancia, los 3 canales RGB son irrelevantes.

De esta forma dados los siguientes ejemplos:



29. Imagen sin escala de grises

30.Imagen con escala de grises

Ambos ejemplos tienen el mismo tamaño, 89x50. La imagen “Imagen sin escala de grises”, en blanco y negro, tiene un peso de 3.93 Kb y la de la “Imagen con escala de grises “ tiene un peso de 9.48 Kb. Por lo tanto, supone más del doble de tamaño y por ello como es lógico se tiene que cargar más información en memoria a la hora de leer la imagen y calcular sus posibles características.

3.8.2 **Parametrización del modelo para la clasificación**

Tras realizar el análisis de cuáles son los parámetros que obtienen mejores resultados con el sistema de visión artificial, se ha llegado a la conclusión de que es importante identificar la altura al suelo sobre el vuelo que ha realizado el UAV, y calcular cuántos píxeles representarán la persona en la imagen que se entrega al clasificador.

A continuación, se va a analizar los valores de precisión de varias pruebas. Para ello se van a tener en cuenta los valores obtenidos en la matriz de confusión. Y después cabe estudiar la precisión que ofrece este modelo. Para analizar la precisión se va a utilizar la métrica de precisión total que combina las métricas de verdaderos positivos y falsos positivos.

La medida de precisión total se representa siguiendo la siguiente fórmula:

$$\text{Precisión total} = \frac{Vp + VN}{VP + VN + FP + FN}$$

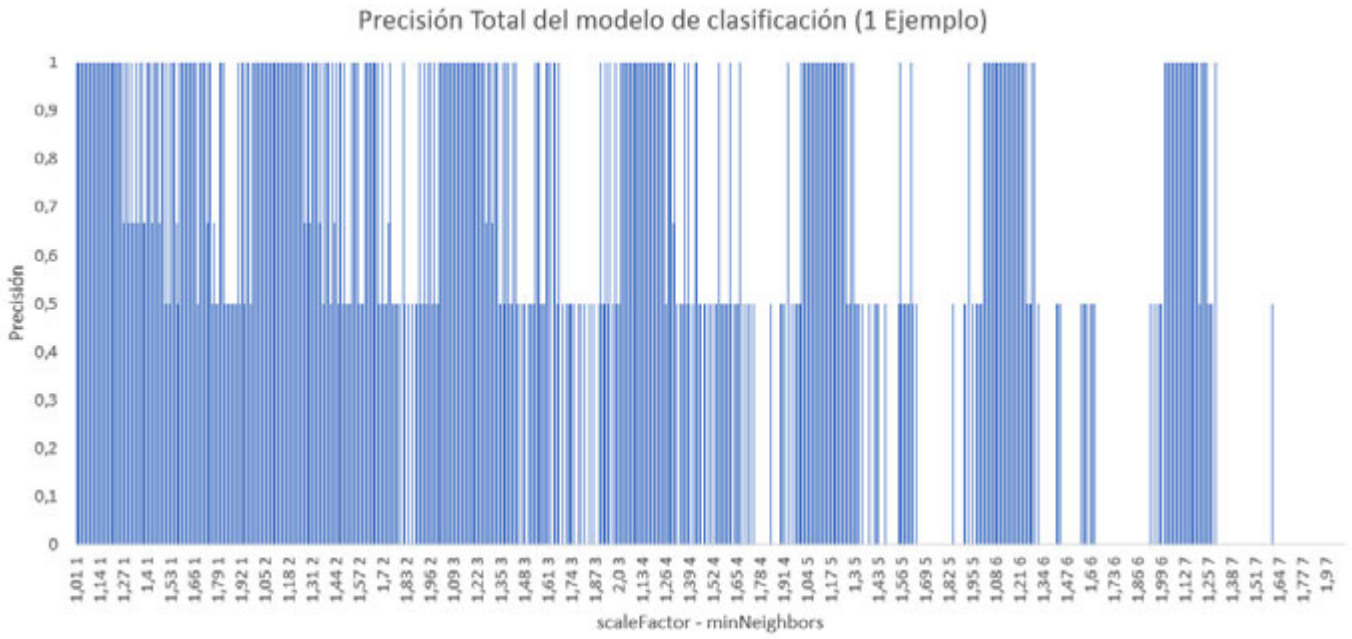
VP: Verdaderos Positivos, representa los elementos que se han clasificado como positivos y son positivos

FN: Falsos Negativos, representan las predicciones que se clasifican como negativas, pero son positivas, es decir objetos de interés que están en las imágenes, pero no se han detectado.

VN: Verdaderos negativos, representan las predicciones que se clasifican como negativas y realmente son negativas.

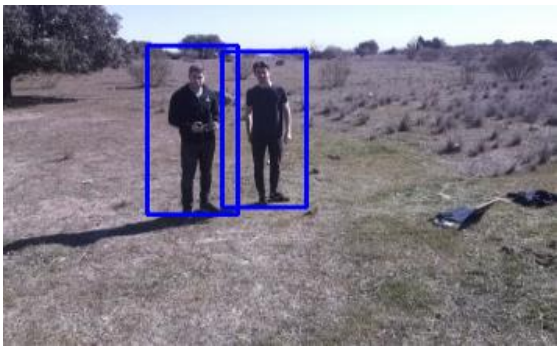
FP: Falsos positivos, representan las predicciones que se clasifican como positivas, pero son negativas.

En el siguiente gráfico se presenta la precisión sobre una sola imagen. La imagen se ha seleccionado al azar y en ella se encuentran dos personas. (Ejemplo scaleFactor 1.85 - minNeighbors 1). En el eje x se representa la precisión del modelo siguiendo la ecuación de Precisión total. Y en el eje x, se representan los valores de scaleFactor y minNeighbors con el formato siguiente: (Valor_scaleFactor Valor_minNeighbors)



31.Precisión Total del modelo de clasificación (1Ejemplo)

Como se puede ver para este caso concreto en el que se encuentran dos personas relativamente cerca, se obtiene un mejor resultado con los valores más pequeños de scaleFactor. Y el valor de minneighbors va disminuyendo su precisión total a medida que aumenta.

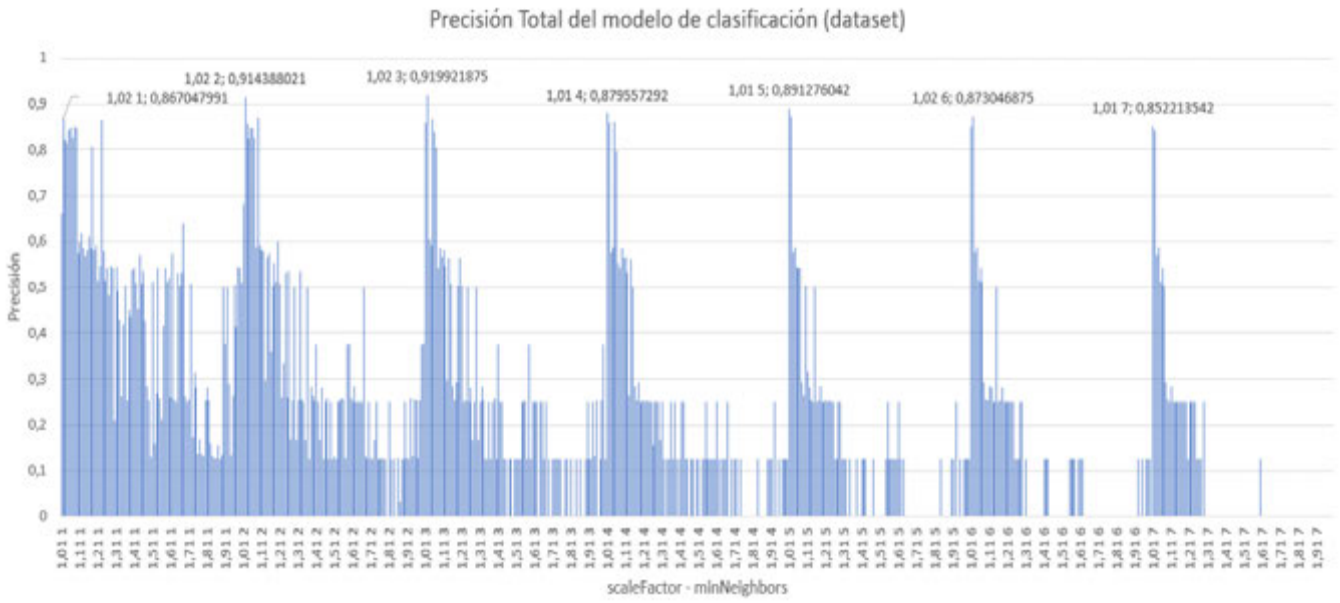


32.Ejemplo scaleFactor 1.85 - minNeighbors 1

33.Ejemplo scaleFactor 1.15 - minNeighbors 1

Aquí se puede observar dos ejemplos. “Ejemplo scaleFactor 1.85 - minNeighbors 1“ está utilizando los valores de 1.15 para scaleFactor y 1 para minNeighbors. Para la imagen “Ejemplo scaleFactor 1.15 - minNeighbors 1” se tienen los valores 1.85 para scaleFactor y 1 para minNeeighbors.

Estos valores, representan información muy precisa pero que solo lo aporta sobre un ejemplo concreto. Es necesario analizar la precisión del modelo sobre un dataset con un con una gran cantidad de imágenes.

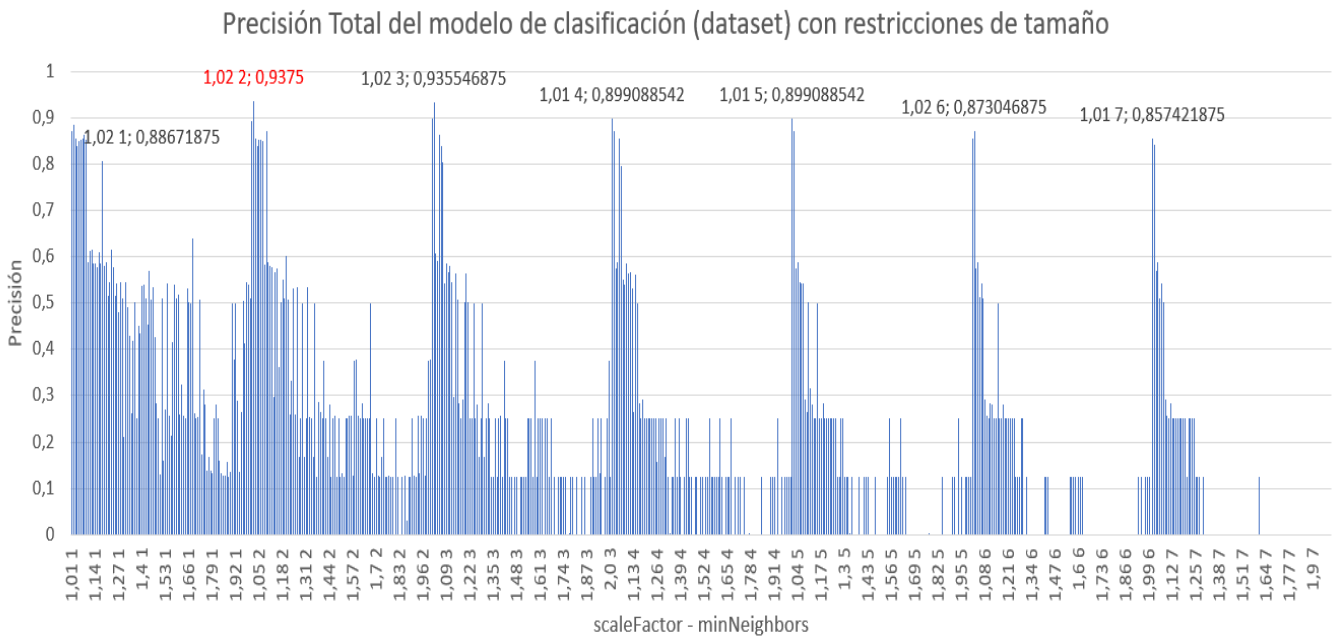


34.Precisión total del modelo de clasificación (dataset)

Como se puede ver en el anterior gráfico, los parámetros más repetidos de scaleFactor que obtienen entre un 85% y un 92% son 1.01 y 1.02. Los parámetros que más precisión obtienen con este dataset son: 1.12 para scaleFactor y 3 para minNeighbors.

En resumen, al aumentar el valor de minNeighbors se reduce la precisión total del modelo.

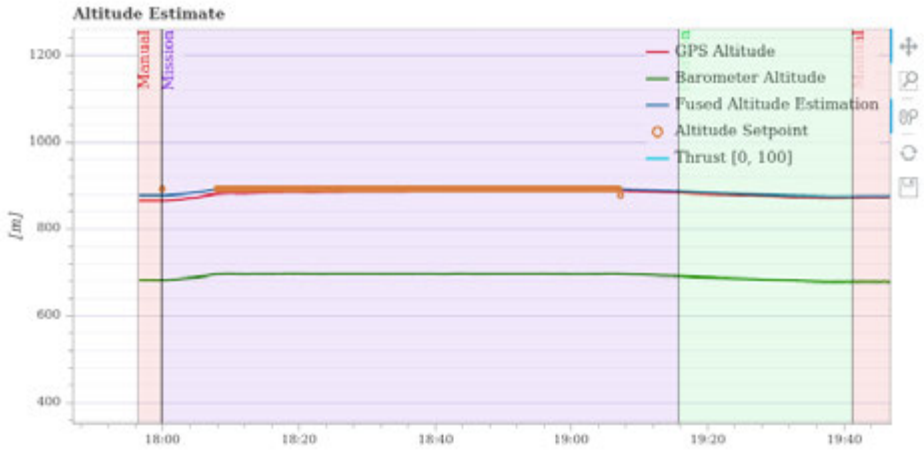
Para analizar estos parámetros no se han manejado restricciones de máxima o mínima detención. Si se introducen estas variables, se puede llegar a conseguir los siguientes resultados:



35.Precisión total del modelo de clasificación (dataset) con restricciones de tamaño

Como se puede ver, la precisión del modelo mejora su puntuación con respecto al modelo sin restricciones de tamaño. De esta forma se encuentran los valores con mejor resultado en los mismos parámetros (o muy próximos) que en la parametrización sin restricciones. El resultado de la precisión es de 93.75%, y los parámetros finales del modelo son 1.02 para scaleFactor y 2 para minNeighbors.

Es importante destacar que estos parámetros obtienen su máxima precisión para condiciones específicas con las que se han realizado las pruebas. Para ello la cámara ha necesitado una inclinación de un 45% sobre el eje horizontal. Además, se han realizado vuelos únicamente a 5 metros de altura, pero debido a las restricciones meteorológicas, se le debe añadir un error de +- 1 metro por pequeñas variaciones de la altura en el vuelo.



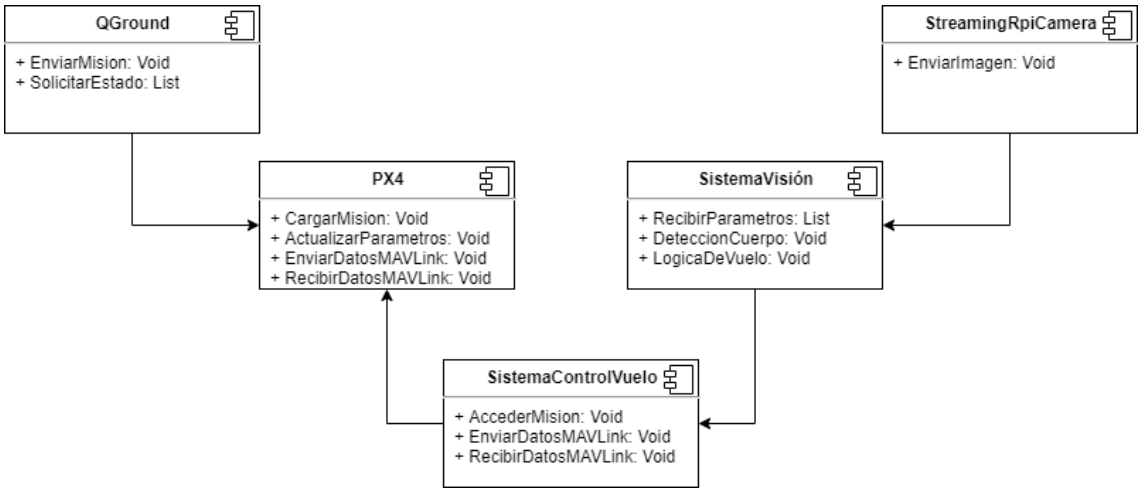
36.Altitud estimada

Se puede ver en la anterior gráfica, como el valor de la altitud recogida por el GPS, apenas tiene variación en el proceso de misión.

4. DISEÑO DEL SISTEMA DEL CONTROL DE UAV

Un sistema de control de vuelo para un UAV es aquel que otorga el control de vuelo a un piloto o en este caso a un software.

A través del siguiente diagrama de componentes se va a mostrar como interaccionan los distintos módulos de la gestión autónoma de misiones en UAVs basada en visión artificial:



37.Diagrama de componentes de la gestión autónoma

Lo primero que debe suceder es que haya una interrupción de un plan de vuelo establecido por el usuario, para ello el sistema de visión artificial debe detectar un elemento. Si se cumplen unas condiciones específicas, la Raspberry Pi llamará al planificador de los planes de vuelo de tal forma que las acciones le comunicarán a la controladora de vuelo las instrucciones necesarias que debe seguir para conducir el UAV al destino establecido. Una vez que la acción finalice, se determinará en función de la situación encontrada, si se debe continuar con el plan de vuelo interrumpido o aterrizar el UAV.

Cuando el sistema de visión artificial esté listo y la conexión entre los dos ordenadores embarcados en el drone estén preparados para comunicarse entre ellos, se debe empezar a diseñar el sistema de control del UAV.

Lo más importante para esta parte es considerar cual es la acción que debe tomar el sistema de control cuando el clasificador haya identificado un elemento. Para ello se va a discutir cuales son las alternativas adecuadas para este tipo de problema y que solución está dentro de las posibilidades de la controladora de vuelo Pixhawk. Una vez decididas las acciones que debe ejecutar el UAV, se diseñará una lógica de vuelo para ejecutar estas acciones.

Para ello, primero de todo se va a explicar cómo construir este sistema, y cómo instalar el software necesario para que la comunicación entre el sistema de control del UAV y el sistema de visión artificial puedan establecer una comunicación bidireccional.

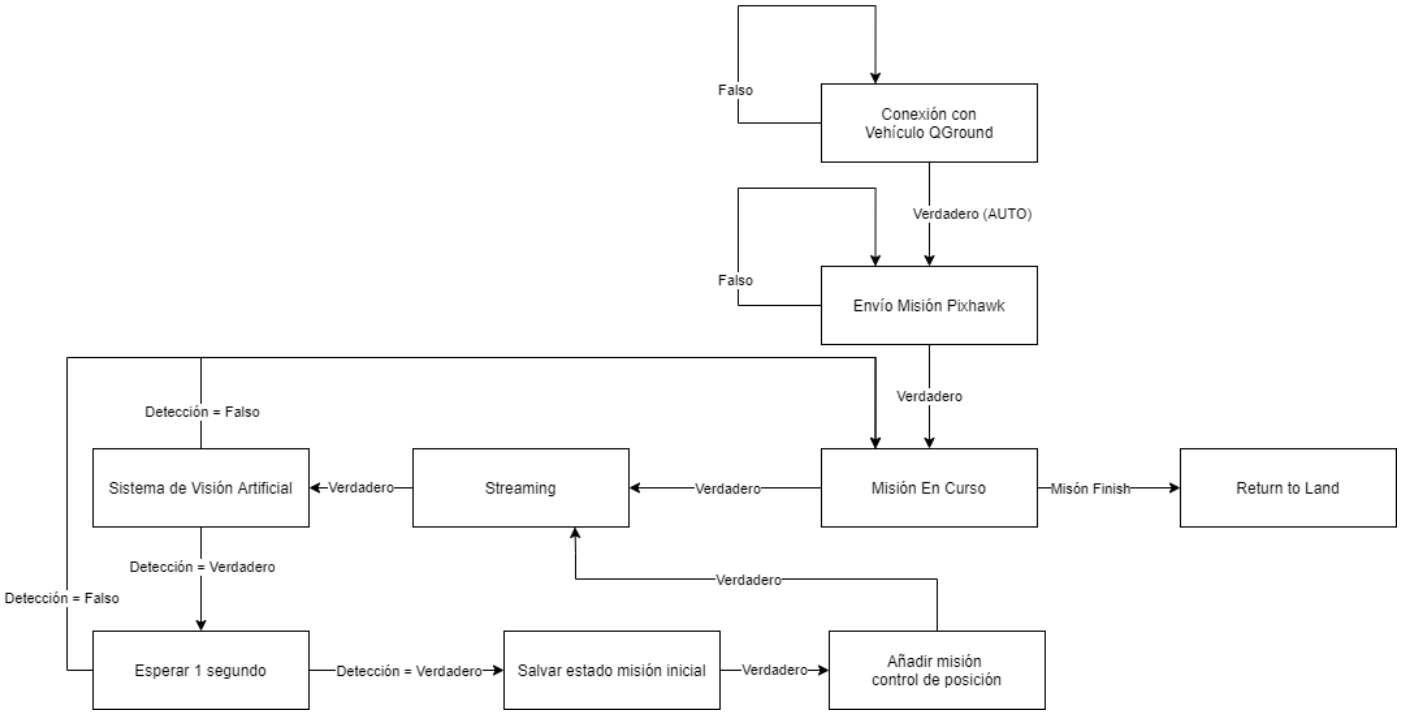
Según la ley es de obligación que haya siempre un piloto detrás de todos los sistemas de Gestión Autónoma de Misiones en UAVs Basada en Visión Artificial. Y es por lo que se debe configurar software que proporcione al piloto toda la información y el estado de la misión del UAV. Para ello, se va a instalar la herramienta QGround. Real Decreto 552/2014, artículo 23 quáter.

Requisito	¿Cuándo es necesario?
Un equipo de comunicaciones adecuado capaz de sostener comunicaciones bidireccionales con las estaciones aeronáuticas y en las frecuencias indicadas para cumplir los requisitos aplicables al espacio aéreo en que se opere.	Dependiendo del espacio en que se opere, contrastado con el estudio aeronáutico de seguridad.
Un sistema para la terminación segura del vuelo.	En toda operación.
Dispositivo de limitación de energía del impacto.	En caso de las operaciones sobre aglomeraciones de edificios en ciudades, pueblos o lugares habitados o de reuniones de personas al aire libre.
Equipos para garantizar que la aeronave opere dentro de las limitaciones previstas, incluyendo el volumen de espacio aéreo en el que se pretende que quede confinado el vuelo.	En toda operación.
Medios para que el piloto conozca la posición de la aeronave durante el vuelo.	En toda operación.
Luces u otros dispositivos, o pintura adecuada para garantizar su visibilidad.	En toda operación.
Luces de navegación y luces anticolisión.	Vuelos nocturnos. (Art. 25 RD 1036/2017).
Transpondedor Modo S. El transpondedor deberá desconectarse cuando lo solicite el proveedor de servicios de tránsito aéreo.	Todas las aeronaves pilotadas por control remoto (RPA) que pretendan volar en espacio controlado, excepto operaciones dentro del alcance visual del piloto (VLOS) de aeronaves cuya masa máxima al despegue no exceda de 25 kg.
Dispositivo de visión orientado hacia delante.	Operaciones más allá del alcance visual del piloto (BVLOS).

3.Real Decreto 552/2014, artículo 23 quáter [1]

4.1 Lógica de vuelo utilizada

A partir del siguiente diagrama se podrá evaluar para cada caso y cada condición, que debe realizar el sistema:



38.Diagrama de Lógica de vuelo

4.2 Planes de Vuelo

Las misiones básicas que se deben utilizar para el funcionamiento de la librería DroneKit son las siguientes

Descargar misión

Lo primero se deberá conectar con el vehículo para descargar la misión actual. Este plan será utilizado posteriormente para cambios que se realicen sobre una misión ya empezada.

```
# Connect to the Vehicle (in this case a simulated vehicle at 127.0.0.
1:14550)

vehicle = connect('127.0.0.1:14550', wait_ready=True)

# Download the vehicle waypoints (commands). Wait until download is co
mplete.

cmds = vehicle.commands

cmds.download()

cmds.wait_ready()
```

Limpiar misión

En el caso de realizar cambios en una misión ya empezada, es necesario comunicarle a la controladora de vuelo cuál es la misión que debe seguir, por lo tanto, se eliminará cualquier misión anterior.

```
# Connect to the Vehicle (in this case a simulated vehicle at 127.0.0.
1:14550)

vehicle = connect('127.0.0.1:14550', wait_ready=True)

# Get commands object from Vehicle.

cmds = vehicle.commands

# Call clear() on Vehicle.commands and upload the command to the vehic
le.

cmds.clear()

cmds.upload()
```

Añadir nuevos comandos a la misión

En el caso de querer modificar la misión, se necesita comunicarle a la controladora de vuelo cuáles son las indicaciones nuevas y la parametrización necesaria para poder ejecutarlos. Por lo tanto, es necesario recopilar la misión actual, y añadir los nuevos comandos a la estructura de datos que contiene las instrucciones de la misión.

```
# Connect to the Vehicle (in this case a simulated vehicle at 127.0.0.
1:14550)

vehicle = connect('127.0.0.1:14550', wait_ready=True)

# Get the set of commands from the vehicle

cmds = vehicle.commands

cmds.download()

cmds.wait_ready()

# Create and add commands

cmd1=Command( 0, 0, 0, mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT,
mavutil.mavlink.MAV_CMD_NAV_TAKEOFF, 0, 0, 0, 0, 0, 0, 0, 10)

cmd2=Command( 0, 0, 0, mavutil.mavlink.MAV_FRAME_GLOBAL_RELATIVE_ALT,
mavutil.mavlink.MAV_CMD_NAV_WAYPOINT, 0, 0, 0, 0, 0, 0, 10, 10, 10)

cmds.add(cmd1)

cmds.add(cmd2)

cmds.upload() # Send commands
```

Modificar misiones

En el caso de hacer modificaciones sobre cualquier punto intermedio, final o inicial de la misión actual, se debe realizar una modificación entera de la misión. Para ello es necesario exportar toda la misión a una estructura de datos y modificar la misión en el momento que se necesite. Y, por último, borrar la misión actual y enviar la nueva.

```
# Connect to the Vehicle (in this case a simulated vehicle at 127.0.0.
1:14550)
vehicle = connect('127.0.0.1:14550', wait_ready=True)
# Get the set of commands from the vehicle
cmds = vehicle.commands
cmds.download()
cmds.wait_ready()
# Save the vehicle commands to a list
missionlist=[]
for cmd in cmds:
    missionlist.append(cmd)

# Modify the mission as needed. For example, here we change the
# first waypoint into a MAV_CMD_NAV_TAKEOFF command.
missionlist[0].command=mavutil.mavlink.MAV_CMD_NAV_TAKEOFF
# Clear the current mission (command is sent when we call upload())
cmds.clear()
#Write the modified mission and flush to the vehicle
for cmd in missionlist:
    cmds.add(cmd)
cmds.upload()
```

Enviar el drone a una ubicación especifica

Es necesario tener las coordenadas de la ubicación a la que se quiere enviar el drone.

```
# Set mode to guided - this is optional as the simple_goto method will
change the mode if needed.
vehicle.mode = VehicleMode("GUIDED")

# Set the LocationGlobal to head towards
a_location = LocationGlobal(-34.364114, 149.166022, 30)
```

```
vehicle.simple_goto(a_location)
```

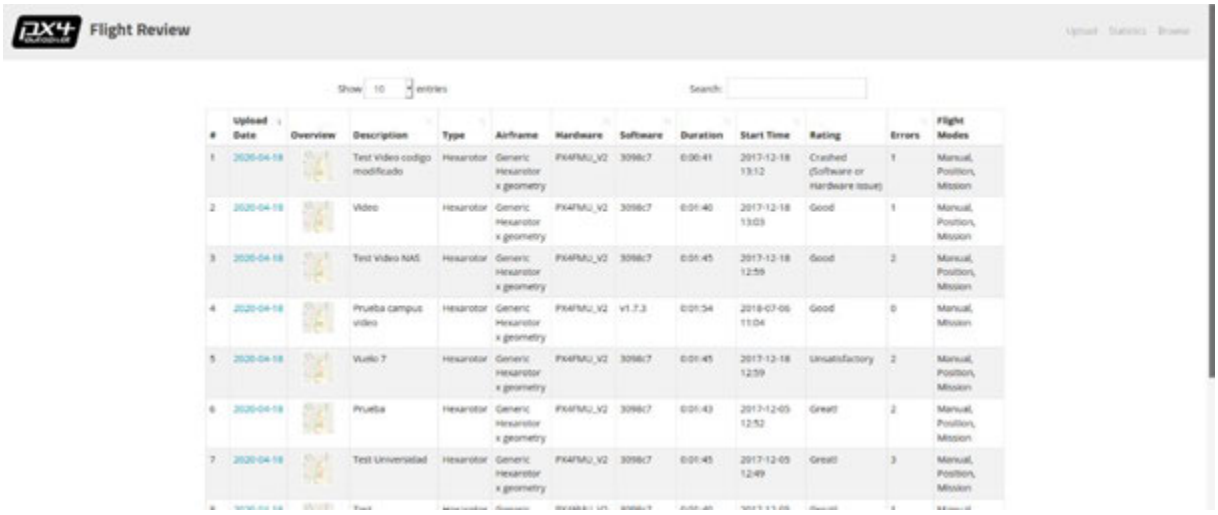
Mantener el drone en la ubicación actual con un control de altura a la espera de nuevas indicaciones

```
vehicle.mode = VehicleMode('LOITER')
```

A partir de estas funciones se puede modificar y crear cualquier tipo de misión de control básico del drone.

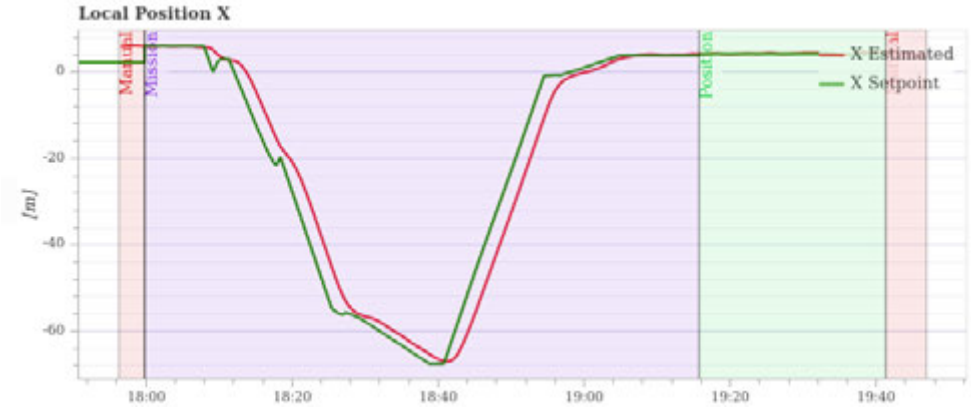
4.3 Validacion y verificacion de los datos

Además, para poder analizar los datos resultantes de estos vuelos y su funcionamiento, se ha implementado una plataforma de extracción de las gráficas obtenidas a través de los LOGS del vuelo realizado. Esta plataforma es una web, que utiliza el proyecto de openSource de PX4 Flight Review. URL: https://github.com/PX4/flight_review.

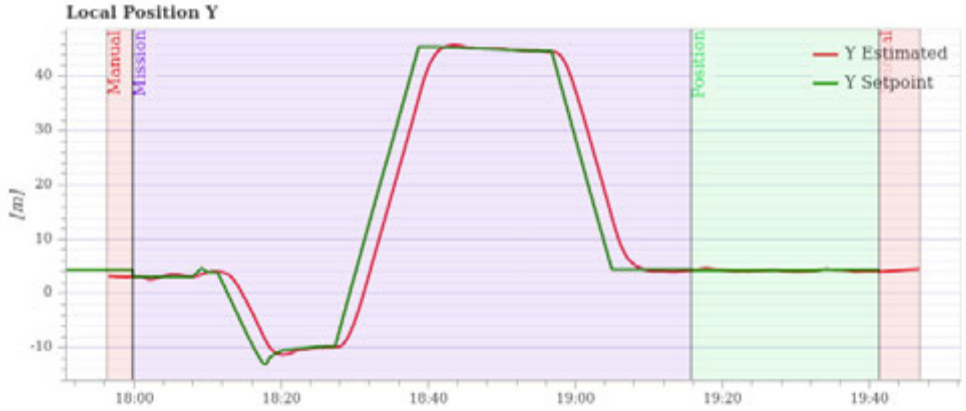


39. Plataforma web Flight Review

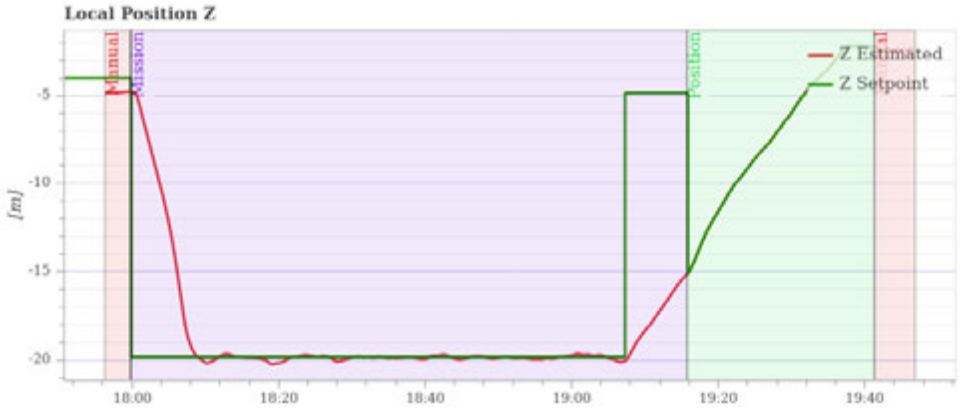
En esta interfaz se pueden controlar todos los vuelos recopilados por los usuarios. Para registrarlos se introducen a partir de una sección para subir el fichero LOG recogido por el PX4.



40.Gráfico posición X del drone en misión

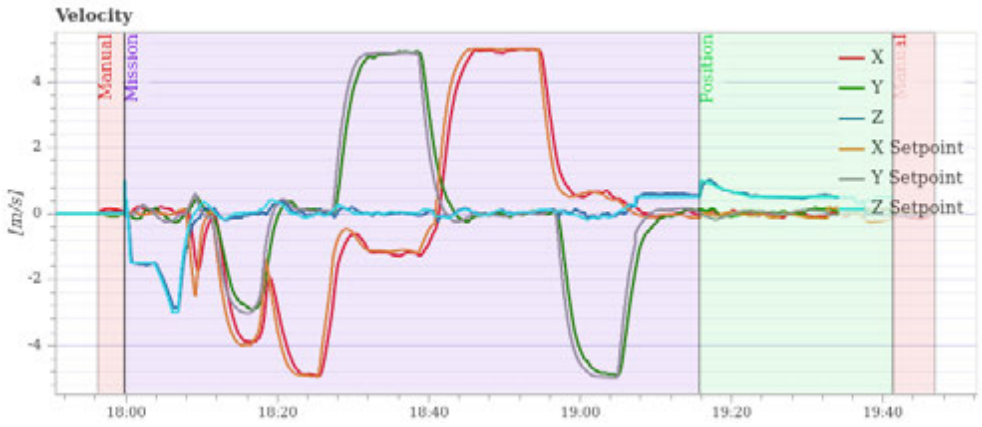


41.Gráfico posición Y del drone en misión



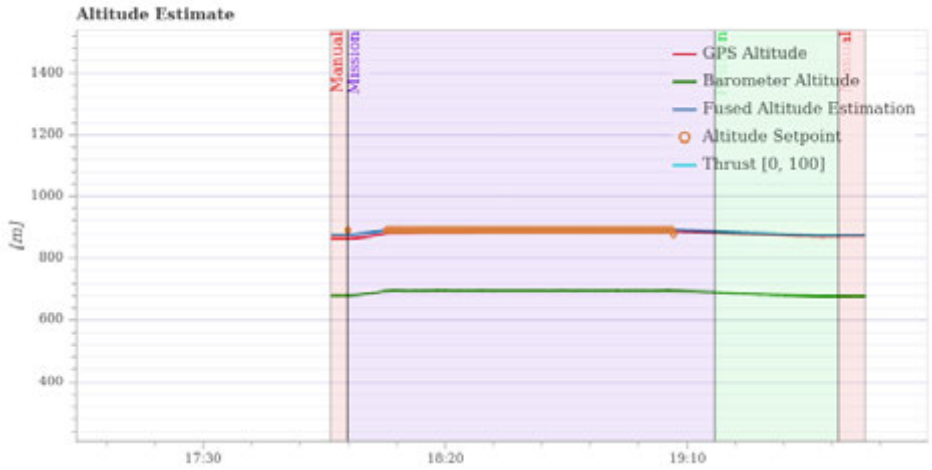
42.Gráfico posición Z del drone en misión

En esta gráfica, se puede verificar que la posición recogida por los sensores del UAV, correspondientes a las interacciones que se debe producir el sistema de control de UAV. De esta forma contempla como la position Z, X e Y debe quedar estabilizada en el momento que el sistema de visión detecte una persona y el sistema de control de vuelo mande una interrupción al PX4.



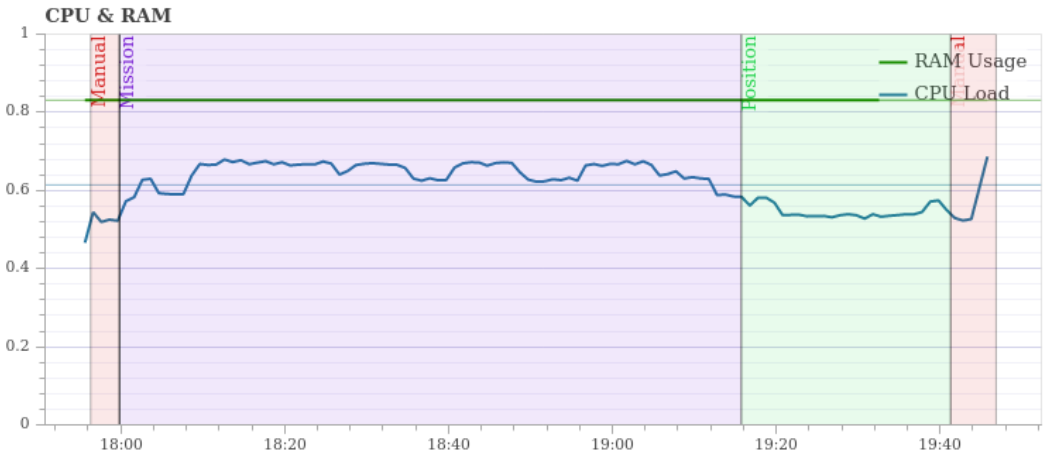
43.Velocidad del drone en misión

La velocidad debe quedar estabilizada en el momento que se produzca una interrupción por el sistema de control, y queda reducida prácticamente a 0. De esta forma el UAV permanecerá en la misma posición con pequeñas variaciones debido a las posibles alteraciones de las condiciones meteorológicas.



44.Estimación altura drone en misión

A partir de este grafico, se puede verificar que la altura que ha tenido el UAV durante todo el vuelo ha sido la misma, puediendo asegurar que el sistema de vision artificial ha funcionado en las mejores condiciones. Ademas, se puede controlar la altura, nada mas detectar una persona por el sistema de vision artificial, que le comunica al sistema de control de vuelo que debe mandar una interrupción para que el UAV permanezca en modo "LOITER" hasta que la persona desaparezca. Aunque dado que la altitud siempre deberá ser la misma, esto no es de tanta utildad, y es mas informativo utilizar otras graficas anteriormente enseñadas.



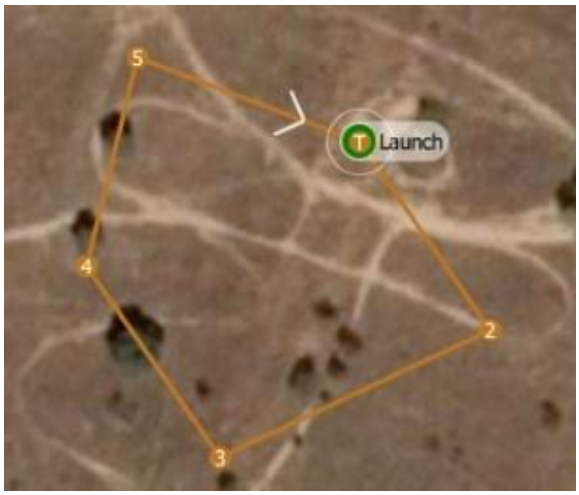
45.Carga del procesador y RAM en Pixhawk durante misión

Esta última grafica comprueba la carga del procesador Pixhawk para controlar en el momento en el que está recibiendo información de la Raspberry Pi para indicarle las instrucciones ante una interrupción. De esta forma, el uso de la CPU y RAM debe subir por el acceso a los procesos de vuelo que permiten salvar el estado actual y añadir nuevas instrucciones.

5. RESULTADOS Y EVALUACIONES

En este apartado se van a comentar distintos casos de uso que se han propuesto para aplicar los sistemas creados para este proyecto. En estos casos se identificará la finalidad de la prueba, el resultado esperado, el resultado obtenido y un breve comentario de cómo han funcionado los algoritmos frente al tipo de circunstancia que se esté tratando.

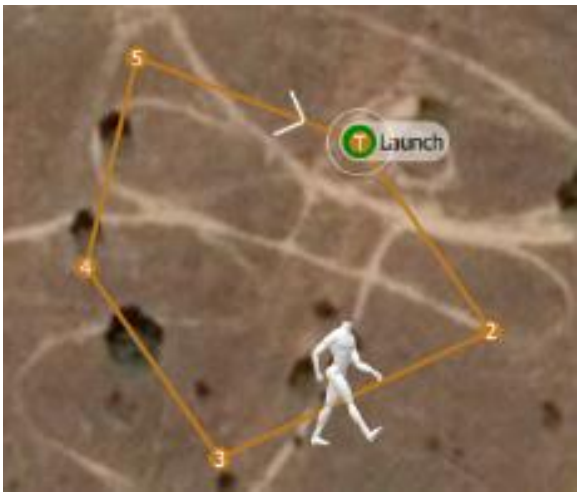
Prueba 1



46.Prueba 1

Descripción	**Resultado Esperado**	**Resultado obtenido**
En esta prueba se evaluará la solidez del sistema y la capacidad para detectar posibles errores en un terreno libre de personas examinando cómo actúa el sistema de visión artificial. Primero de todo se enviará la matriz de Waypoints al Pixhawk. Tras iniciar su vuelo desde el punto Launch, el UAV debe recorrer todos los puntos hasta llegar al 5 y entonces retornar a su punto de inicio.	El resultado óptimo del sistema de visión artificial y el sistema de control de vuelo, no debería haber tenido ninguna "falsa" detección de personas. De modo que no hubiese tenido que interactuar el sistema de control de vuelo verificando si la detección es consistente. Y por lo tanto habría procedido con la ejecución del plan sin producir ninguna interrupción.	El resultado que ha obtenido el sistema de visión artificial ha sido detectar de forma espontánea "posibles" cuerpos. Pero de tal forma que, al no haber sido consistente con el tiempo de ejecución, y el objeto no ha sido detectado por un lapso mayor a 1 segundo. Se ha proseguido con la ejecución de la misión inicial.

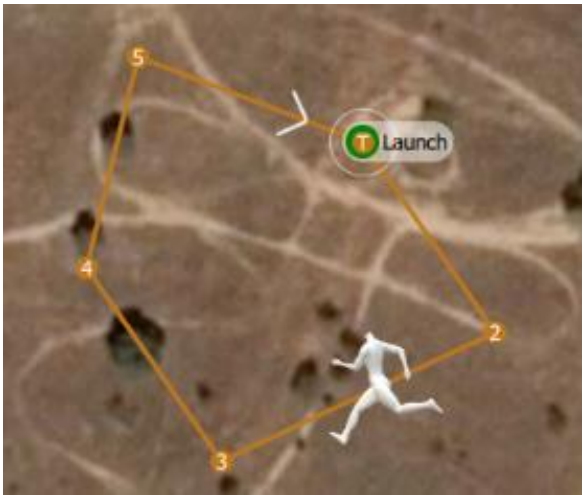
Prueba 2



47.Prueba 2

Descripción	Resultado Esperado	Resultado obtenido
En esta prueba se evaluará cómo el sistema es capaz de detectar una persona. Para ello se estimará el momento concreto en el que se detecta a esta persona y la distancia en la que se encuentra. Dado que el UAV tendrá la capacidad de focalizar el cuerpo de la persona desde un punto muy lejano, se valorará la inteligencia del sistema para detectar personas a larga distancia.	El sistema deberá adaptarse a los parámetros del clasificador preconfigurados, y tendrá que detectar personas que ocupen el tamaño configurado. Para ello ha de hacerlo cuando la persona se encuentre a una distancia horizontal aproximada de 5 metros y vertical de 6 metros. Dado que el tamaño promedio que ocupará una persona cuadrará con los parámetros.	El sistema se comporta correctamente. Y es capaz de detectar de forma aproximada a la persona a una distancia que se mantiene dentro de error configurado. Una vez detectada la persona, el sistema de control de vuelo modificará la misión actual, e introducirá el parámetro de LOITER para permanecer en el mismo punto hasta que la persona deje de permanecer en el foco de la cámara. Cuando la persona se ha movido, el UAV recupera su misión inicial desde el punto que se ha interrumpido.

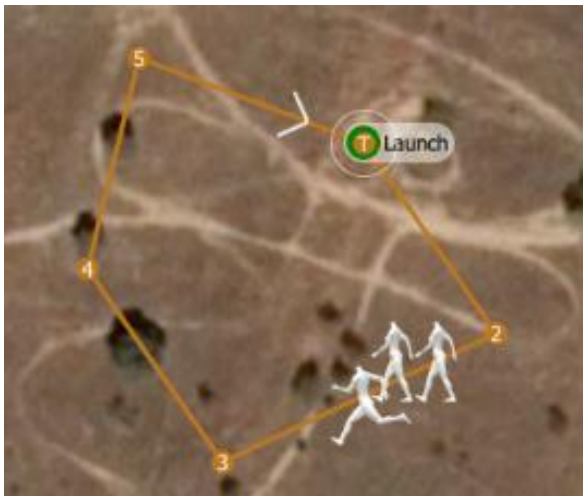
Prueba 3



48.Prueba 3

Descripción	Resultado Esperado	Resultado obtenido
Se presenta una persona corriendo que atraviesa de forma perpendicular la dirección del UAV.	El UAV debe localizar el cuerpo y realizar una interrupción antes de que salga del foco de la imagen. Esto debe suceder a pesar de que el sistema reanalice la imagen para verificar que el cuerpo detectado es real. Dado que desde el punto que se realiza la detección permite al corredor permanecer dentro del rango de la cámara por más de 3 segundos.	La cámara no tiene la resolución ni el framerate necesario para poder capturar de forma clara la persona corriendo. Por ello tras posibles detecciones de la persona, no es capaz de verificar al cabo de 1 segundo que la persona detectada sigue en el mismo punto. Esto se debe a la calidad de la cámara y a las capacidades del sistema de visión artificial.

Prueba 4



49.Prueba 4

Descripción	Resultado Esperado	Resultado obtenido
El campo de visión de UAV presenta una multitud de personas. Dos de ellas personas que andan o permanecen en la misma posición sin muchas alteraciones y otra que se mueve rápidamente.	El sistema, como se ha explicado anteriormente, priorizará la detección de personas en la imagen sobre la capacidad de detectar una cantidad real de personas. Por ello, debe detectar varios cuerpos en movimiento, y a pesar hacerlo sobre una persona en movimiento, es capaz de contrastar la detección con los demás cuerpos de la imagen.	El UAV se detendrá tras detectar personas en varios fotogramas seguidos, tras un lapso de 1 segundo. Y permanecerá estacionado en la posición de la detección hasta que no exista ningún cuerpo dentro del rango de visión de la cámara.

6. ESTUDIO SOCIO-ECONÓMICO

En este apartado se analizarán los costes totales relacionados con la elaboración de este proyecto. Para ello se contabiliza el tiempo utilizado por un solo ingeniero Informático recién graduado, considerado como Ingeniero Junior.

Considerando que la realización de este proyecto conllevaría una duración de 6 meses a jornada completa ha de analizarse el total de horas y el salario que se le debe atribuir a este trabajador.

En total este trabajador habrá realizado un total de 8 horas al día durante 5 días a la semana, esto durante 6 meses:

$$Horas\ Totales = 8\ {}^{horas}/_{día} \times 5\ {}^{días}/_{semana} \times 4\ {}^{semanas}/_{meses} \times 6 = 960\ horas$$

Al tratarse de un Ingeniero Informático Junior, el sueldo asociado por convenio será de 23.000€ brutos anuales. Por ello al tratarse de medio año, se reduce el coste total del ingeniero a la mitad, en total 11.500 €.

Sueldo bruto anual	23.000,0€
Retenciones por IRPF	3.010,7€
Cuotas a la Seg. Social	1.460,5€
Sueldo neto anual	18.528,8€
Tipo de retención sobre la nómina	13,09%
Sueldo neto mensual (12 pagas)	1.544,1€

4.Repercusiones económicas sobre personal

Por ello el Ingeniero tendrá un salario neto de 1.544,1 € durante 6 meses.

Además de los costes relacionados con el personal, el proyecto tiene algunos gastos materiales:

Material	Coste
Ordenador para desarrollar el proyecto HP Pavilion 15-cb	1100,0 €
Raspberry Pi 3 B + Pi Camera	40,0 € / 15,0 €
Drone hexarotor + PixHawk	500,0 € / 90,0 €
Material adicional	50,0 €
	1795,0 €

5.Repercusiones económicas sobre material

Por lo tanto, los gastos totales para la elaboración de este proyecto contando los gastos de personal y de material son:

$GastoTotales = GastosPersonales + GastosMateriales = 11.500{,}0\ € + 1.795{,}0\ € = \ 13.295{,}0\ €$

7. CONCLUSIONES Y TRABAJO FUTURO

A día de hoy los drones representan una herramienta fundamental en muchas de las áreas profesionales y con este proyecto se ha querido demostrar que con unos bajos recursos económicos y poca materia es posible crear un sistema autónomo capaz de realizar toma de decisiones en situaciones específicas. Para ello, se ha implementado un sistema de gestión de la misión, que permite reaccionar ante las percepciones generadas por el sistema de visión artificial embarcado en él. A partir de esta detección se crea una interrupción en la misión para tomar el control del vuelo y mandarle nuevas directrices al PX4. Después de realizar la acción que el sistema de control de vuelo ha solicitado, la misión inicial se recupera y se continua con su ejecución normal.

El desarrollo de este proyecto se ha diferenciado principalmente en 3 componentes:

El sistema de visión artificial, para los cuales se han valorado las mejores alternativas aptas para un cómputo en tiempo real con una Raspberry Pi. La alternativa más adecuada debido a su capacidad para obtener mejores resultados a menor coste ha sido los clasificadores Haar, por medio de unos métodos se consigue mejorar y agilizar la evaluación de cada fotograma de la imagen capturada por el sistema de grabación de la Raspberry Pi. En concreto el clasificador creado ha utilizado un total 1500 ejemplos que han sido recogidos en un 30% por la Raspberry Pi camera, y el 70% han sido generados a partir de imágenes no positivas e imágenes positivas. Para los ejemplos negativos se han obtenido un total de 2000 ejemplos.

El sistema de control de UAV ha sido elaborado a partir de una sola lógica de interacción, pero se han presentado todas las bases para posibles futuras interrupciones con más elementos y otras posibles consecuencias. De esta forma, el UAV es capaz de detenerse tras detectar que el cuerpo humano está en el campo de visión, y a partir de ahí salva la misión inicial, e introduce una nueva en la que únicamente se queda en modo control de posición hasta que el cuerpo humano sale del campo de visión del drone para así retomar la misión inicial.

Por último, el tercer componente importante es la interconexión entre la controladora de vuelo y la Raspberry Pi, para lo cual se deben habilitar los puertos en ambos dispositivos y configurar los elementos de hardware que interconecten los ordenadores, analizado en Anexo 1 Comunicación Raspberry - Px4.

Para futuros proyectos, se debería valorar la posibilidad de ampliar la capacidad de computo del ordenador que contiene el sistema de visión artificial. Para ello algunas de las posibles futuras implementaciones serían, utilizar Deep learning para identificar un mayor conjunto de objetos y así poder elaborar lógicas de vuelo mucho más complejas. Una vez ampliada la capacidad de cómputo, se añadirían a la misión más UAV´s capaces de ejecutar misiones como agentes y con la posibilidad de comunicación y toma de decisiones entre ellos para aumentar el área sobre el que sobrevolarían. De esta forma, se conseguiría obtener una mayor cantidad de interrupciones, así como un sistema más

seguro si se quisiera enfocar a tareas de vigilancia. Sobre el hardware de visión, se podría ampliar la cantidad de posibles sensores sobre los que pudiera trabajar el sistema de visión artificial, esto implicaría un rediseño del chasis del UAV para poder albergar más hardware y también cargar más baterías a causa del incremento de energía necesario para realizar los planes de vuelo.

ANEXO 1 COMUNICACIÓN RASPBERRY - PX4

Una vez que el sistema de visión funciona correctamente, el sistema debe comunicarse con el sistema de control del UAV. Para ello se debe realizar una comunicación entre dos ordenadores a bordo.

La comunicación entre los dos ordenadores a bordo habilitará la posible transmisión de información con el fin de realizar un sistema de vuelo mucho más completo. Esto permite recopilar más información relevante durante el vuelo y que esta pueda ser tratada para modificar el plan de vuelo si se considera necesario.

Deberá ser importante establecer una comunicación segura a través los conectores correspondientes y crear una fijación que imposibilite la desconexión durante el vuelo. En el caso de evitar una desconexión durante el vuelo, tanto el sistema debe estar preparado para recuperar la última misión establecida, y además es recomendable que el piloto deba estar atento para cambiar al piloto manual.

Además, será importante que la conexión entre los dos ordenadores sea de calidad y no existan cortocircuitos que puedan dañar las placas base de ambas computadoras. Es esencial que estén bien configurados para hacer funcionar correctamente la librería de control de vuelo DroneKit.

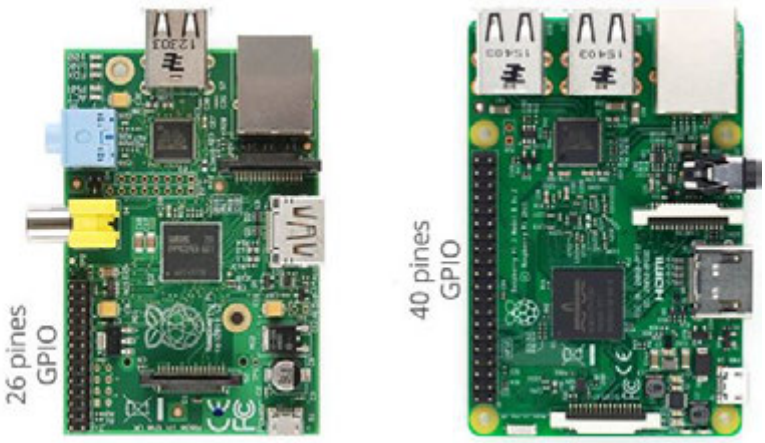
A continuación, se disponen de forma clara, como se deber realizar la conexión en cada extremo de la comunicación:

Interfaz de comunicación Raspberry

Desde la primera Raspberry Pi, se implantó un sistema de entrada y salida con la finalidad de comunicar elementos externos llamado GPIO (General Purpose Input Output). Gracias a que inicialmente el ordenador fue diseñado como un elemento educativo, este sistema de entrada y salida tiene una gran accesibilidad y facilidad de uso.

Dicho sistema de entrada y salida, lo forman un conjunto de pines directos a la placa madre, comúnmente distribuidos en 2 filas de forma paralela, por lo que su identificación es muy rápida.

A lo largo de los modelos de Raspberry Pi, se han ido introduciendo mayor cantidad de pines ampliando las posibilidades de estos modelos, pasando de 26 a 40 pines. Esto ha sido así gracias a la eliminación de sistemas de comunicación que quedaron inutilizados.



50.Diferencia GPIO entre generaciones de RPI

Con el fin de entender la finalidad de cada pin, se han agrupado siguiendo el siguiente esquema de colores:

- Amarillo, dispone de 2 pines, la utilidad es proporcionar una alimentación estable a 3.3V, se suele utilizar para alimentar pequeños dispositivos o ventiladores externos.

- Rojo, dispone de 2 pines, la utilidad es proporcionar una alimentación a 5V. Esta alimentación puede servir para la Raspberry sin la necesidad de conectar un adaptador de corriente micro USB. También es útil como interfaz de alimentación de salida, con la posibilidad de alimentar dispositivos a 5V.

  Es importante destacar, que como se ha comentado anteriormente, estos pines no disponen de protección para comprobar que el nivel de tensión es el adecuado, por lo tanto, a la hora de realizar la conexión, en el caso de sobrealimentar la RPI esta podría llegar a quemarse.

- Naranja, dispone de 26 pines, con el propósito de generar una alimentación de entrada/salida de 3.3V estables. Es importante tener en cuenta que una sobrealimentación de la placa madre, podría ocasionar que se produzca una sobretensión en los componentes y estos lleguen a quemarse.

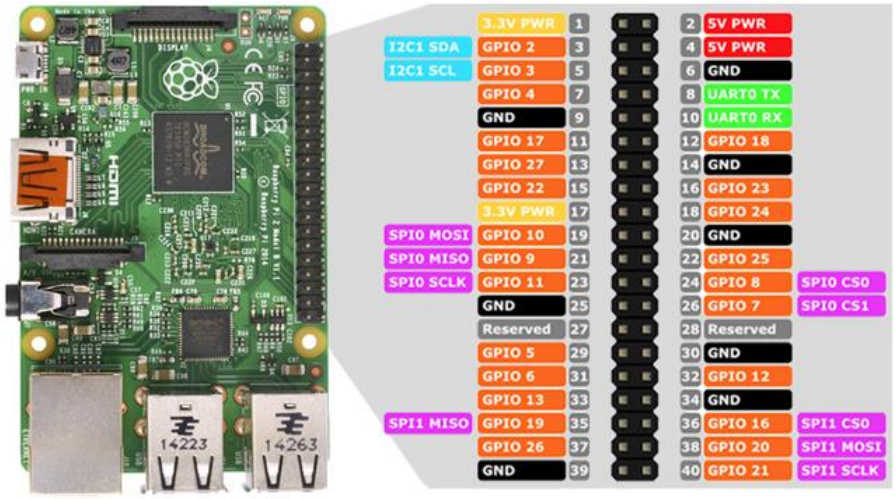
- Gris, dispone de 2 pines reservados para la RPI.

- Negro, dispone de 8 pines, son conexiones a tierra que tienen como finalidad proteger al usuario y el sistema de posibles electrocuciones y disminuir en la medida de lo posible el ruido electromagnético.

- Azul, dispone de 2 pines. Sirve como medio de comunicación mediante el protocolo I2C [18]. I2C es un protocolo en serie de dos hilos de baja velocidad,

que sirve para conectar dispositivos que cuentan con este método de comunicación. Con este protocolo se utiliza una relación maestro-esclavo

- Verde, dispone de 2 pines UART (Receptor / Transmisor Asíncrono Universal) que tienen como finalidad dar acceso al uso del ordenador sin necesidad de un teclado o ratón.
- Morado dispone de 5 pines de la misma forma que I2C. Los pines SPI sirven para comunicar dispositivos compatibles con este protocolo y utilizan la misma relación maestro-esclavo.

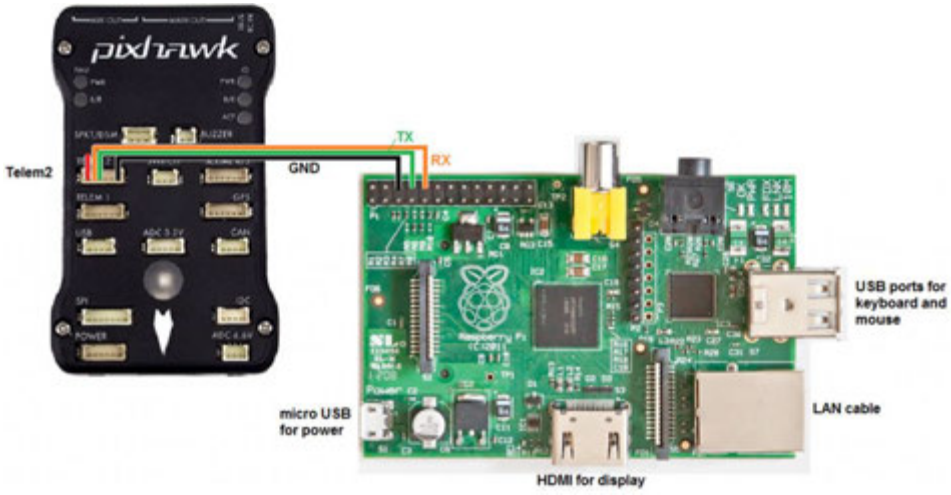


51.Esquema GPIO

Es importante conocer las limitaciones de los pines GPIO, y no se debe conectar componentes que trabajen a 5V en entradas que funcionen 3,3V. Además, es importante comprender que los pines de 3.3V fueron diseñados para ofrecer una corriente de 3mA.

Una vez expuesto todos los conceptos relacionados con el GPIO de la Raspberry Pi, se va a explicar cómo se debe realizar la comunicación con la controladora de vuelo.

Tan solo se van a utilizar 3 pines del GPIO de la Raspberry Pi, los pines del puerto serie (UART) y el pin de tierra. El pin RX, se encargará de la recepción de la información, de esta forma debe estar conectado en el otro extremo al pin TX. Y a su vez el pin TX de la Raspberry Pi ha de estar conectado al pin RX de la controladora de vuelo. En el caso de solo haber conectado un pin TX a uno RX, la comunicación será unidireccional, por lo tanto, es muy importante comunicar ambos pares de pines.



52.Interconexión RPI con Pixhawk

Para poder utilizar el puerto serie del GPIO de la Rasberry Pi, no es suficiente con conectar los cables a sus respectivos pines, además es necesario activar estos pines en la configuración interna de la Raspberry Pi. Para poder habilitar la comunicación UART se deben seguir los siguientes pasos:

1. Acceder a la configuración de la Raspberry Pi, con el comando sudo raspi-config
2. Una vez dentro del menú, se deberá seleccionar la opción opciones de interfaz.
3. Activar la opción Serial Port a yes, para que pueda recibir o enviar datos a través de estos puertos.

Interfaz de comunicación PX4

La controladora de vuelo Pixhawk cuenta con una gran cantidad de opciones para adaptar componentes externos y poder enviar y recibir datos de ellos.

53.Conexiones RPI [19]

Una vez se tiene claro cuál es la función de cada conexión entrante de la controladora de vuelo Pixhawk, se va a explicar cómo se realiza la comunicación con el ordenador Raspberry Pi.

Este conector se ejecuta la conexión a través del puerto "Telem 2". Pudiendo utilizar cualquiera de los demás puertos de Telemetría, pero dado que el primer puerto ya está siendo utilizado por la telemetría que envía al usuario, la opción más ordenada es conectarlo al 2 puerto.

Y ahora se necesita crear un adaptador personalizado para introducir los cables salientes de la Raspberry Pi ya que la controladora de vuelo utiliza conectores cerrados, a diferencia de la Raspberry pi que utiliza pines aislados.

El esquema que se debe seguir para crear este conector debe ser el siguiente:

Pin	Signal	Volt
1 - red	VCC	+5V
2 - blk	TX	+3.3V
3 - blk	RX	+3.3V
4 - blk	CTS	+3.3V
5 - blk	RTS	+3.3V
6 - blk	GND	GND

6.Esquema de colores-voltaje de GPIO - RPI

Para el conector, únicamente se utilizarán los pines 2,3 y 6. Tal y como se ha comentado anteriormente la conexión TX de un extremo debe conectarse con la RX del otro, y viceversa. Es importante destacar que para este esquema se está alimentando de forma externa cada ordenador de abordo por separado, para evitar problemas de intensidad de corriente.

Otro distinto tipo de solución sería alimentar la Raspberry Pi a través de la controladora de vuelo, a través de los puertos de alimentación. Y también podría plantearse la solución de alimentar el Pixhawk a través de la Raspbery Pi.

A diferencia de la Raspberry Pi, la controladora de vuelo no requiere ninguna configuración interna del software para habilitar los puertos conectados. Por lo tanto, el sistema estaría disponible para enviar y recibir información.

ANEXO 2 INSTALACIÓN DRONEKIT

Para instalar la librería de DroneKit se deberá acceder a la Raspberry Pi con CLI. Primero se instalan algunas dependencias necesarias para hacer funcionar la librería DroneKit:

```
sudo apt-get install python-pip python-dev
```

Una vez todas las dependencias han sido instaladas, se procederá con las librerías de DroneKit y DroneKit Software in the loop:

```
pip install dronekit

pip install dronekit-sitl
```

Para utilizar las librerías de Dronekit, existen varias opciones:

- Utilizar herramientas propias de la librería, lo cual es una forma rápida y cómoda de visualizar información del estado del UAV. Pero no es una solución cómoda para aplicar planes de vuelo, ya que la mayoría de las herramientas traen planes predefinidos y poco útiles para este proyecto.
- Crear un Script propio que interprete toda la información recogida de la controladora de vuelo y pueda crear acciones personalizadas por el usuario para aplicar cuando se considere necesario y la lógica del plan de vuelo lo requiera.

Una vez todas las librerías hayan sido instaladas, en este proyecto se utilizarán un conjunto de scripts programados específicamente para este proyecto, que manejaran toda la información recopilada de los puertos serie RX y TX (previamente han debido ser habilitados para crear la comunicación bidireccional entre la controladora de vuelo y la Raspbery Pi).

ANEXO 3 INSTALACIÓN QGROUND

El software de QGround va a ser utilizado para que previamente se configure la misión que va a realizar el UAV. Además, el software proporciona información instantánea que obtiene de la telemetría del drone. Esto es esencial para que todas las operaciones que se vayan a realizar con el UAV sean seguras y legales.

Se utilizará un equipo montado en tierra para poder recibir toda la comunicación del UAV. El equipo con el cual se realizará todo el control de tierra del UAV tiene las siguientes especificaciones:

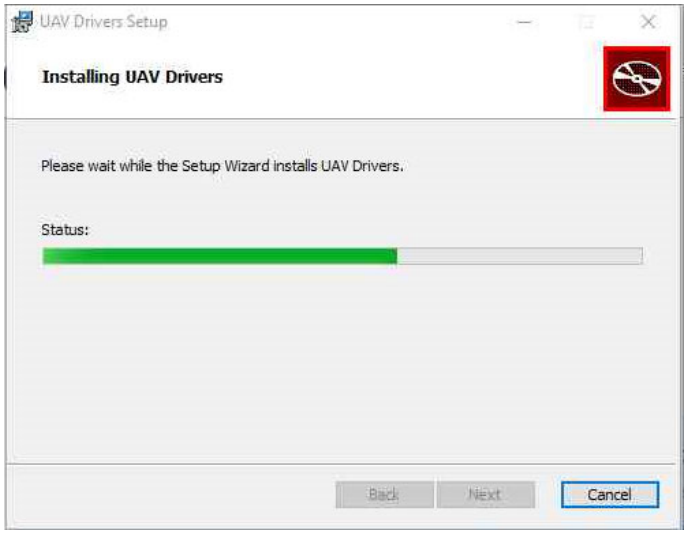
- Procesador: Intel i7 7700HQ 4 núcleos, 8 hilos
- RAM: 20GB
- SO: Windows 10
- Grafica GTX 1050

Previamente antes de proceder con la instalación es necesario establecer los controladores que permiten utilizar la telemetría del UAV.

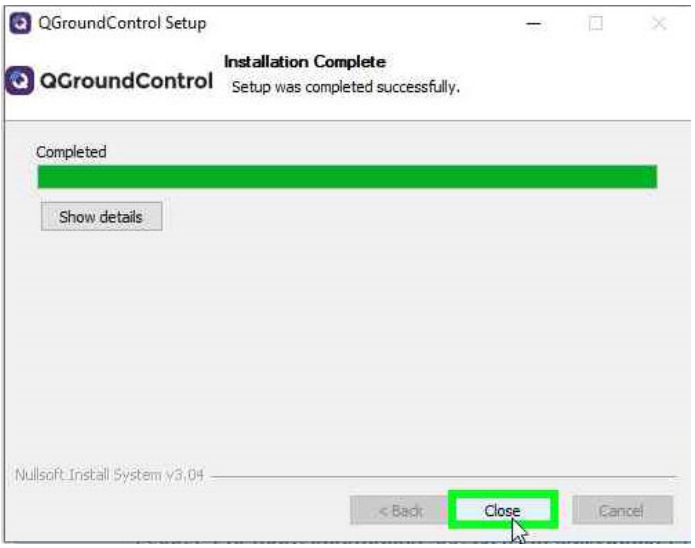
https://dev.px4.io/v1.9.0/en/middleware/drivers.html

Para la instalación de QGround se debe descargar el fichero ".exe" en caso de Windows de la siguiente dirección:

https://s3-us-west-2.amazonaws.com/qgroundcontrol/latest/QGroundControl-installer.exe



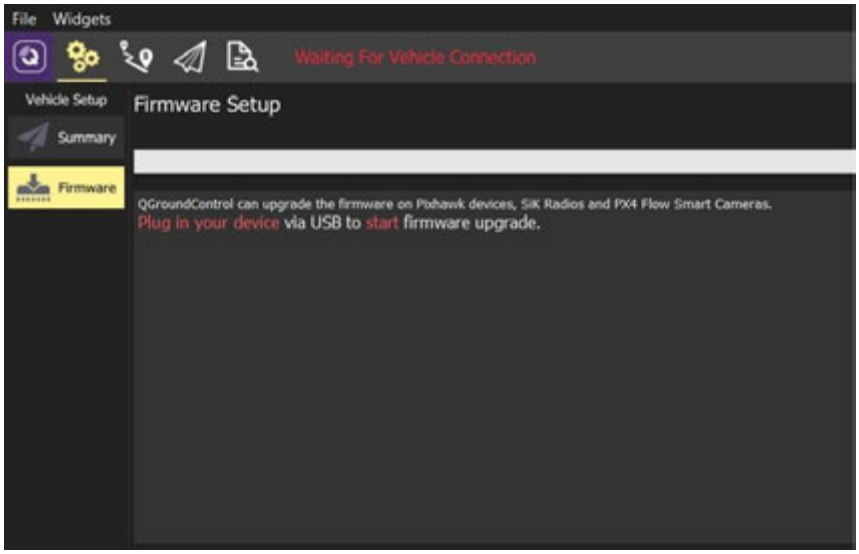
54.Proceso instalación QGround(1)



55.Proceso instalación QGround(2)

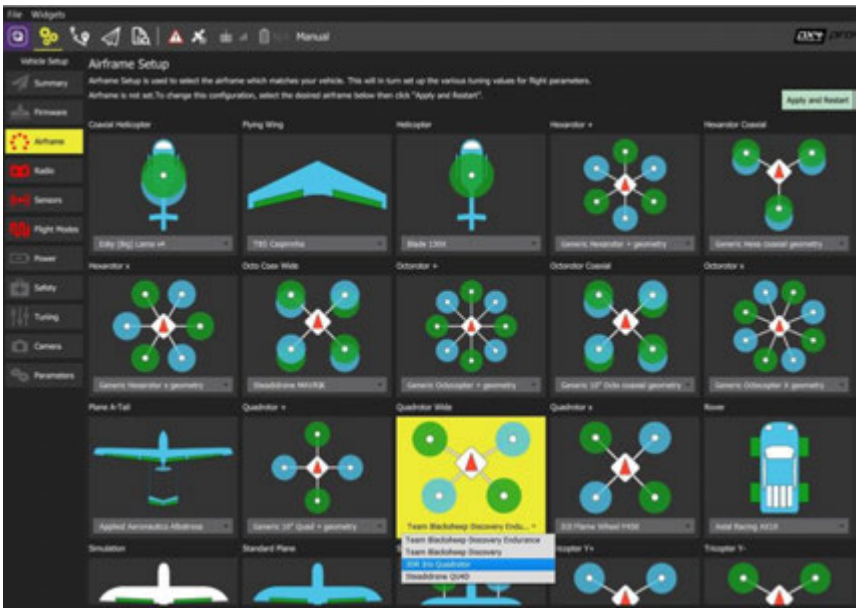
Una vez el software se haya instalado correctamente es necesario configurar un nuevo vehículo aéreo. Para ello se deben seguir los siguientes pasos:

8. Seleccionar el firmware de la controladora de vuelo.



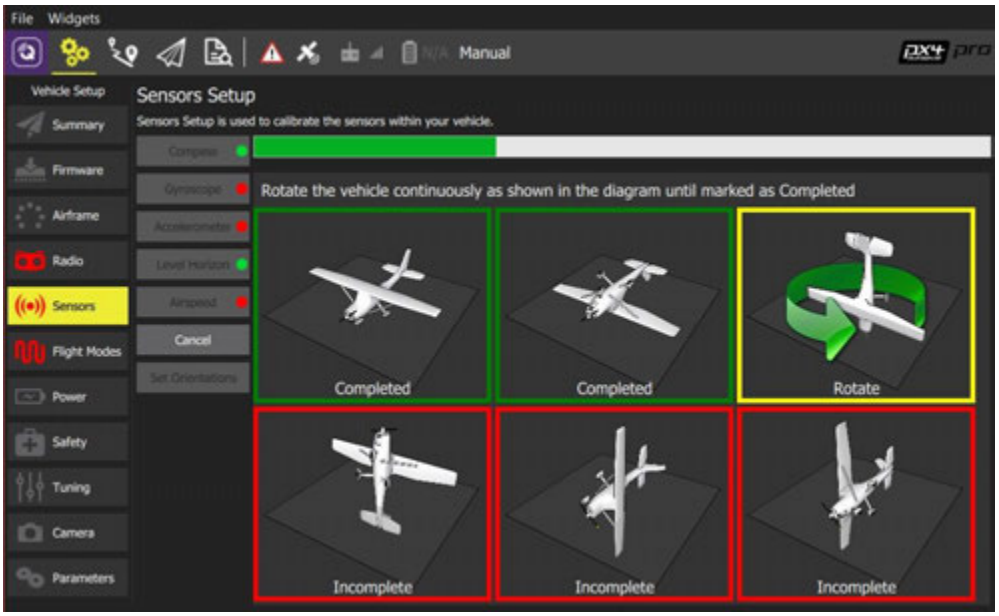
56.Selección firmware QGround

9. Configurar la estructura del drone.



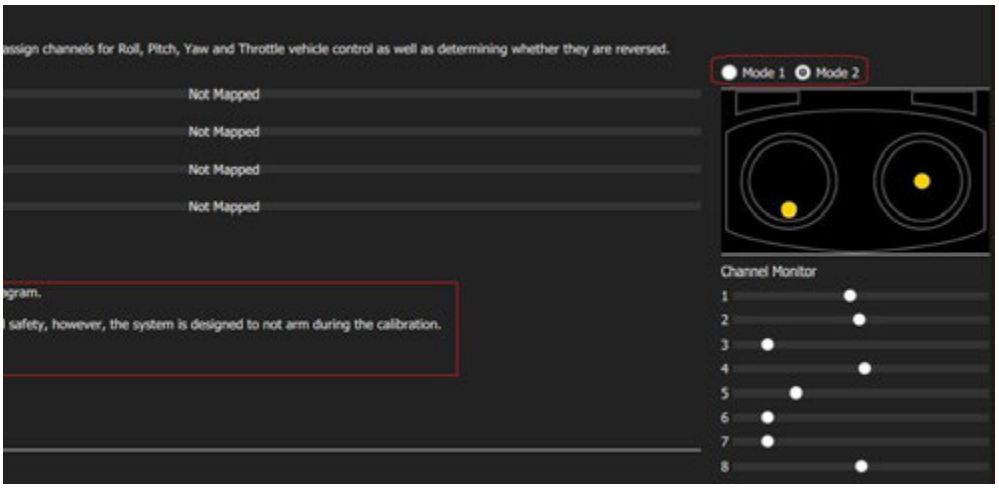
57.Selección estructura drone

10. Realizar las calibraciones correspondientes a todos los motores y sensores principales.



58.Procesos de calibración de los sensores

11. Calibrar los potenciadores del mando radiocontrol y los ajustes de emergencia.



59.Calibración mando radio control

12. Una parte muy importante consiste en configurar la controladora de vuelo con las fuentes de energía con la que se va a alimentar al drone. Es fundamental que esta información sea precisa, ya que en caso contrario se puede ocasionar un accidente como consecuencia de una mala lectura de los valores de voltaje de la batería.

ANEXO 4 RESUMEN EN INGLÉS

Abstract

This paper covers all theoretical and practical concepts about the design of an artificial vision system. The main reason of this artificial vision system is to detect people by a UAV and taking the pertinent actions in each case.

The system is based on Pixhawk flight technology, with the onboard computer Pixhawk and the PX4 flight software controller. Also, the software to launch pre planned mission has been QGround, which is part of the Project Pixhawk.

During the pre-planned mission, the artificial vision system will be executed over the component Raspberry Pi. This component is a small external onboard computer to compute all the data collected while the mission lasts. Besides, inside this computer an algorithm will be implemented by following a flight logic that allows detecting people who are within the UAV´s angle of vision. And due to this detection, the flight logic will determine which action the flight controller should take. To achieve this system works, will be necessary to implement the communication between the Raspberry Pi, which contains all the flight logic system and the artificial vision system, and the Pixhawk, which contains the PX4 flight software controller. This communication will give the Raspberry Pi the chance to interrupt the flight if needed.

To sum up, the UAV will be able to suspend the current mission due to an interest event, execute an action over the flight controller and then recover the status of the initial mission once the interruption is over.

Introduction

In this first chapter every concept of this Project will be explained. First of all, it is necessary to analyse the main reason of developing this project and to know how the laws will interference in this project. Also, this section will show the first achievements during the development of the Project. And to conclude this section, the structure of the whole document will be detailed.

Motivation

Nowadays, unmanned aerial vehicles are becoming trend, it could be because of professional photography and filmmaking drones, the gamification of UAV’s and the research of a drone public transport.

This drone technology is being impulse for many professional sectors and is a pleasure for me to contribute improvement this technology.

The focus of this project is to improve professional tasks and show how to develop a system with artificial vision functionalities that can recognise interest elements.

Drones are used for package transport with process automatization, surveillance tasks, rescue tasks, helping with some agriculture jobs and cattle raising and much more utilities. Therefore, is necessary to have drone hardware and software ready to be used in many circumstances and being a robust product with fails tolerance.

Regulatory framework

At present, there are many restrict aspects before using drones. This means that there are, as well, many restrict aspects to develop software for drones.

The first problem before fly is to know the permission to fly in this area, the reason of the fly and how the flight plan would be.

If the flight is going to be as a recreational use or as a research work, this means that it wouldn´t be necessary to have the licence to flight neither to have the AESA (Agencia Estatal de Seguridad Aerea) authorization. On the other hand, if the flight has professional purpose, then it must be necessary to have those permissions, also the pilot may have medical insurance, liability insurance.

Also, there are other restrictions about the time slot to fly, the maximum distance between the drone and the pilot, distance to the nearest airport and many aspects to keep in mind before fly.

Following these rules, the develop of this project, always keep in mind as a priority to not break the law and following all the recommendations of AESA.

Objective

The objective of this project is to achieve the implementation of a system that can do missions, have a compute system onboard and to make changes to the system during the fly. If the system meets the requirements, then the people recognise system could be created. By unifying every aspect explained in this document, an autonomous system will be created to do missions and cause interruptions on the plans if needed.

For the system develop, that can replay premade missions, the configuration will be through a tool called QGround. QGround allows the pilot to establish some necessary parameters to play a mission over the flight controller. Those parameters are, point of interest, speed, altitude, sleep time between two points of interest… QGround is used to communicate with the flight controller Pixhawk over the MavLink port, and then to the software flight controller Px4.

Raspberry Pi 3B+ is necessary to compute the capture data. It will be plugged through an external battery, which also power up the whole drone.

The artificial vision system will be able to detect elements of interest from a sequential of images captured with the Raspberry Pi.

To conclude, the system will need to be able to suspend the current mission in the case of detect elements of interest. If suspended the mission, the system will send instructions to the flight controller to realize some pertinent some actions, and then restore the suspend mission.

Structure

The present document shows the theorical and practical aspects about the design of an artificial vision system on a UAV. First, the theorical aspects will be explained. Then, every part of the design will be evaluated, we differentiate two main parts. The first one, will evaluate the precision of an artificial vision algorithm to detect people. This will have some problems, one of them es that the altitude during the fly can make the artificial vision system fail. After the vision system, a flight logic needs to be implemented. This flight logic will determinate which action should make the drone when the interruption occurred. And a third part will be explained as annexed, and show the way these two computers, Raspberry Pi and Pixhawk, are connected to work together.

To sum up, the system will be evaluated to determinate the best configuration of the system.

State of the art

Drone

In the aviation, drones are known as an unmanned aerial vehicle. But drones are not as we think they are. Drones started to exist in the First World War (1914-1918), where the main purpose were to train the United Kingdom army. Because of this advanced technology and the reliability of drones, it started to be used in spying and combat.

At the beginning of century XXI, drones started to be used as a professional tool. This meant that, drones were redesign as a small product with a light fuselage and les autonomy.

From the beginning to now, drones can be differentiated:

- Military drones
- Commercial drones

- Civil drones
- Recreational drones

Nowadays there are different kinds of drones:

- Fixed wing drone:

  This kind of drone use a long wing that allow to the drone to glide in the sky and use the air stream to rise and descend. They have a great autonomy because of the low power consumption. It causes a long flight and is the best kind of drone to use in long flight applications which require much precision.

  Unlike the rotatory wing, the fixed wing drones cannot take off and land vertically, because of this a landing track will be required. So, this kind of drones are not recommended to do hover flights.

60.Fixed wing drone

- Rotatory wing drone:

  The rotatory wing drone has in its fuselage some motors. These motors create a directional air stream to the ground. Because of this way of flight, long wing drone can glide, and those ones create their air stream way to fly.

  This kind of unmanned aerial vehicle has much more applications in hover flights. Also has the capability of take off and land with an easiest way.

  There are some different rotatory wing drones, depending of the number of motors they have.

  The most common motors are the one with brushes and the brushless. The brushes are used to change the motor coil polarity mechanically. Brushesless motors uses this polarity change electronically. Because of this, brushes motors waste a lot of heat and this means they lose efficiency. Besides, brushes motors are cheaper to manufacture and because of this are recommended for recreational uses.

61. Rotatory wing drone(hexacopter)

There are too many kinds of sensors we could put over the drone.

- Some of them are used to merge the data and get a more stable flight. This is done by the flight controller. Some of them are, gyroscope, accelerometer, barometer, sonar, GPS, compass…
- Other sensors can be used externally to the flight control. Some of them are LiDAR, image sensors, thermographic, static pressure…

Besides drones are called as unmanned aerial vehicle, there are some ways at the time to use it:

- They could be controlled by a pilot in a ground station.
- They could be controlled by a software of automate missions. In which a mission is replay as a set of instructions established before.

In the case of this project, we are using an hexacopter, with a hexagonal distribution of motors. It is power up with a lipo battery with 4 cells, and 5000mAh of capacity and a "c" value of discharge of 40C.

OpenCV

OpenCV word is differentiated in three parts, Open as an open and free software, C means computer and v means vision. So OpenCV is an open source library for commercial use which is focused in artificial vision and image processing. OpenCV started on 1999, was created by Intel.

This library has a lot of algorithms to use for image processing, about 2500 kinds of algorithms. And it has large tools to capture real time images, video processing, image modifying and detection of faces, bodies…

At the beginning, the library started to be as an object-oriented programming for the C++ programming language. With the passage of time OpenCV updated his compatibilities with more programming languages such as C++, Python, Java and Matlab.

Raspberry Pi

Raspberry Pi es a small computer oriented to students and researchers. It started on 2012 with the first model Raspberry Pi Model B. From that moment, there are many models, until 2020 there are 7 Raspberry Pi models.

In 2015 Raspberry release a new different model, Raspberry Pi Zero. Its unique reason was to be a tiny computer to include in every system. Until now there are 3 different version of this tiny computer.

Nowadays, Raspberry Pi is used in IOT projects and artificial vision projects. This is because it can be included in many platforms.

In this project we are using Raspberry Pi 3 Model B, which has the following specs:

- Chipset Broadcom BCM2837 at 1,2 GHz
- ARM Cortex-A53 at 64 bits y four cores
- 1 GB Memory LPDDR2
- 40 GPIO pins
- Dimensions: 85 x 56 x 17 mm

To power up this Raspberry Pi 3 Model B, we use a micro USB with 2,5 A. In this project we have adapted a tension regulator, to reduce the voltage of the main lipo battery from 14.8V to 5V.

Pixhawk

Pixhawk is an open software project. Mainly, Pixhawk is designed to be academic, hobby and for industrial community.

Pixhawk stand out because of the compatibility with high number of commercial sensors. This makes Pixhawk to be unique and the most valued alternative to create a robustness and reliability system. Also, Pixhawk project stand out for has more project products, such as QGround, which are design to be compatible. This compatibility makes a great user experience, much more than the competence. For this services and products integration, Pixhawk can be updated and fixed every hardware and software error.

In this project we have used 2.4.8 Pixhawk version, which has the following specs:

- 32-bit STM32F427 Cortex M4 core with FPU
- 168 MHz/256 KB RAM/2 MB Flash
- The 32-bit STM32F103 failsafe co-processor

Furthermore, it has the following included sensors:

- ST Micro L3GD20 3-axis 16-bit gyroscope
- ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer
- Invensense MPU 6000 3-axis accelerometer/gyroscope
- MEAS MS5611 barometer

MavLink

MavLink is a communication protocol. Mainly it was built to communicate drones, but also it can be used in many other projects.

MavLink es a king of point to point protocol. MavLink can send information to the drone or even also receive the drone status. MavLink controls some subprotocols, such as mission and parameters protocol. MavLink uses XML format to send or receive the messages.

Because of this protocol, the pilot is capable to receive live information about Pixhawk sensors, furthermore it can send external sensor information.

To make the connection possible and the package transmission will be effective, the system needs to be efficient. That's why every package is sent via MavLink has 8 bytes size. Also, Pixhawk has protection lose system to avoid the packet loss. Some of them are, authentication system, system to avoid package corruption…

Nowadays this project is used with the programming languages, such as C, C++, Python(2.7+,3.3+), C#, Objective, Java, JavaScript, TypeScript, Lua, Swift, Clojure, Go, Haskell.

Python

Python comes up late 1980, in 1989 by Guido van Rossum. His main purpose was to continue his past project. The purpose was to create an easy programming language.

Nowadays is rewarded fast programming language and high learning curve much more than before. At this point, Python stands out for its simplicity, its extensive library, fast develop and its dynamic typing.

Python is a multiparadigm language which support object orientation and imperative programming. A Python representative feature is to have the chance to define the variables and structures at execution time. Thankfully of this, is not necessary to define structures or variables before using it, this makes a fastest programming experience.

Nowadays this language is used in:

- Artificial Intelligence
- Big Data
- Data Science
- Testing Framework
- Web development

QGround

QGround is part of Pixhawk project. QGround is a fully dedicated software to control and design the flight. Nowadays is the most common used tool because of its great market power.

QGround gives a lot of parametrization over Pixhawk controller. Then it can gives an easy interface for communication via MavLink with drone without the needed of programming. Also, it gives the pilot the chance to observe the drone real time state.

QGround is compatible with most of Operative Systems:

- Windows
- OS X
- Linux
- iOS
- Android

Dronekit

DroneKit is an open software project, available to everyone. DroneKit wants to offer a fast and reliable communication between the flight controller and applications. Because to the low latency technology is used in DroneKit, it gives the chance to build a much more complex UAV with many functionalities.

DroneKit is a library that supports vehicles that are compatible with the MavLink protocol. And this is why this library gives greater utility to any type of vehicle (air, land, sea, etc.).

DroneKit currently uses the Python language as a programming language. Thanks to this, you have a wide range of possible applications and create an environment that is much easier to use and more versatile.

DroneKit is characterized by having many qualities:

- It is capable of connecting to one or multiple vehicles from a simple script.
- If you want to process vehicle telemetry for possible applications, it is capable of extracting this information in real time without the need to use graphical interfaces that make it impossible to process this information in real time.
- You can take control of the vehicle and drive it with the “GUIDED mode” wherever the user needs.
- Able to create missions, send them to the vehicle and initialize them.
- Has the ability to overwrite RC channel settings.

```
from dronekit import connect
# Connect to UDP endpoint.
vehicle = connect('127.0.0.1:14550', wait_ready=True)
# Use returned Vehicle object to query device state - e.g. to get the mode:
print("Mode: %s" % vehicle.mode.name)
```

Conclusionas and future work

Nowadays, drones represent a fundamental tool in many professional areas. With this project we wanted to show that which a low economic resource and not much materials you can achieve an autonomous system able to take decisions in specific situations. To achieve this autonomous system UAV can intervene the premade mission and take flight control when a person is detected. From this person detection an interruption in the mission is created to take flight control and send PX4 the new actions to realize. After the new actions are done, the system will activate the suspended premade mission to continue with the normal execution.

On the develop of this project there are 3 main parts:

The artificial vision system, in which had been valuated the best alternatives for compute images in real time with Raspberry Pi. The chosen one was Haar classifier, because of the cost produced compared with others. And then we can improve the Haar cascade algorithm with methods to speed up every photogram evaluation captured by the Raspberry Pi record system. The classifier had used 1500 photograms, which 30% of them has been collected by Raspberry Pi camera, and the 70% remaining was

generated from no positive images and positive images. For the negatives images we got 2000 photograms.

The control flight system was made by an only one logic interaction. But has been presented the bases to futures possible interruptions with more interest elements. By this way, the UAV can stop before detecting human body. From here the previous mission, the new mission will hover over the human body unit the human body disappear. Then it can resume the initial mission.

At last, the third important component is the interconnection between the flight controller and the Raspberry Pi. For this interconnection, we needed to enable communication ports in both dispositive and then configure the hardware to make this interconnection effective.

For future works, it should be evaluated the chance to extend the Raspberry Pi compute capacity to improve the artificial vision system. For this some possible could be to use Deep Learning to identify much more elements and elaborate more complex flight logics. Once the compute is improved, it will be added the chance to execute the mission with agents and their communication. This will be great idea to take decision more decisive if the other agents agree the actions. By this way it achieves more interruptions and it produced a better system to use in surveillance. On the vision hardware, it could be extended with much more sensors, to work improving the vision system. This change should be accompanied by more battery capacity to make more longer flights.

BIBLIOGRAFÍA

[1] Agencia Estatal Boletín Oficial del Estado, "BOE-A-2017-15721," 29 12 2017. [Online]. Available: https://www.boe.es/eli/es/rd/2017/12/15/1036.

[2] Aerial Insights, "¿Cuántos tipos de drones existen en el mercado?," [Online]. Available: https://www.aerial-insights.co/blog/tipos-de-drones/.

[3] EcuRed, "OpenCV," [Online]. Available: https://www.ecured.cu/OpenCV.

[4] OpenCV, [Online]. Available: https://opencv.org/about/.

[5] Wikipedia.org, "Raspberry Pi," [Online]. Available: https://es.wikipedia.org/wiki/Raspberry_Pi#Comunidad.

[6] MavLink, [Online]. Available: https://mavlink.io/en/.

[7] T. Khan, "Computer Vision — Detecting objects using Haar Cascade Classifier," 18 12 2019. [Online]. Available: https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-cascade-classifier-4585472829a9.

[8] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple," 2001.

[9] mrnugget. [Online]. Available: https://github.com/mrnugget/opencv-haar-classifier-training.

[10] OpenCV, "Cascade Classifier Training," [Online]. Available: https://docs.opencv.org/master/dc/d88/tutorial_traincascade.html.

[11] D. Jacobs, "Gradientes de imagen," Class Notes for CMSC 426, 2005.

[12] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," INRIA Rhone-Alps, Francia.

[13] "Using Histogram of Oriented Gradients (HOG) for Object Detection," [Online]. Available: https://iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/.

[14] C. A. Poynton , "Digital Video and HDTV: Algorithms and Interface".

[15] "Histogram of oriented gradients," [Online]. Available: https://www.learnopencv.com/histogram-of-oriented-gradients/.

[16] “cv2 resize interpolation methods,” [Online]. Available: https://chadrick-kwag.net/cv2-resize-interpolation-methods/.

[17] “Image Interpolation,” [Online]. Available: https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20162017/LectureNotes/IP/Images/ImageInterpolation.html.

[18] “Descubre Arduino,” [Online]. Available: https://descubrearduino.com/.

[19] “ArduPilot,” [Online]. Available: https://ardupilot.org/copter/docs/common-pixhawk-overview.html.

[20] E. P. Barrero, “Aceleración del algoritmo de Viola-Jones mediante rejillas de procesamiento masivamente paralelo en el plano focal,” 2015.