



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
INFORMÁTICA

MÁSTER UNIVERSITARIO EN VISIÓN ARTIFICIAL

TRABAJO FIN DE MÁSTER

Sigue persona con drone

Autor: Aitor Martínez Fernández

Tutor: José María Cañas Plaza

Cotutor: Julio Vega

Curso académico 2019/2020

Agradecimientos

Muchas gracias a todos!

Resumen

Resumen

Índice general

Índice de figuras	V
Índice de tablas	VI
1. Introducción	2
2. Objetivos	3
3. Estado del arte	4
4. Herramientas utilizadas	5
4.1. Kibotics	5
4.2. Tensorflow	6
4.2.1. Object Detection API	7
4.3. Opencv	7
4.4. Google Colaboratory	9
4.5. DJI Tello	9
4.6. Mixamo	10
4.7. Blender	11
4.8. LabelMe	11
5. Drone real	13
5.1. Desarrollo del <i>driver</i>	13
5.1.1. Convertir el driver de Python 2.7 a Python 3.x	14
5.1.2. Poca fiabilidad en el envío de mensajes del <i>driver</i> al drone	14
5.1.3. Si pasan 5 segundos sin recibir mensajes el drone se desactiva	14
5.1.4. Las velocidades de entrada están en cm/s y grados/s	15
5.2. Detección de la persona	15
5.3. control PID	16
6. Drone simulado	18

7. Conclusiones	19
Bibliografía	20

Índice de figuras

4.1. Ejercicios Kibotics	5
4.2. Ejercicio con Scratch	6
4.3. Funciones de OpenCV	8
4.4. Google colab	9
4.5. Drone Tello	10
4.6. Ejemplos de Mixamo	10
4.7. Blender	11
4.8. Etiquetado de LabelMe	12
5.1. Persona seleccionada	17
5.2. Torso detectado	17

Índice de tablas

5.1. Cambios de Python 2 a Python 3 efectuados	14
--	----

Acrónimos

API Application Programming Interface.

CUDA Compute Unified Device Architecture.

FPN Feature Pyramid Networks.

FPS Frames Per Second.

GUI Graphical User Interface.

IA Inteligencia Artificial.

IOU Interseccion Over Union.

NMS Non-Maximum Suppression.

OpenCV Open Source Computer Vision Library.

SSD Single Shot MultiBox Detector.

TFM Trabajo de Fin de Máster.

VA Visión Artificial.

Capítulo 1

Introducción

introduccion

Capítulo 2

Objetivos

Objetivos

Capítulo 3

Estado del arte

Estado del arte

Capítulo 4

Herramientas utilizadas

EL objetivo de este trabajo es preparar la infraestructura para desarrollar prácticas de visión con drones en la plataforma Kibotics[1] tanto con drone real como simulado. Para ello se han necesitado las siguientes herramientas.

4.1. Kibotics

Kibotics[1] es una plataforma de enseñanza de robótica infantil desarrollada por las Asociación de robótica JdeRobot¹

La batería de ejercicios(figura 4.1) que incluye el entorno *Kibotics*[1] se simula usando el simulador robótico *Websim*. Se trata de un simulador diseñado para el aprendizaje de conceptos básicos de programación de robots especialmente para niños.

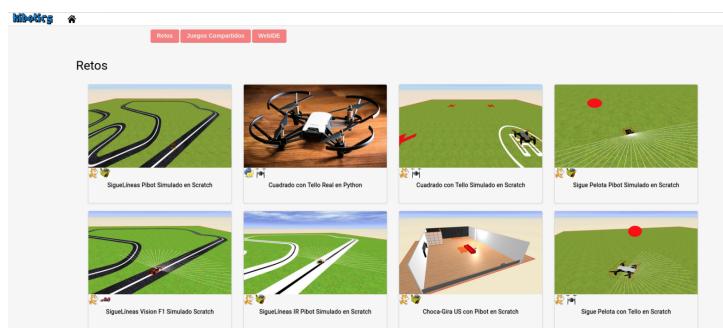


Figura 4.1: Ejercicios Kibotics

El simulador permite que los usuarios puedan programar fácilmente los movimientos

¹<https://jderobot.github.io>

de los robots, ya que simplemente tienen que acceder a la información que recogen sus sensores y enviar las órdenes precisas a los actuadores del robot. Estas ordenes se deben programar, en *Python* o *Scratch* (figura 4.2), dentro del editor que incorpora la interfaz de *Websim*.

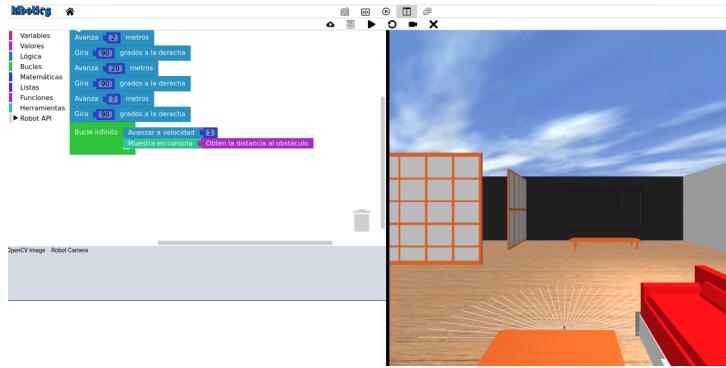


Figura 4.2: Ejercicio con Scratch

Se ha usado para la parte simulada.

4.2. Tensorflow

Es una plataforma de código abierto *end-to-end* para el *Machine Learning* que fue liberada bajo licencia de *Apache 2* a finales de 2015 y que está disponible en *github* [2]. Fue desarrollada por el equipo de investigación en *Machine Learning* “*Google Brain*” en *C++* y *Python* y es usada en multitud de productos y servicios de Google como *Gmail* o *Google Translation*. Google ofrece en su plataforma *Cloud* ejecutar *Tensorflow* en *Tensor Processing Unit* (TPU), un nuevo tipo de procesadores en *Cloud* optimizados para ejecutar Inteligencia Artificial (IA). *TensorFlow* está orientado a problemas de *Deep Learning* y permite entrenar y construir redes neuronales.

TensorFlow puede correr tanto en CPUs como en GPUs (haciendo uso de *Compute Unified Device Architecture (CUDA)*). Está disponible en *Linux* de 64 bits, *MacOS*, y plataformas móviles que incluyen *Android* e *iOS*, además ha incluido soporte para *Javascript*, por lo que puede ser usado en cualquier navegador. Actualmente es el entorno más popular en *Deep Learning*.

Puede ejecutar de forma rápida y eficiente gráficos de flujo. Un gráfico de flujo está formado por operaciones matemáticas representadas sobre nodos, y cuya entrada y salida

es un vector multidimensional o tensor de datos, por este motivo recibe el nombre de *TensorFlow*.

Las ventajas de este software se extienden a muchas disciplinas a parte de la tecnología TIC. Se emplea en imágenes médicas para la detección de tumores por ejemplo, también se usa en la detección y combinación de estilos artísticos en la pintura, etc.

En este TFM se emplean la versión 1.15.0 y 2.2.0 de *Tensorflow* para *Python* y la versión 2.0.1 para *Javascript*.

4.2.1. Object Detection API

*Object Detection API*² es un *framework* de código abierto construido sobre *TensorFlow* que facilita la construcción, el entrenamiento y el despliegue de modelos de detección de objetos desarrollado por *Google*. Todas las redes de detección preentrenadas que se pueden descargar desde la web de *Tensorflow* han sido entrenadas con este API.

Hasta Julio de 2020, cuando publicaron la primera versión con soporte para *Tensorflow* 2.x, solo tenía solo soporte para la versión 1.x.

4.3. OpenCV

*OpenCV*³ es una librería de código abierto desarrollada por *Intel* y publicada bajo licencia de BSD. Esta librería implementa gran variedad de herramientas para la interpretación de la imagen. Sus siglas provienen de los términos anglosajones “*Open Source Computer Vision Library*”, y tal y como se puede deducir es una librería destinada a aplicaciones de visión por computador en tiempo real. Puede ser empleada en MacOS, Windows y Linux, y existen versiones para *C#*, *Python* y *Java*, a pesar de que originalmente era una librería en *C/C++*. Además hay interfaces para *Ruby*, *Python*, *Matlab* y otros lenguajes.

En 2017 añadieron **soporte para *Javascript*** mediante *emscripten*⁴ que es un software que permite convertir bibliotecas escritas en *C++* en binarios para *Javascript*. Esto permite tener muchas de las funcionalidades desarrolladas en *C++* en el navegador.

OpenCV implementa algoritmos para técnicas de calibración, detección de rasgos,

²https://github.com/tensorflow/models/tree/master/research/object_detection

³<https://opencv.org/>

⁴<https://emscripten.org>

CAPÍTULO 4. HERRAMIENTAS

rastreo, análisis de la forma, análisis del movimiento, reconstrucción 3D, segmentación de objetos y reconocimiento, etc. Los algoritmos se basan en estructuras de datos flexibles acopladas con estructuras IPL (*Intel Image Processing Library*), aprovechándose de la arquitectura de Intel en la optimización de más de la mitad de las funciones. Incorpora funciones básicas para modelar el fondo, sustraer dicho fondo y generar imágenes de movimiento MHI (*Motion History Images*). Además incluye funciones para determinar dónde hubo movimiento y en qué dirección.

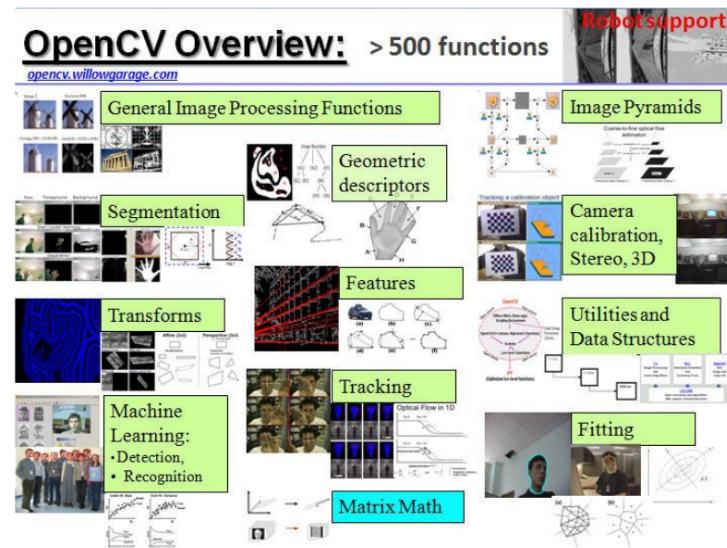


Figura 4.3: Funciones de OpenCV

Fue diseñado para tener una alta eficiencia computacional, está escrito en C y puede aprovechar las ventajas de los procesadores multinúcleo. Contiene más de 2500 funciones que abarcan muchas áreas de la visión artificial. También tiene una librería de aprendizaje automático (MLL, *Machine Learning Library*) destinada al reconocimiento y agrupación de patrones estadísticos.

Desde su aparición *OpenCV* ha sido usado en numerosas aplicaciones. Entre las cuales se encuentra la unión de imágenes de satélites y mapas web, la reducción de ruido en imágenes médicas, los sistemas de detección de movimiento, la calibración de cámaras, el manejo de vehículos no tripulados, el reconocimiento de gestos, etc. *OpenCV* es empleado también en reconocimiento de música y sonido, mediante la aplicación de técnicas de reconocimiento de visión en imágenes de espectrogramas del sonido.

Hay una gran cantidad de empresas y centros de investigación que emplean estas técnicas como IBM, Microsoft, Intel, SONY, Siemens, Google, Stanford, MIT, CMU,

Cambridge e INRIA.

En este proyecto se hace uso de la versión *OpenCV 4.2* de *Python* y 3.3.1 de *Javascript*.

4.4. Google Colaboratory

*Google Colaboratory o colab*⁵ (figura 4.4) es un servicio de *Google* que permite ejecutar código *Python* desde el navegador basándose en la tecnología *Jupyter notebook*⁶. En el servidor hay un interprete de *Python* que recibe el código que se escribe en el navegador y devuelve el resultado para mostrarlo en su celda correspondiente.

La mayor ventaja de este servicio con respecto a ejecutar el propio código en local es que se tiene acceso tanto a *GPUs* como a *TPUs* proporcionadas por *Google*, es decir, se pueden hacer cálculos muy costosos computacionalmente en ordenadores básicos.

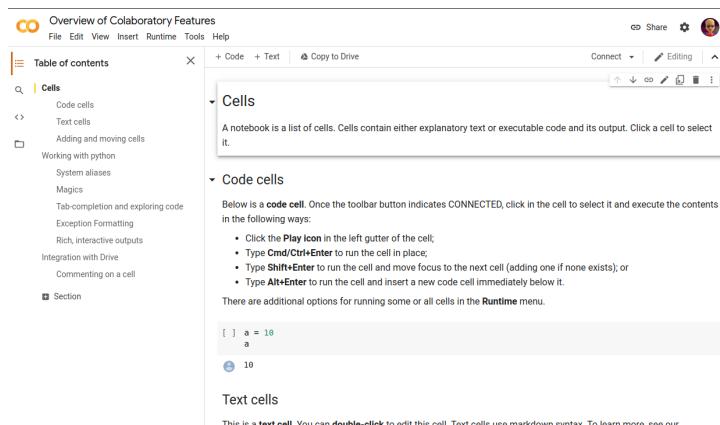


Figura 4.4: Google colab

4.5. DJI Tello

Es un pequeño drone (figura 4.5) fabricado por la empresa DJI⁷. Tiene las ventajas de ser programable mediante un API además de tener un precio muy razonable, en torno a 100, lo que le permite ser una muy buena opción de cara a la enseñanza.

⁵<https://colab.research.google.com>

⁶<https://jupyter.org>

⁷<https://www.dji.com/es>



Figura 4.5: Drone Tello

4.6. Mixamo

Mixamo[3] es una empresa del grupo *Adobe Systems* que desarrolla y vende servicios basados en la web para la animación de personajes en 3D. Utilizan métodos de *Machine Learning* para automatizar los pasos del proceso de animación de personajes, incluyendo desde el modelado 3D hasta la animación 3D. Desde su web ofrecen modelos y animaciones gratuitos (4.6).

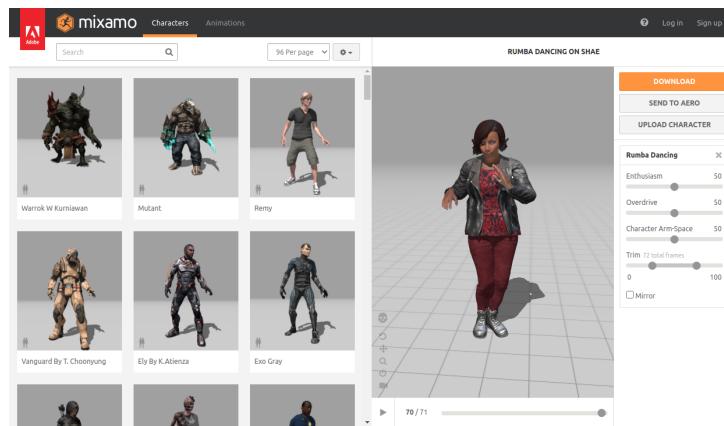


Figura 4.6: Ejemplos de Mixamo

En este proyecto se ha usado para obtener los modelos de las personas para la versión simulada.

4.7. Blender

*Blender*⁸ (figura 4.7) es un programa dedicado especialmente al modelado, iluminación, renderizado, animación y creación de gráficos tridimensionales.

Inicialmente fue distribuido de forma gratuita pero sin el código fuente. Posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, macOS, GNU/Linux (incluyendo Android), Solaris, FreeBSD e IRIX.

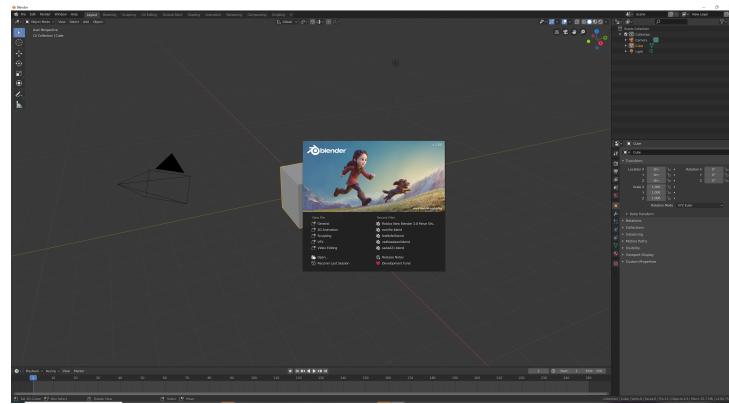


Figura 4.7: Blender

En este TFM se ha usado para modificar los modelos extraídos de *Mixamo*

4.8. LabelMe

LabelMe[4] es un proyecto creado por el Laboratorio de Ciencias de la Computación e Inteligencia Artificial del MIT (CSAIL) que proporciona un conjunto de datos de imágenes digitales con anotaciones. Además también tiene una herramienta de etiquetado^{4.8} de *datasets* que permite subir tu propio *dataset* a un servidor para poder etiquetar cada imagen desde el navegador.

⁸<https://www.blender.org>

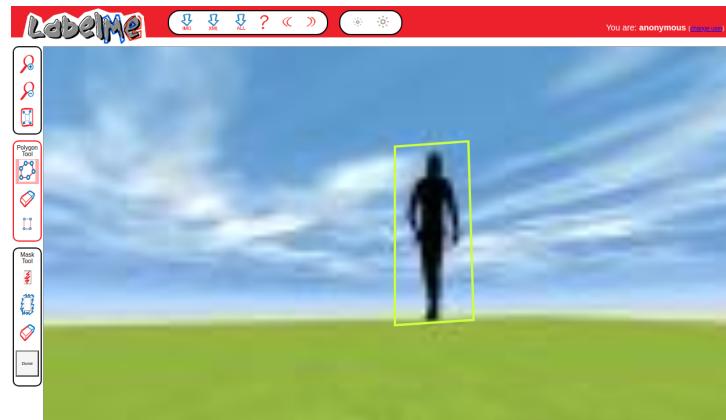


Figura 4.8: Etiquetado de LabelMe

En este proyecto se ha usado etiquetar el *dataset* para el reentrenamiento de la red para el simulador.

Capítulo 5

Drone real

En esta sección se va a explicar lo que se ha necesitado para realizar un sigue persona con un drone real con el objetivo de ser incluido en la plataforma *Kibotics* para enseñanza de robótica.

Para ello se ha tenido que perfeccionar el driver aportado por el fabricante[5]. Además como va a ser usado por niños, se ha tenido que hacer mas amigable el interfaz, por ejemplo, en vez de tener que programar la detección de un objeto de un color indicado, se ha creado un método que recibe como parámetro el nombre del color y devuelve el cuadro que rodea dicho objeto.

5.1. Desarrollo del *driver*

El *driver* desarrollado para la plataforma **Kibotics** parte del proporcionado por el fabricante del drone. Añadiendo una serie de cambios por las limitaciones detectadas, además de necesitar simplificar el uso ya que está orientado a enseñanza infantil.

Este *driver* permite recibir imágenes de la cámara del drone, informaciones básicas como batería restante o tiempo de vuelo. Además permite controlar el drone tanto en velocidad, como en posición (avanza x metros, gira x grados,...),...

Los problemas encontrados han sido los siguientes:

- *Driver* necesario en **Python 3.x** pero está escrito en **Python 2.7**.
- Poca fiabilidad en el envío de mensajes del *driver* al drone.
- Si pasan 5 segundos sin recibir mensajes el drone se desactiva.

- Las velocidades de entrada están en **cm/s** y **grados/s**.
- Va a ser usado por niños así que conviene simplificar su uso.

5.1.1. Convertir el driver de Python 2.7 a Python 3.x

Esto ha sido fácil porque casi todo el código funcionaba en **Python 3**, las únicas diferencias eran la manera de importar los módulos que varía de entre las dos versiones y que en **Python 2** los bytes se representan como cadenas de texto(*String*) y en **Python 3** son del tipo bytes.

Tipo de cambio	Python 2	Python 3
Representación de bytes	' '	b' '
Importar desde mismo repositorio	import module	import .module

Tabla 5.1: Cambios de **Python 2** a **Python 3** efectuados

En la tabla ?? se pueden ver los cambios efectuados.

5.1.2. Poca fiabilidad en el envío de mensajes del *driver* al drone

La comunicación con el Drone consiste en enviar un mensaje y recibir una respuesta que indica que lo ha recibido, o en el caso de pedirle algún dato, como la batería restante el dato en si.

EL problema radica en que o se pierde el mensaje que se envía o se pierde la respuesta. Por lo que se ha implementado un mecanismo de reenvío de mensajes. Si no se recibe respuesta se vuelve a enviar el mismo mensaje hasta tener respuesta o se alcance el número máximo de reintentos.

5.1.3. Si pasan 5 segundos sin recibir mensajes el drone se desactiva

Este problema ha sido abordado añadiendo un hilo de **Python** que se encargue de enviar velocidades cada **25ms**. De esta manera evitamos que el Drone se desactive y además conseguimos un mejor comportamiento en el control en velocidad que usando la mecánica de reenvío de mensajes. No importa que se pierdan mensajes porque enseguida se envía otro con la velocidad actualizada.

5.1.4. Las velocidades de entrada están en cm/s y grados/s

La intención ha sido que se use el sistema internacional en todo momento y en este caso son **m/s** y **rad/s**. Esto es tan fácil como realizar las conversiones pertinentes en las funciones para establecer las velocidades.

5.2. Detección de la persona

Para la detección de la persona se ha utilizado una red de detección **SSD Mobilenet v2 FPN lite** preentrenada con el dataset coco.

Esta red tiene 3 componentes principales:

- *Feature Pyramid Networks (FPN) lite*[6]. Permite extraer características de la imagen con indiferencia del tamaño del objeto
- *Mobilenet v2*. Es una red convolucional de extracción de características.
- *Single Shot MultiBox Detector (SSD)*[7]. Comprueba si hay cada clase en las regiones de interés seleccionadas y permite un mayor ajuste de dicha región.

Se ha elegido dicha red porque funciona de manera fiable y rápida. Para poder trabajar con ella en se usa **Tensorflow 2.0**. Además se ha tenido que desarrollar un recubrimiento para poder usar la red de manera fácil. Esta clase desarrollada permite cargar la red indicada por configuración ya sea un modelo de **Tensoflow** o un grafo. Además agrega un postprocesado de la información de salida de la red:

- convirtiendo en *numpy arrays* los datos desde tensores
- Desechando los resultados con una puntuación menor al límite indicado por configuración.
- Filtrando los resultados para quedarse con las clases buscadas, en este caso personas.
- También se convierten los tamaños de las regiones de interés de porcentaje a píxeles.

Además se ha creado también otro nivel de abstracción superior para la fuente de la imagen para ayudar a su uso, así mediante configuración se indica si la fuente es el drone, una webcam, un directorio de imágenes o un vídeo y la fuente en sí y no hay que preocuparse de procesar la imagen para convertirla en RGB en caso de que sea la fuente

no lo sea por ejemplo. La red recibirá siempre la imagen igual provenga del origen que provenga.

La red al final devuelve 3 *arrays*, el primero con la clase a la que pertenecen las predicciones , el segundo con las regiones de interés o *bounding boxes* (x mínima, y mínima, x máxima, y máxima) y el tercero con las puntuaciones.

5.3. control PID

El control PID es un bucle de control que calcula la siguiente operación en cada iteración.

$$u(t) = K_p e(t) + K_i \int_t^0 e(t') dt' + K_d \frac{de(t)}{dt}$$

Donde **e** es el error con respecto al objetivo y las **K** constantes que deben calcularse experimentalmente.

Una vez que se procesa la imagen recibida por el drone y se tienen las predicciones, se selecciona la persona con mayor puntuación y de su *bounding box* se extrae posición central, el área y la posición superior.

Para controlar el drone se va a hacer con tres velocidades, avance, giro y movimiento vertical. Para ello se utiliza un control PID (en este caso debido al funcionamiento del drone solo proporcional y derivativo) para cada una de las velocidades.

En el caso de la velocidad de avance, se toma como referencia el área, para el giro la posición del centro en las coordenadas x de la imagen y para el la altura se usa la posición superior del *bounding box*. En el caso de la velocidad vertical se ha decidido usar como referencia la posición superior del *bounding box* posicionándolo en torno a un 10 % de la parte superior de la imagen con el objetivo que se vea la cara de la persona. se probó también la manera fácil que es centrar verticalmente la persona en la imagen, pero puede ocurrir que solo se tenga una parte de la persona y esté centrado. como se puede ver en las imágenes 5.1 y 5.2 en ambos casos el centro del cuadrado está muy próximo al centro vertical de la imagen, pero no por ello estar bien.

En cambio tomando como referencia el punto más alto del cuadrado se tiende a obtener la imagen 5.1 que además facilita la detección de la persona.

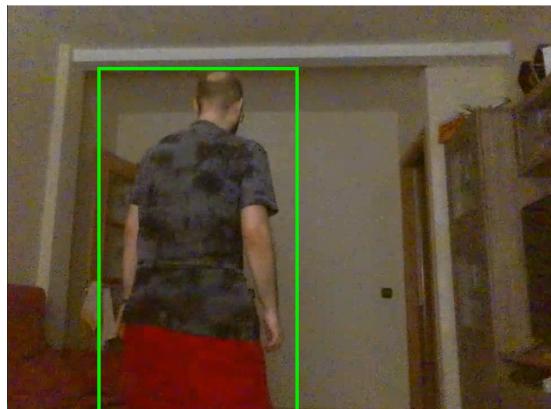


Figura 5.1: Persona seleccionada



Figura 5.2: Torso detectado

Capítulo 6

Drone simulado

sim

Capítulo 7

Conclusiones

conclusiones

Bibliografía

- [1] Asociación de robótica e inteligencia artificial JdeRobot. Kibotics. <https://kibotics.org>.
- [2] Tensorflow. <https://github.com/tensorflow/tensorflow>.
- [3] Adobe Systems Incorporated. Mixamo. www.mixamo.com.
- [4] Labelme. <http://labelme.csail.mit.edu/Release3.0/>.
- [5] DJI. Dji tello driver python. <https://github.com/dji-sdk/Tello-Python>.
- [6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.
- [7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.
- [8] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.
- [9] Ignacio Condés Menchén. *Deep Learning Applications for Robotics using TensorFlow and JdeRobot*. PhD thesis, Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad Rey Juan Carlos, 2018. [Accedido 24 de Junio de 2019].
- [10] Marcos Pieras Sagardoy. *Visual people tracking with deep learning detection and feature tracking*. PhD thesis, Escuela Técnica Superior de Ingeniería de Telecomunicación, Universidad Rey Juan Carlos, 2017. [Accedido 24 de Junio de 2019].