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

Índice de tablas

Acrónimos

API Application Programming Interface.

FPN Feature Pyramid Networks.

GUI Graphical User Interface.

SSD Single Shot MultiBox Detector.

VA Visión Artificial.

Capítulo 1

Introducción

introduccion

Capítulo 2

Objetivos

Objetivos

Capítulo 3

Estado del arte

Extado del arte

Capítulo 4

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[1]. 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.

4.1. 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.

4.1.1. 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 4.1: Cambios de **Python 2** a **Python 3** efectuados

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

4.1.2. 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.

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

Este problema ha sido abordado añadiendo un hilo de **Python** que se encargue se 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.

4.1.4. 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.

4.2. 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*[2]. 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)[3]*. 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 la regiones de interés o *bounding boxes* (x mínima, y mínima, x máxima, y máxima) y el tercero con las puntuaciones.

4.3. 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 4.1 y 4.2 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 4.1 que además facilita la detección de la persona.

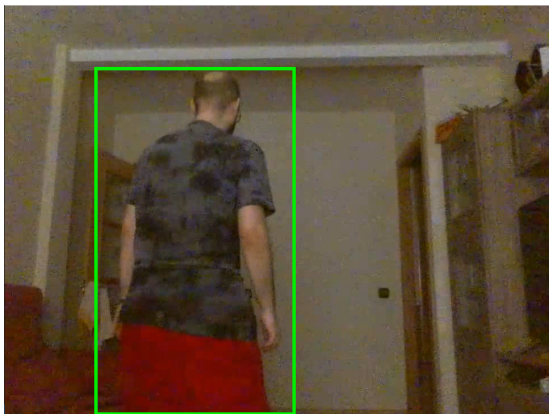

Figura 4.1: Persona seleccionada

Figura 4.2: Torso detectado

Capítulo 5

Drone simulado

sim

Capítulo 6

Conclusiones

conclusiones

Bibliografía

[1] DJI. DJI Tello driver python. `https://github.com/dji-sdk/Tello-Python`.

[2] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[3] 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.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[5] 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].

[6] 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].