UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

TRABAJO DE FIN DE GRADO

TÍTULO DEL TFG: Análisis de Imagen aplicada a la gestión de aeropuertos

TITULACIÓN: Grado en Ingeniería de sistemas Aeroespaciales
Grado en Ingeniería de sistemas de Telecomunicaciones

AUTOR: Martin Fernández Marcellán

DIRECTORES: Francesc Tarrés Ruiz
Joan Moltó Rando

FECHA: 16 de junio de 2021

Título: Análisis de Imagen aplicada a la gestión de aeropuertos
Autor: Martin Fernández Marcellán

Directores: Francesc Tarrés Ruiz
Joan Moltó Rando

Fecha: 16 de junio de 2021

Resumen

Como el título indica, el trabajo trata de la aplicación de técnicas de procesado de imágenes en la gestión de los aeropuertos. Concretamente, se centra en un cometido: Reconocimiento de aviones a partir de la detección de sus matriculas.
El proceso para obtener la matricula de un avión comienza obviamente por tomar la foto que se va a procesar. Para ello se ha usado un detector de movimiento que hace fotos a los aviones que pasan por delante. Esas fotos son enviadas a un servidor que sirve de almacén de fotos. Una vez las fotos estén en el servidor, el cliente las descarga por grupos y las procesa con un algoritmo de detección de objetos de deep learning. Este algoritmo encuentra e identifica los aviones que hay en las fotos y con esta información escoge las tres mejores fotos.
Las fotos seleccionadas son nuevamente procesadas, esta vez por un sistema de reconocimiento óptico de caracteres (OCR) que usa técnicas de deep learning. Como resultado se obtiene la transcripción de los textos que aparecen en la foto, estando entre ellos la matricula. Para distinguir la matricula de otros textos que no son relevantes en este caso, se aplican varios filtros de postprocesado. Para terminar, el sistema decide de que matricula se trata comparando los resultados del OCR con una lista de todas las matriculas posibles. Usando este sistema se ha obtenido un porcentaje de acierto de hasta un 91 % en un grupo de 66 imágenes.
Un sistema como este se puede utilizar para llevar el control de las operaciones de un aeropuerto no controlado o para ubicar una aeronave en un aparcamiento o hangar a través de las fotos tomadas por las cámaras de seguridad.

Title : Image analysis applied to airport management
Author: Martin Fernández Marcellán

Advisors: Francesc Tarrés Ruiz
Joan Moltó Rando

Date: June 16, 2021

Overview

As the title suggests, the project is about applying image processing techniques to airport management. Concretely, about the recognizement of aircraft by detecting their registration.
The first thing to do to get the registration of an aircraft is taking a photo of it. In order to do that, a motion detector takes photos of the aircraft when the aircraft is in front of the camera. The photos are sent to a server that is used as a storage unit, then the client downloads groups of photos to begin the processing stage. The first step is an object detection deep learning algorithm which finds and identifies all the objects that appear in each photo. Knowing the position and the size of the airplanes that appear in each photo, the client chooses the best three photos.
The selected photos advance to the next processing step which consists in a deep learning optical character recognition (OCR) algorithm. The result of processing photos with the OCR is the transcription of all the texts that appear in the image. The only interesting texts for this project are the registrations, so post processing filters are in charge of discarding every text that does not look like a registration. Finally, the system decides which registration appears in the image by comparing the results of the OCR to a list of possible registrations. By using this system, a 91% accuracy has been reached with a group of 66 images.
The main aims of the system are the control of the take off and landing operations in no controlled airports and the control of the parkings and hangars to locate any aircraft in the airport using the security camera images.

Amona eta Güelitari

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

INTRODUCCIÓN

El análisis de imágenes consiste en la extracción de información de los datos proporcionados por sensores específicos para esta tarea como son las cámaras de fotos o cámaras de infrarrojos. La idea de este proyecto es presentar dos posibles usos de este método de obtención de información aplicada a la gestión de aeropuertos y aeródromos. Dichos usos consisten en identificar los aviones que realizan las operaciones de despegue y aterrizaje del aeropuerto y en determinar que avión ocupa cada hangar o puesto de aparcamiento. El objetivo de ambas aplicaciones se logra a partir del reconocimiento óptico de las matriculas.

A lo largo del documento se concretarán ambos usos centrándose sobre todo en el primero, del que se dispone un prototipo. Ambas tareas pretenden sustituir tareas mecánicas y sencillas que resultan demasiado simples como para que un ser humano invierta su tiempo en realizarlas.

Para realizar las dos funciones, se hace uso de algoritmos de *deep learning* y el prototipo antes mencionado utiliza un microcontrolador de la marca Raspberry Pi. Por lo tanto, antes de comenzar con la explicación de los diferentes elementos que componen la cadena de procesado, se presentan varios conceptos como inteligencia artificial o *deep learning* en el primer capitulo ”Presentación” 1 para facilitar la comprensión del resto del texto. Una vez definidos los conceptos imprescindibles, los capítulos dos y tres sirven de explicación teórico práctica de los algoritmos utilizados en el trabajo: Algoritmos de detección de objetos 2 y algoritmos de reconocimiento óptico de caracteres u OCR 3.

A continuación, el capítulo cuarto corresponde a la presentación de la cadena de procesos por los que pasa una foto que va a ser procesada por el sistema de reconocimiento de matrículas 4. El quinto capítulo tratará de la obtención de imágenes usando para ello un microcontrolador Raspberry Pi 3 y su posterior envío al servidor del cual obtendrá las imágenes el software encargado del procesado 5.

Pese a que en la etapa de post procesado se filtran muchos datos aumentando la eficacia del sistema, es necesario diseñar un método que asegure una correcta lectura de matrícula. Dicho método es el tema central del sexto capítulo 6.

El séptimo capítulo presenta una estimación del impacto de este sistema en dos ámbitos: Primero, en el aeronáutico aportando ejemplos concretos y cifras para acreditar la necesidad de estos, y a continuación el medioambiental midiendo el potencial impacto del sistema 7. Para terminar, el último capítulo tratará de cerrar el proyecto aportando conclusiones y posibles mejoras o aplicaciones futuras 8.

CAPÍTULO 1. PRESENTACIÓN

Este capítulo consta de tres secciones diferentes: Primero se exponen ciertos conceptos clave para facilitar la comprensión global del texto, después se hace una breve introducción al microcontrolador utilizado en el prototipo y para finalizar el capitulo, se ponen en relieve las diferencias entre detectar matriculas de coches y de aviones haciendo hincapié en las dificultades particulares del caso aeronáutico.

1.1. Conceptos clave

1.1.1. Algoritmo

Tomando como referencia la definición de algoritmo que da la RAE: “Un conjunto finito y ordenado de operaciones que permite hallar la solución a un problema” [1], resulta evidente que para procesar de manera igual múltiples fotos, la presencia de algoritmos en dicho proceso es inevitable. Existen infinidad de algoritmos y se pueden agrupar en infinidad de subgrupos diferentes en función de la característica que se tenga en cuenta o se le dé más importancia.

Una característica que se puede utilizar para dividir los algoritmos en dos subgrupos es si son o no capaces de aprender a realizar las tareas de manera autónoma. Se ha seleccionado esta característica porque a lo largo del proyecto se usan varios algoritmos que sí son capaces de aprender autónomamente y se ha creído conveniente centrar la explicación previa en estos ya que conceptos como *machine learning* y *deep learning* acostumbran a dar lugar a equívocos o confusiones.

Normalmente la confusión es causada por no saber el orden de jerarquía que existe entre estos: El *deep learning* es un disciplina del *machine learning* y a su vez el *machine learning* es un campo de la inteligencia artificial como se puede ver en la figura 1.1:

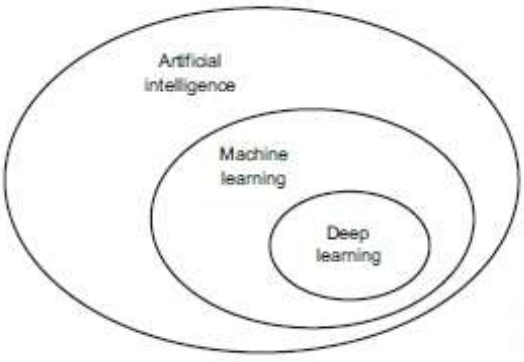


Figura 1.1: Jerarquía entre inteligencia artificial, *machine learning* y *deep learning*. Fuente: [2]

1.1.2. Inteligencia artificial

Por lo tanto hay que comenzar por definir qué es la inteligencia artificial para luego poder concretar a qué áreas de esta se refieren los algoritmos que conciernen. La inteligencia artificial surge en la década de 1950 del deseo de automatizar tareas intelectuales realizadas hasta el momento por humanos. En este campo entran ambos tipos de algoritmos según la distinción hecha anteriormente. Hasta la década de 1980 la mayor parte inteligencia artificial era de tipo simbólico, lo que quiere decir que solo era capaz de realizar tareas si éstas estaban previamente escritas en el código. Un ejemplo de inteligencia artificial simbólica son los primeros programas que fueron capaces de jugar al ajedrez a partir de una serie de instrucciones concretas que indican al programa que movimiento realizar dependiendo de la situación.

1.1.3. Machine learning

Alrededor de las décadas de 1980 y 1990 se desarrollaron los primeros algoritmos que eran capaces de aprender a resolver ciertos problemas. Esto supuso un cambio radical en la inteligencia artificial e hizo que la mayoría de algoritmos que se crean sean capaces de aprender o lo que es lo mismo, que sean de *machine learning*. Ese progreso supone un cambio radical a la hora de diseñar algoritmos ya que hasta el momento los algoritmos de inteligencia artificial eran programas a los que se les introducían unas reglas y unos datos para obtener una respuesta concreta de qué hacer.

Con la llegada del *machine learning*, lo que el algoritmo recibe son los datos y las soluciones asociadas a estos para que el programa genere unas reglas que permitan solucionar problemas parecidos. Siguiendo con el ejemplo del ajedrez, se introducen situaciones de juego con su mejor movimiento para que el programa aprenda las reglas del ajedrez y las mejores maneras de jugarlo de tal manera que luego sea capaz de jugarlo autónomamente. Las reglas que genera el algoritmo son estructuras estadísticas basadas en los datos y soluciones que ha recibido, sin embargo, estas estructuras estadísticas no son comparables con las que se pueden obtener con un análisis estadístico de un problema. La principal diferencia radica en la ingente cantidad de datos que es capaz de gestionar un sistema de *machine learning*, por ejemplo, en algoritmos dedicados al análisis de imágenes se utilizarían miles de imágenes de miles de píxeles para entrenar el algoritmo; estas cantidades de datos son demasiado grandes para la estadística clásica.

El mayor cambio respecto a los algoritmos de inteligencia artificial anteriores es que a estos se les introducen como parámetros de entrada los elementos que tiene que analizar y la solución que se espera obtener, pero para que un algoritmo “aprenda” a realizar la tarea que se le encomienda de manera correcta es imprescindible que haya una manera de medir su progreso. Para esto último es especialmente importante definir que una función que mida correctamente la distancia entre los resultados parciales y la respuesta correcta y que sirva de realimentación para mejorar progresivamente los resultados.

Al fin y al cabo, los algoritmos de *machine learning* tratan de encontrar la mejor manera de representar los datos para poder transformarlos de manera correcta. Un ejemplo de transformación de datos es precisamente el objetivo del proyecto: Transformar un conjunto de píxeles en un texto, que en este caso es la matricula de un avión. En cuanto a la correcta representación de los datos, un ejemplo claro es el de un algoritmo dedicado a

encontrar objetos azules en imágenes. Las imágenes está compuestas por píxeles y el valor de estos puede ser representado de dos maneras; el formato RGB (Rojo, Verde, Azul) y el formato HSV (Matiz, Saturación, Valor). Para este algoritmo en cuestión, la representación RGB es la correcta ya que uno de sus componentes muestra el nivel de azul que tiene cada píxel, facilitando así la detección de zonas azules en la imagen.

1.1.4. Deep learning

El *deep learning* consiste en usar múltiples capas que son capaces de representar los datos de maneras diferentes. Dichas capas se apilan una detrás de la otra de manera que cuanto más profundo (deep) se encuentra la capa más relevante es la representación que hace. Por lo que el "deep" de *deep learning* no se trata de un conocimiento más profundo si no que del número de capas que tiene la red neuronal.

Con estos conocimientos se puede hacer una composición esquemática de cómo funcionan los algoritmos de *deep learning*: Un conjunto de datos recorre varias capas que transforman estos datos y los representan de diferentes maneras según marcan los pesos de cada capa. Una vez recorridas todas las capas se obtiene una predicción la cual se compara con el resultado esperado y a partir de esta comparación se calcula el valor de la "loss function" o función de aprendizaje. Con este valor de la función de aprendizaje,un optimizador calcula los nuevos pesos de cada capa completando la realimentación. El objetivo de esto es que el valor de la función de aprendizaje (el error que cometen las predicciones del algoritmo) se reduzca gracias a la realimentación.

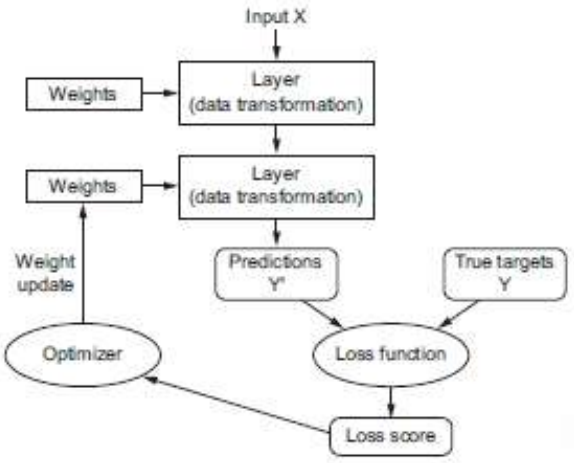


Figura 1.2: Estructura de la realimentación de un algoritmo de deep learning

1.2. Raspberry Pi

Para la obtención de imágenes se pretenden usar imágenes tomadas por cámaras de seguridad y una cámara situada en la plataforma. En el prototipo diseñado, el dispositivo situado en plataforma no solo toma fotos de los aviones, se encarga además de mandar dichas imágenes a un servidor para que estas sean procesadas y se obtenga la matrícula

deseada. Teniendo en cuenta las tareas encomendadas, se ha decidido que el dispositivo situado en la plataforma sea un ordenador de placa reducida de marca Raspberry Pi.

Raspberry Pi es una empresa dedicada a la fabricación de este tipo de ordenadores de dimensiones reducidas que permite colocarlos en cualquier punto del aeropuerto sin mayor problema ya que son fácilmente alimentados mediante baterías portátiles.

Para el prototipo se ha seleccionado el modelo “Raspberry Pi 3” con 4GB de memoria RAM, suficiente para poder obtener fotos y enviarlas al servidor por wifi. Para la captura de fotos se ha conectado un módulo de cámara de 8 megapíxeles que garantiza la calidad necesaria para que las matrículas sean legibles. Se dispone también de una carcasa que protege al ordenador y a la cámara de las inclemencias meteorológicas y permite que esta se atornille a un trípode común facilitando su colocación en zonas del aeropuerto en las que no suele haber objetos en los cuales colocar la cámara a una altura conveniente.

1.3. El problema del reconocimiento de caracteres en distintas aplicaciones

El hecho de diseñar un sistema que automatice la detección de matrículas de aviones puede sugerir que se trata de una mera adaptación de la detección de matrículas de coches que lleva años usándose en aparcamientos públicos no gratuitos o en peajes. Pero si se analizan las situaciones y condiciones en las que se realizan las detecciones, se pueden encontrar diferencias sustanciales que suponen grados de dificultad considerablemente distintos.

En el caso de las matrículas de los automóviles, la detección es sencilla por la localización de la matricula en el vehículo. El hecho de que este situada en la parte delantera y trasera facilita la detección respecto a los aviones cuyas matriculas están en ambos lados del fuselaje. La matricula delantera de los automóviles sirve para ser detectada en los accesos a aparcamientos o similares en los que el vehículo se posiciona en una posición concreta que permite que la cámara encargada de la detección pueda estar enfocando a una región muy concreta en la que se fuerza que a que aparezcan las matrículas. La matricula trasera es sumamente útil para casos como los radares ya que obtener una foto de la parte trasera de un vehículo que excede los límites de velocidad es mucho más sencillo que obtener una foto de su lateral.

En el caso de las matrículas de los automóviles, la detección es sencilla ya que se produce en situaciones donde el vehículo es forzado a parar delante de una barrera, lo que provoca que las matrículas de todos los vehículos queden en una posición muy similar siempre. La poca varianza de posición facilita mucho la tarea ya que permite que la cámara encargada de la detección pueda estar enfocando a una región muy concreta en la que se fuerza que a que aparezcan las matrículas.

Otra gran ventaja de los automóviles es que las matrículas son un elemento que se atornilla al coche y no se pinta en su carrocería. Esto permite que las matrículas de los vehículos de carretera sean estandarizadas en su tipo y tamaño de letra, el color de las letras y del fondo sobre el cual se sitúan. Es cierto que la combinación de colores letra-fondo cambia en función del país o del tipo de vehículo, pero en todos los casos se garantiza el uso de colores que facilitan la legibilidad de las mismas mediante combinaciones de alto contras-

te, por ejemplo negro y blanco o negro y amarillo. Este código de colores permite distinguir las letras y números que componen la matrícula mediante la técnica de procesamiento de imágenes conocida como “Thresholding” [3].

La técnica de “Thresholding” es una técnica de binarización de imagen, la cual consiste en convertir una imagen multinivel (varios niveles de gris) en una de solo dos niveles: blanco y negro o “1” y “0” en imágenes con el histograma normalizado. Para realizar tal conversión, se precisa de un umbral que defina la frontera de nivel de gris que determina si un pixel se vuelve negro o blanco. Para determinar dicho umbral, el histograma es una herramienta muy útil debido a la claridad para mostrar la distribución de los niveles de gris en la imagen. Los valores habituales del umbral oscilan entre 0.3 y 0.5 para el caso de detección de matrículas aunque hay varias maneras de obtener el umbral perfecto como puede ser el “K-means” que es un método iterativo cuyo objetivo es encontrar el punto medio dejando una cantidad igual de pixeles a cada lado del umbral.

En el caso de las matrículas de aviones es muy diferente al de los automóviles. Los aviones son un medio de transporte muy diverso, los tamaños pueden variar muchísimo, por ejemplo, el Airbus A320 mide 37,6 metros de largo, tiene una altura de 11,8 metros y 34,1 metros de envergadura; el Airbus A380-900, sin embargo mide 79,4 metros de largo, tiene una altura de 24,45 metros y 79,75 metros de envergadura. La diferencia entre estos aviones es notable sin ser ninguno de ellos el más grande ni el más pequeño de los muchos y diferentes modelos que hay. Tanta diferencia de dimensiones provoca el primer problema: las matrículas están situadas a alturas muy diferentes, esto hace que no se pueda tener una cámara enfocando un lugar concreto ya que no se podría garantizar que todas las matrículas vayan a ser fotografiadas por dicha cámara. Esta tendrá que estar enfocando a una zona más amplia que le permita abarcar toda el área donde pueda haber una matrícula, lo cual supone un incremento de gasto en una cámara mejor o una pérdida de calidad en las fotos y por tanto en la detección de la matrícula. En el caso de los automóviles también hay tamaños diferentes pero las matriculas son iguales para todos ellos y su localización, pese a no ser la misma, no varía mucho.

Una vez se tenga una cámara que garantice poder fotografiar todas las matrículas sin importar su altura se presenta otro problema, los aviones la llevan pintada en el fuselaje el cual varía de color en función del dueño del avión y por lo tanto también lo hacen las letras y números que componen la matrícula. Además, no hay estandarización para el tipo, tamaño y color de las letras de las matrículas por lo que existen casos en los que es difícil detectarlas debido a que el color del fuselaje y el de las letras no son fácilmente distinguibles como pueden ser el rojo y el negro.

Expuestas las condiciones particulares que tiene la detección de matrículas de aviones, queda clara la necesidad de desarrollar un algoritmo específicamente para esta tarea. Una vez esté operativo se podría usar para otras tareas parecidas como por ejemplo llevar el control de los barcos que entran y salen de un puerto. Se trata de un caso con bastantes similitudes con el de los aviones ya que los barcos son también de gran variedad de tamaños y los colores del casco son también muy variantes al igual que el color de las letras del identificador.

CAPÍTULO 2. ALGORITMOS DE DETECCIÓN DE OBJETOS

2.1. Contexto en el proyecto

Como su propio nombre indica, el cometido de un algoritmo de detección y reconocimiento de objetos es analizar imágenes en busca de todo tipo de objetos. Un ejemplo de utilidad de estos algoritmos son los drones que son capaces de perseguir autónomamente a un ser humano.

En este proyecto, este algoritmo cumple dos funciones diferentes:

- En primer lugar, tiene que localizar todos los objetos que aparecen en la imagen mediante un rectángulo o ”bounding box” que lo rodee.
- A continuación, tiene que distinguir si aquello que pasa por la pista o por el hangar es un avión y no otra cosa carente de interés para el programa

Es particularmente importante fijar las funciones que va a realizar ya que existen varios tipos de algoritmos que se pueden denominar ”reconocedores de objetos ”. Los términos que se utilizan para definir el uso del sistema es crucial porque la diferencia entre clasificar, localizar , detectar y segmentar es muy relevante.

2.2. Clasificar, localizar, detectar y segmentar

Los sistemas de **clasificación** automática de objetos basados en deep learning, se ocupan únicamente de asociar un objeto a una fotografía. Para esto, se le enseña a identificar un numero finito de objetos y así cuando reciba una fotografía para analizarla decidirá a cual de los objetos de la lista se parece más. Normalmente, por cada foto el sistema retornará o el nombre del objeto que ha reconocido con el porcentaje que asigna a dicho objeto o un top-5 en el que mostrará las cinco opciones más probables con sus porcentajes asociados. Un sistema de clasificación que este correctamente entrenado para la tarea encomendada mostrará la respuesta correcta entre las tres primeras. En el contexto del proyecto, es necesario que el algoritmo seleccionado tenga la capacidad de clasificar objetos para cumplir el primer cometido que se le ha asignado anteriormente.

A un sistema que es capaz de clasificar objetos se le puede añadir la capacidad de **localizar** los objetos que hay en la imagen. Para localizar el objeto, el sistema encuadra los píxeles en los que detecta la presencia de dicho objeto. Como ya se ha mencionado en el segundo requerimiento, es necesario que sea capaz de localizar a los aviones para poder facilitar el post-procesado de las imágenes.

Ambas capacidades, clasificación y localización, se definen para imágenes en las que solo hay un objeto. A la localización y clasificación de objetos en una foto en la que aparecen varios se le llama **detección**. La detección es necesaria en los aeropuertos para llevar a cabo el control de los aparcamientos y hangares puesto que en estos lugares los aviones

están colocados uno detrás de otro y las cámaras de seguridad enfocan a varios a la vez. Además de los aviones podrá detectar objetos de interés como automóviles o personas.

La localización y detección se realiza mediante los rectángulos llamados *bounding box* que rodean al objeto delimitando su presencia. Esta manera de localizar los objetos es la mas sencilla pero puede resultar insuficiente para tareas en las que se requiera de mucha precisión y los objetos se encuentren muy próximos. Puede suceder que los rectángulos se superpongan enturbiando la imagen resultante dificultando su interpretación. Para estos casos se utiliza la **segmentación** que muestra donde se encuentra cada objeto delimitando su contorno usando varias rectas para ello. La segmentación no es necesaria en este caso ya que aumenta el tiempo de computación y la mejora que aporta a la hora de localizar aviones no es decisiva.

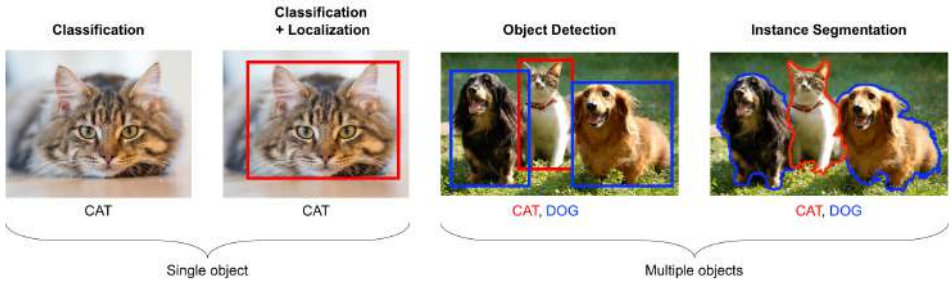


Figura 2.1: Ejemplo práctico de la diferencia entre clasificación, localización, detección y segmentación. Foto de [4]

2.3. Origen de los sistemas de detección de objetos

Hoy en día, el desarrollo de sistemas de detección y localización automática de objetos se realiza a partir de un sistema de challenges. Esto consiste en que una institución investigadora propone un reto e investigadores de todo el mundo mejoran sus sistemas para hacerse con los premios económicos y el reconocimiento público de la comunidad. En estos retos todos los competidores utilizan las mismas bases de datos para entrenar las redes neuronales, asegurando de esta manera la equidad y facilitando comparación entre competidores. De esta manera, hay varias bases de datos que al ser las más conocidas son las que más habitualmente se utilizan para entrenar los sistemas: ”Pascal” o ”COCO” para este área concreta.

Los detectores de objetos comenzaron en la década de 1990 para detectar matriculas de coches, caras y personas. Para detectar este tipo de objetos, los algoritmos recorrían la imagen buscando rasgos característicos de cada objeto como la tonalidad de la piel humana o los cambios de negro a blanco que hay en las matriculas. La búsqueda de estos rasgos característicos consiste en analizar los píxeles por grupos de un tamaño determinado en función del rasgo que se busque y comprobando si el grupo de píxeles analizado cumple los requisitos para ser considerado un rasgo característico. Esta búsqueda se realiza en la imagen original y en diferentes versiones re-escaladas de la misma puesto que el tamaño de la ventana de píxeles que se analizan es invariante para cada rasgo, la escala de la imagen tiene que variar para que sea posible detectar objetos de diferentes

tamaños. Un ejemplo típico de rasgo característico del ser humano es la cara,la cual se busca con ventanas de 24x24 o 32x32 píxeles.

El coste computacional de esta manera de localizar objetos es elevado ya que consiste en repetir varias veces el mismo proceso, para reducir el número de operaciones a realizar y para aumentar la cantidad de objetos que se pueden detectar, estos sistemas utilizan hoy en día algoritmos de deep learning.

2.4. AlexNet

El primer sistema realmente eficaz de deep learning para clasificación de imágenes data de 2012 y es conocido como ”AlexNet” [26] y consiguió reducir el error de top-5 del 25 % al 16 %. El sistema consiste en una red neuronal con once capas de profundidad de las cuales cinco son convolucionales, tres completamente conectadas y tres de ”Max Pooling”.

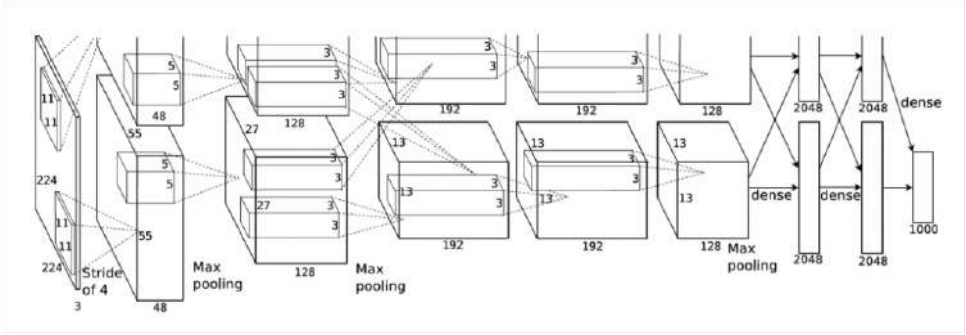


Figura 2.2: Arquitectura del sistema ”AlexNet”. Foto de [5]

- Las capas convolucionales son la forma que tienen los sistemas de deep learning para realizar la búsqueda de rasgos característicos que hacían los sistemas anteriores. Estos sistemas, en vez de re-escalar las imágenes y buscar grupos de píxeles que cumplan ciertas condiciones aplican una función local en cada capa que permite obtener características genéricas de la imagen. Estas capas funcionan de manera jerárquica por lo que cuanto más profunda sea la capa, más nivel tendrán las características extraídas, por ejemplo, las capas más cercanas a la imagen original detectaran los contornos que hay en la imagen mientras que las más profundas pueden detectar formas con geometrías especificas.

- Las capas de Max Pooling sirven para reducir las dimensiones de la red. Para esto, divide la imagen que recibe en varias secciones y busca el valor máximo entre los píxeles de cada sección construyendo una nueva imagen más pequeña con los máximos de cada sección. La división de la imagen la hace un filtro y la relación de tamaño entre el filtro y la imagen es lo que define el tamaño de la imagen resultante.

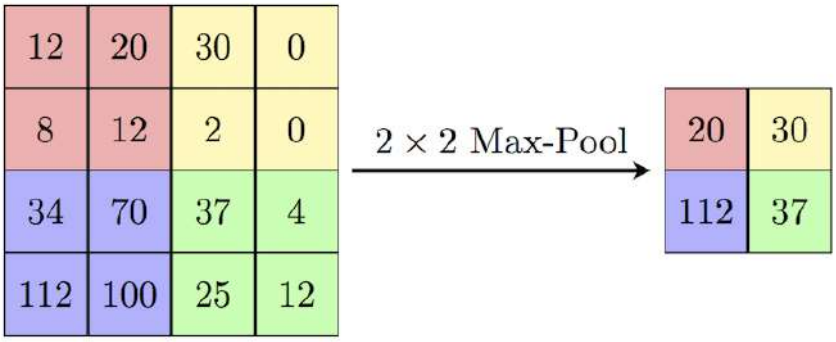


Figura 2.3: Ejemplo de Max Pooling usando un filtro 2x2 para una imagen 4x4. Foto de [6]

- Las capas totalmente conectadas sirven para relacionar los nodos de una capa con los resultados de la capa anterior. Suelen encontrarse al final de la red

Tal y como se puede observar en la figura 2.3, el número de píxeles que se procesan a la entrada de la red es de 224x224x3 = 150528 y la primera capa de convolución utiliza elementos de 11x11 dando lugar a 48 salidas por ventana. Esto supone un total de 11x11x3x48 = 17424 parámetros para aprender. De la misma manera, la segunda capa, una de max pooling, aplica filtros de 5x5 para obtener 128 salidas de cada una de las 48 obtenidas en la capa anterior. Dando lugar a un total de 5x5x48x128 = 153600 parámetros de aprendizaje diferentes.

Tal cantidad de parámetros provoca que el dispositivo que este ejecutando el sistema disponga de gran cantidad de memoria para poder almacenar todos los parámetros de la red. Por ejemplo, la primera capa de AlexNet necesita guardar 2x55x55x48 = 290400 píxeles o lo que es lo mismo, 290,4 MB de memoria.

Siguiendo el paso de AlexNet, se han desarrollado varias arquitecturas diferentes para la clasificación de objetos usando deep learning como ”ZFNet” [27] ganadora de la competición *Imagenet* del 2013 que consiguió un error de top-5 de 11.2% con una arquitectura parecida a AlexNet. También existen redes con más capas como ”VGG Net” [28] que tiene 19 capas convolucionales o redes como ”GoogleNet” [29] que usan módulos *Inception* que consisten en núcleos de convolución de diferentes tamaños. Un de los sistemas más importantes que se han desarrollado es ”Resnet” [30], una solución introducida por Microsoft que utiliza elementos de interconexión entre capas para acelerar el aprendizaje.

2.5. RetinaNet

Para este proyecto de análisis de imágenes aplicado a aeropuertos necesitamos que el sistema sea no solo de clasificación si no de detección y para ello se ha decidido usar un sistema con arquitectura RetinaNet.

El modelo RetinaNet es un modelo publicado en 2018 [7] el cual destaca por ser uno de los mejores de una sola etapa, lo que significa que la red es capaz de localizar y clasificar todos los objetos de la foto haciendo pasar la foto una única vez por su red neuronal. La principal ventaja de usar un detector de una etapa es el ahorro de tiempo de computación y uso de memoria.

La principal innovación de RetinaNet es su función de pérdida o ”Loss function” llamada *Focal Loss*. Se trata de una mejora de las pérdidas de entropía cruzada y su principal cometido es reducir el desequilibrio propio de los modelos de una sola etapa entre la clase de fondo o *background* y las detecciones positivas.

Los modelos de una sola etapa se basan en proponer una gran cantidad de regiones en las que a priori puede haber un objeto, dichas regiones son llamadas *anchor boxes* y son la primera aproximación de las definitivas *bounding boxes*. En caso de que un objeto se encuentre en una *anchor box*, se colocará una *bounding box* en la posición precisa del objeto. Esta manera de proceder provoca que la clase más detectada una, vez procesada la imagen, sea la de *background*, que significa que el contenido de la *anchor box* no es un objeto. La relación entre la clase *background* y el resto puede es de 1000 a 1, por lo que la mayoría de errores de clasificación esta asociado a esta clase. Reducir el error asociado a esta clase puede provocar un deterioro de la calidad de las predicciones. Para evitar este contratiempo se diseña *Focal Loss*, esta función da menos importancia a los errores cometidos en las secciones de fácil predicción, como por ejemplo, las que no contienen objetos.El *Focal Loss* consiste en lo siguiente:

$$L_{cls} = -\sum_{i=1}^{K}(y_i log(p_i)(1-p_i)^{\gamma}\alpha_i + (1-y_i)log(1-p_i)p_i^{\gamma}(1-\alpha_i))$$

Figura 2.4: Ecuación del Focal loss. Fuente: [23]

Donde K corresponde al número de clases que el sistema es capaz de detectar, y_i toma el valor 1 cuando la clase detectada se corresponde con el objeto y p_i representa la probabilidad de que el objeto se haya detectado correctamente. El parámetro gamma se utiliza para restar peso a las clases que son fáciles de predecir como *background* para evitar que al error aumente de una manera artificial. El factor alfa es otra manera de dar proporcionalidad a los errores, generalmente el valor de alfa es inversamente proporcional a la frecuencia con la que la clase de objeto en cuestión aparece.

RetinaNet utiliza un esqueleto de red parecida a la ”Feature Pyramid Network” llamada así por la forma piramidal que adopta su visión esquemática [referencia]. Para detectar objetos con diferentes escalas en una sola imagen, esta se submuestrea para obtener una imagen de menor tamaño y peor calidad, este proceso se repite varias veces creando una forma de pirámide entre todas las capas. Cada capa es analizada por una o varias

capas convolucionales para obtener los objetos. Existen varios tipos de arquitecturas piramidales:

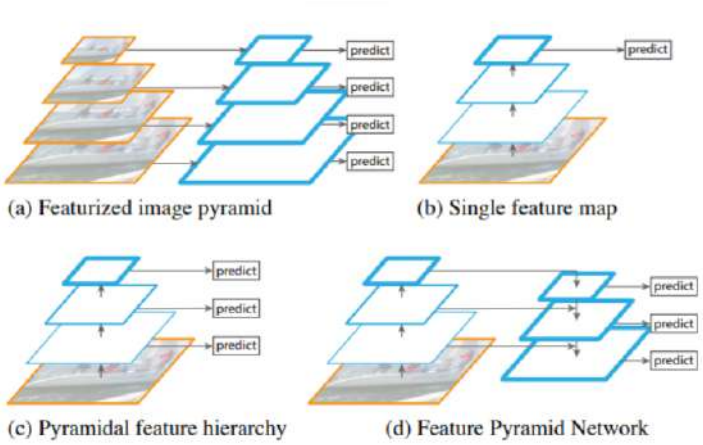


Figura 2.5: Diferentes tipos de estructuras piramidales. Foto de [7]

La estructura ”Featurized image pyramid” sigue la estructura explicada,se submuestrea la imagen y una capa convolucional analiza cada capa en busca de objetos. La estructura ”Single feature map”, es sin duda la más rápida ya que predice sus resultados a partir dela imagen que se encuentra en la cima de la pirámide, la más pequeña. Los resultados que aporta no son malos pero si mejorables. La ”Pyramidal feature hierarchy” genera sus predicciones en cada etapa del submuestreo.

Por último, la ”Feature pyramid network”, la utilizada por RetinaNet:

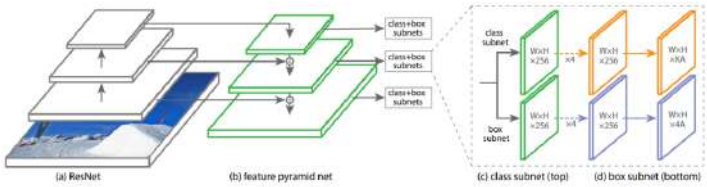


Figura 2.6: Arquitectura del sistema ”RetinaNet”. Foto de [7]

Esta estructura consta de cuatro etapas, en la primera la red (en la imagen de ejemplo ”ResNet”) calcula las diferentes imágenes de menor escala que componen la pirámide. En la pirámide paralela (feature pyramid net), se sobremuestrean las imágenes obtenidas y mediante las conexiones laterales se juntan las imágenes con dimensiones iguales.

El sistema RetinaNet toma los niveles P_3,...,P_7 de la red, dichos niveles tienen el mismo numero de canales (256) y sus *anchor boxes* tienen tres posibles relaciones de aspecto: 1:2, 1:1 o 2:1 y toman tres escalas diferentes: 2^0, $2^{\frac{1}{3}}$ o $2^{\frac{2}{3}}$. Por lo tanto hay 9 *anchor boxes* posibles por elemento

Si se toma por ejemplo una imagen de 1200x1200 píxeles los diferentes niveles de imágenes submuestreadas tendrán los siguientes tamaños: P_0 = 1200x1200, P_1 = 600x600, P_2 = 300x300, P_3 = 150x150, P_4 = 75x75, P_5 = 38x38, P_6 = 19x19 y P_7 = 10x10 por lo que en el nivel P_3 habrá 150x150x9=202500 *anchor boxes*

A continuación, en la red de clasificación se calcula la probabilidad de que un objeto se halle en cada *anchor box* y para terminar la red de regresión devuelve el offset para adecuar las *bounding boxes* a las *anchor boxes* que contienen objetos.

2.6. Keras-Retinanet

Para hacer la detección de aviones se ha decidido escoger el sistema ”Keras-Retinanet” que se puede encontrar en GitHub: https://github.com/fizyr/keras-retinanet. Se selecciona este programa debido a sus buenos resultados y sus bajos tiempos de computación.

Keras-Retinanet esta entrenado para identificar 80 objetos diferentes entere los que se encuentran los aviones. La red neuronal utilizada ha sido entrenada con el banco de imágenes ”COCO” usando la red ”resnet50”. El autor ofrece al usuario la posibilidad de elegir entre varias redes neuronales y en caso de que ninguna incluya los objetos que quiere identificar o los resultados no le resultan satisfactorios, el autor ofrece una script con la que se puede entrenar el sistema.

2.6.1. Rendimiento

En este apartado se pretenden mostrar las prestaciones del código Keras-Retinanet en diferentes situaciones :

- La primera situación que se ha seleccionado es una en la que se pueda ver la eficacia del sistema en imágenes que tienen varios objetos que reconocer.

Figura 2.7: Imagen original. Foto de ejemplo de [8]

En la imagen original se pueden reconocer cuatro objetos incluidos en la lista del sistema: tres personas y una corbata.

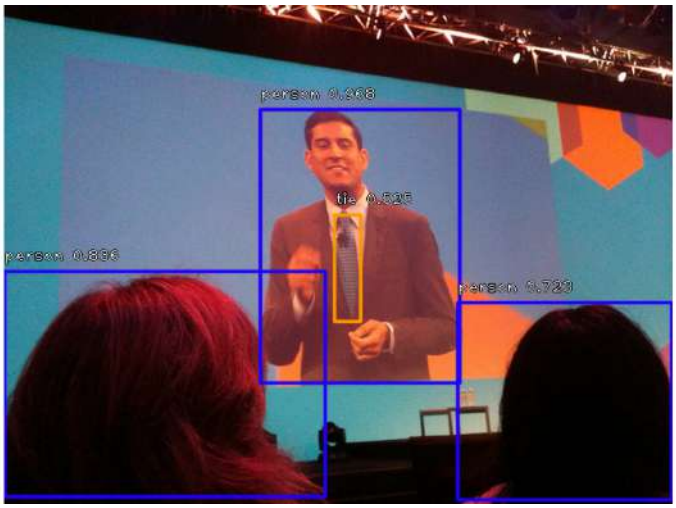


Figura 2.8: Imagen procesada con los objetos dibujados. Foto de ejemplo de [8]

El resultado es satisfactorio ya que reconoce los cuatro objetos y excepto en el caso de la corbata los reconoce con un porcentaje superior al 70 %.

- La segunda situación pretende simular las fotos tomadas al borde de pista para llevar el control de despegues y aterrizajes.

(a) (b)

Figura 2.9: Imágenes originales. Fotos de [9]

Estas fotos son muy fáciles ya que el avión es lo único que hay en ellas. Esta es la situación que se desea conseguir y para ello en el capítulo cinco 5 se realizan cálculos para ajustar la perspectiva de la cámara y así evitar que entren otros objetos.

(a) (b)

Figura 2.10: Imágenes procesadas con los ”bounding boxes” de los aviones encontrados. Fotos de [9]

Por supuesto el resultado es óptimo ya que los aviones son correctamente reconocidos con un alto porcentaje. Concretamente, Keras-Retinanet estima que los objetos de las imágenes (a) y (b) son aviones con un porcentaje de acierto del 99,7% y 99,3% respectivamente.

- La tercera situación simula perspectivas de cámaras de seguridad enfocando a zonas en las que hay varios aviones estacionados.

(a) (b)

Figura 2.11: Imágenes originales. Fotos de [9]

Estas fotos son más complicadas ya que los aviones se encuentran en planos diferentes o recortados.

(a) (b)

Figura 2.12: Imágenes procesadas con los ”bounding boxes” de los aviones encontrados. Fotos de [9]

Se puede ver en la foto (a) que no es capaz de detectar el avión del fondo y en la foto (b) no es capaz de detectar la cola de avión más cercana a la cámara. No parece que esto pueda suponer un gran problema ya que los aviones más cercanos y centrados son correctamente detectados y se espera que haya otras cámaras capaces de lograr mejores imágenes de los aviones mas lejanos o incompletos.

Visto el resultado de los tres casos, se puede concluir que el Keras-Retinanet representa una solución adecuada para cumplir la tarea encomendada.

2.6.2. Postprocesado

El sistema Keras-Retinanet devuelve el nombre, ubicación y porcentaje de acierto asociado a cada objeto que encuentra en la imagen de entrada. Lo que quiere decir que devuelve demasiados datos innecesarios que requieren un postprocesado para seleccionar los que son realmente necesarios para el sistema de detección de matriculas.

En primer cometido del postprocesado es filtrar los objetos que no son aviones de tal manera que todas las fotos que no contengan aviones quedan automáticamente descartadas y todos los objetos que no son aviones tampoco serán analizados.

En segundo lugar, en las fotos en las que se encuentren aviones se calcula el ratio entre el área de la ”bounding box” y el área de la imagen para que en pasos posteriores se utilice este resultado para decidir cual es la foto más adecuada para detectar la matricula del avión. Ya que la foto en la que el ratio de áreas sea mayor el avión en cuestión aparecerá entero y por tanto servirá para detectar la matricula y para guardarla como ”prueba” en caso de que se cometa un error.

Todos los datos asociados a el procesado del Keras-Retinanet se guardan en la tabla ”FOTOS” de la base de datos siguiendo el formato especificado en el cuarto capítulo 4.

CAPÍTULO 3. ALGORITMOS DE RECONOCIMIENTO ÓPTICO DE CARACTERES

3.1. Contexto en el proyecto

Los sistemas de reconocimiento óptico de caracteres o en inglés OCR son softwares dedicados a localizar y reconocer textos en imágenes. En este proyecto los sistemas OCR cobran un protagonismo capital debido a que son estos sistemas los que desarrollan la tarea más importante: Localizar textos en las imágenes y reconocer los caracteres que lo componen para poder saber la matricula del avión.

En el contexto del proyecto en el cual el objetivo es detectar las matriculas de aviones aparcados, de aviones que se disponen a despegar o vienen de aterrizar, la velocidad de procesado no es un aspecto crucial. Ambas aplicaciones mencionadas no precisan de resultados instantáneos ya sea por que son situaciones que se alargan en el tiempo, como el periodo de estacionamiento de un avión o por que el registro de operaciones del aeropuerto no requiere de actualizaciones instantáneas. Prima sobre la velocidad de procesado la eficacia a la hora de reconocer caracteres con diferentes orientaciones, tipos de fuente o color. En resumidas cuentas, se seccionará el sistema cuyos reconocimientos de texto sean los más parecidos a las matriculas que tiene que reconocer. En cualquier caso, se asume que el sistema seleccionado cometerá errores por lo que se ha diseñado un sistema (capítulo sexto 6) que pretende mitigarlos.

3.2. Origen de los sistemas OCR

Los primeros sistemas capaces de reconocer caracteres en imágenes nacen en la década de 1980, aplicando algoritmos sin capacidad de aprendizaje. El objetivo de estos sistemas era leer documentos escritos a maquina ya que es el caso más sencillo por la uniformidad de los caracteres en cuanto a forma, localización y entorno.

En un documento escrito a maquina como este el tamaño y tipo de letra acostumbran a ser iguales a lo largo del mismo, además el texto se encuentra siempre en la misma zona de la página la cual definen los márgenes.

Cuando los algoritmos de aprendizaje autónomo hicieron acto de aparición como una manera eficaz de reconocer texto en imágenes, se comenzaron a analizar fotos que no necesariamente eran documentos de texto como hasta el momento. Cabe mencionar que los primeros sistemas de OCR que trataban de detectar texto en escenas variadas no eran algoritmos de aprendizaje autónomo pero sus resultados no se pueden catalogar como satisfactorios.

El reconocimiento de texto en escenas es más difícil que en documentos por los mismos motivos expuestos anteriormente. El texto puede encontrarse en cualquier parte con cualquier color o forma por lo que se pierde la uniformidad de los documentos. De esta circunstancia surge la necesidad de dividir la comprensión de texto en dos problemas diferenciados: Encontrar el texto en la imagen, detección, e identificar y mostrar los caracteres que se encuentran en dichas regiones, reconocimiento.

Por lo tanto, la mayoría de sistemas OCR cuentan con dos algoritmos diferentes que se encargan cada uno de su función: Un algoritmo detector que busca las diferentes regiones de texto de la imagen y un segundo que reconoce las regiones de texto y transcribe el texto. El resto son sistemas *end-to-end* o *text spotting* que combinan ambos procesos.

3.3. Detección

Los sistemas de detección son el primer paso del proceso de traducir las imágenes a texto cuyo cometido es buscar la posición y forma en la que aparecen las palabras en cualquier imagen. La detección es un problema parecido al de la detección de objetos pero tiene particularidades que requieren de sistemas específicamente diseñados para ello.

Desde los primeros sistemas de detección estos han sufrido grandes avances y mejoras, a continuación se enumeran las más relevantes:

- Reducción del número de operaciones: Antes de la aparición de los sistemas basados en *deep learning* los sistemas de reconocimiento realizaban un alto número de procesos por análisis. Desde el desarrollo de sistemas de aprendizaje autónomo, el número de procesos intermedios ha disminuido notablemente limitando la propagación de errores.

- Descomposición en subtextos: Igual que los seres humanos, los sistemas de detección modernos son capaces de determinar si en una zona concreta hay texto sin necesidad de ver todo el texto. Gracias a esto, los detectores tienen varias maneras de identificar texto las cuales se diferencian principalmente por el tamaño de la sección de la imagen que se analiza. La sección más grande es la llamada instancia de texto la cual abarca todo el bloque de texto, se puede decir que este tipo de detección es parecida a la detección de objetos.

  Cuando el sistema detecta fragmentos de texto y después los combina para obtener la instancia de texto completa se dice que esta uniendo subtextos. Existen dos métodos principales de detectar y unir subtextos: Los métodos basados en componentes predicen regiones en las que hay uno o más caracteres. Los métodos basados en píxeles generan una probabilidad de que un píxel sea parte de un texto para todos los píxeles de la imagen y en la etapa de postprocesado se decide que píxeles son realmente parte de un texto.

- Optimización para problemas concretos: Una vez resuelto el problema de la detección se han creado sistemas específicos para mejorar ciertos casos en los que los resultados no eran del todo óptimos:

- Texto largo: A diferencia de otros objetos, los textos pueden tomar relaciones de aspecto muy diferentes, esto supone un problema porque los textos largos o con una relación de aspecto muy elevada pueden no ser detectados como uno solo y fragmentarse o ser detectados parcialmente. Para solucionar los problemas se han creado sistemas como *R2-CNN* [31] o *Seg-Link* [32].
- Múltiples orientaciones: A diferencia de la mayoría de los objetos, los textos pueden aparecer en diferentes orientaciones diferentes a las clásicas vertical y horizontal como diagonal. Por lo tanto, el sistema de detección tiene que estar preparado para detectar texto en cualquier orientación.
- Formas irregulares: Las formas en las que se puede encontrar texto en las imágenes varían más allá de cambios de relación de aspecto u orientaciones, el texto puede encontrarse con formas irregulares como la circular. Para atajar este problema se han desarrollado sistemas que no dependan de *bounding boxes* rectangulares como *TextSnake* [33] o *CRAFT* [14].
- Optimización para velocidad: Existen aplicaciones que precisan de un detector de texto rápido y eficiente, para casos como este se han desarrollado sistemas que han logrado un equilibrio entre velocidad y resultados. Ejemplo de estos sistemas es *EAST* [10].
- Encontrar el texto designado: Un avance notable en los detectores son los sistemas que son capaces de detectar un texto en concreto a partir de las indicaciones que el usuario le de a este. Un ejemplo de indicación sería ”texto negro grande encima de un coche” y un ejemplo de sistema es el desarrollado por X. Rong, C. Yi, y Y.Tian [34].

3.4. Reconocimiento

Los sistemas de reconocimiento son los encargados de traducir los resultados generados por el sistema de detección de texto. Para ello reciben los recortes de fotos generados por el detector, estos recortes se corresponden con los *bounding boxes* que muestran donde se ha detectado la presencia de texto.

Los sistemas de reconocimiento se pueden clasificar en función de que técnica de segmentación de caracteres utilizan:

- Basados en CTC (*Connectionist Temporal Classification*) [35]: Esta técnica consiste en utilizar una red convolucional para extraer características de la imagen y una red recurrente para calcular la probabilidad de que un *frame* contenga un carácter. Como el número de *frames* no es igual al número de caracteres, se utiliza la técnica de CTC para obtener la secuencia final de caracteres.
- Basados en *attention*: Los modelos basados en *attention* utilizan el contexto para determinar cuales son los caracteres que forman la cadena a analizar. Dicho contexto consiste en una serie de probabilidades de que ciertos caracteres estén juntos. Por ejemplo, si un sistema basado en *attention* para reconocer palabras en inglés [36] reconoce una ”t” y una ”e” pero no reconoce el símbolo que hay entre ellos, el sistema decidirá que se trata de una ”h” porque la palabra ”the” es muy común en el inglés.

3.5. Sistemas OCR utilizados

A la hora de crear un sistema OCR existen una amplia gama de diferentes posibilidades gracias a la gran cantidad de modelos diferentes tanto de detectores como de reconocedores que hay. Por tanto, la tarea de encontrar la combinación que más se ajuste a las necesidades de cada caso es sumamente importante ya que la eficacia del sistema depende de esta.

Como se ha argumentado anteriormente, para el caso de la aplicación de estos sistemas a aeropuertos es crucial que la eficacia sea la máxima posible a costa de que la velocidad se vea resentida. Pese a esta premisa, se ha probado un algoritmo que usa el modelo *EAST* en la etapa de detección confiando en que para casos sencillos sería capaz de hacer una correcta y rápida detección.

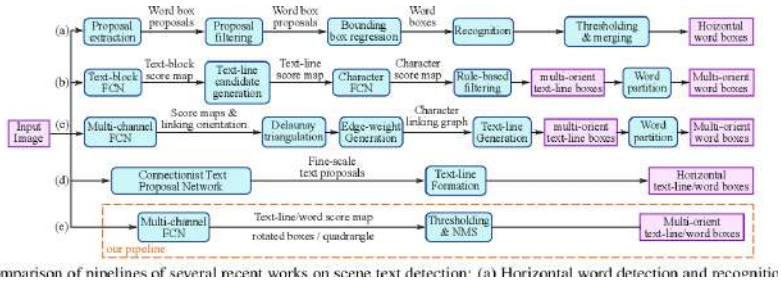
Las próximas secciones tratan de explicar los diferentes sistemas probados y los resultados obtenidos con estos.

3.5.1. Sistema basado en EAST y Tesseract

El primer sistema a analizar es el llamado ”Text Recognition EAST detection and Tesseract”. Se trata de un algoritmo que usa el modelo *EAST* en la etapa de detección y el *Tesseract* en la etapa de reconocimiento. El sistema puede descargarse en GitHub: https://github.com/pranavstha11/Text-Recognition-EAST-detection-and-Tesseract-.

3.5.1.1. Detector EAST

El metodo EAST (Efficient and Accurate Scene Text detector) [10] es creado en 2011 con el claro propósito de reducir etapas de procesado sin sacrificar eficacia, para ello simplifica el *pipeline* consiguiendo que el proceso de detección se realice en solo dos etapas.



Comparison of pipelines of several recent works on scene text detection: (a) Horizontal word detection and recognitio

Figura 3.1: Comparación de *pipelines* de diferentes sistemas de detección de texto el *pipeline* (e) es el del modelo EAST. Foto de [10]

- La primera etapa es una FCN (*Fully Convolutional Network*) que esta diseñada para obtener directamente la predicción de la localización de texto en la imagen. La red consta de cuatro etapas de convolución con diferentes tamaños de filtro para extraer características de la imagen y otras cuatro para unir dichas características con el fin de obtener una predicción píxel por píxel de la presencia de palabras o textos.

Esto elimina pasos intermedios como la proposición de candidatos, formación de regiones de texto y la partición de palabras.

A la salida de esta red se obtienen varias predicciones donde se indica la localización del texto y el porcentaje de fidelidad.

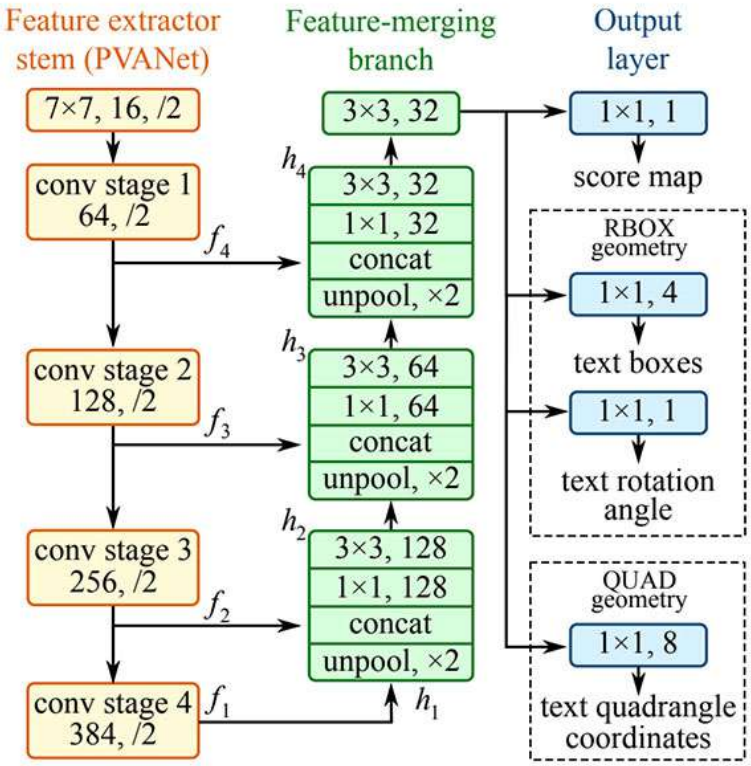


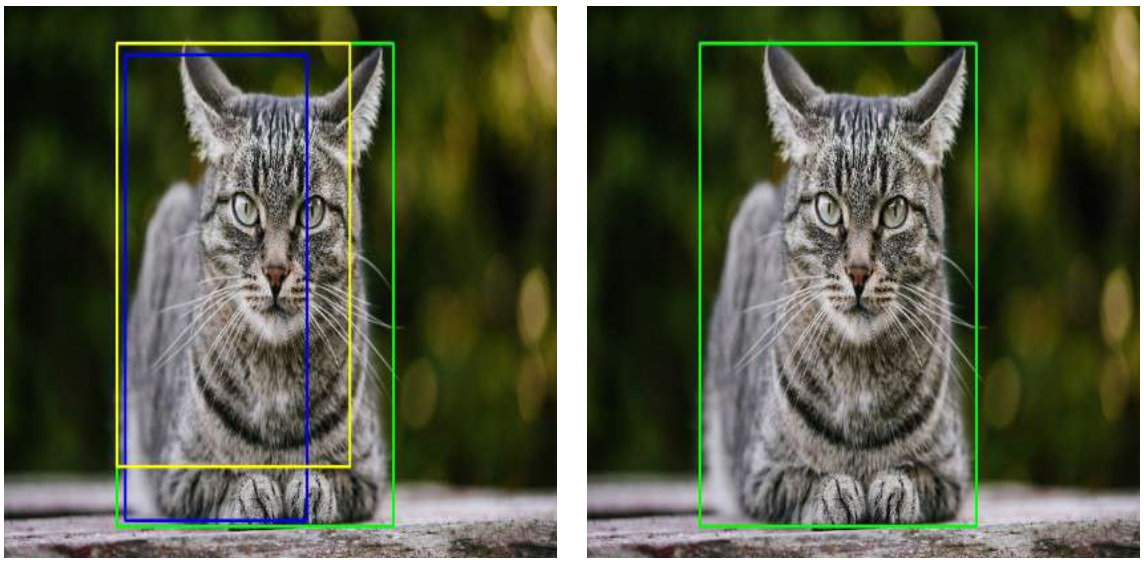
Figura 3.2: Representación de la FCN. Foto de [10]

- La segunda etapa consiste en realizar un filtrado entre los candidatos propuestos por la FCN para encontrar la *bounding box* que mejor se ajusta a cada texto encontrado, para esto se utiliza una técnica llamada *Non Max Suppresion* o NMS. Esta técnica no es exclusiva de sistemas de detección de texto si no que se utiliza también en sistemas de detección de objetos.

  El filtrado de resultados por NMS consta de dos pasos principales, en el primero se eliminan los candidatos que no cumplen con un umbral mínimo de fidelidad. Se trata de candidatos que han detectado erróneamente una región de texto o que han detectado solo una parte de esta y no son suficientemente fiables como para tenerlas en cuenta en la segunda etapa.

  El segundo paso consiste en ordenar los candidatos en función de su fidelidad de mayor a menor, la lógica que sigue este proceso es que si hay varios candidatos cuyas *bounding boxes* se superponen más de un umbral y pertenecen a la misma clase es altamente probable que estén encuadrando al mismo objeto y por tanto hay que buscar al mejor. Por lo tanto se calculará el solapamiento entre todos los candidatos y el más fiable de ellos si el solapamiento es mayor que el umbral, se elimina al candidato menos fiable.

Para calcular el solapamiento entre los candidatos se divide el área de la intersección entre los *bounding boxes* por el área de la unión entre ambos *bounding boxes*. El umbral seleccionado estará entre 0 y 1 al igual que el resultado de la operación.



(a) (b)

Figura 3.3: Ejemplo de selección de la mejor *bounding box* utilizando NMS. Fotos de [11]

3.5.1.2. Reconocedor Tesseract

Tesseract [12] es un sistema de reconocimiento de texto creado por HP en 1984 y desarrollado por Google desde la década de los 2000, por lo que se trata de uno de los sistemas más utilizados para esta tarea.

El sistema esta basado en un *pipeline* de procesado donde el input es un recorte de una foto donde solo se ve la región en la que hay texto. El primer paso del procesado es un análisis de los componentes interconectados que termina con sus contornos guardados. En esta etapa los contornos se unen formando *blobs* o manchas. Estos *blobs* están organizados en lineas de texto las cuales son analizadas para separarlas por palabras.

El reconocimiento consta de dos pasos, en primer lugar se intenta reconocer la palabra directamente. En caso de no encontrar nada satisfactorio, se procede a dividir la palabra en letras para detectarlas una a una y componer la palabra.

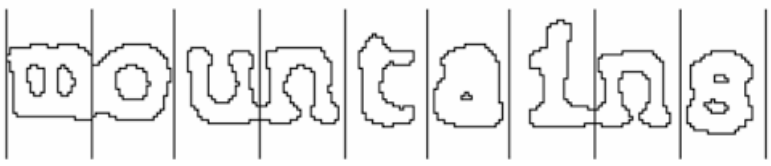


Figura 3.4: Ejemplo de cómo Tesseract divide una palabra. Foto de [12]

La clave de que el sistema de reconocimiento funcione correctamente esta en la detección de los espacios entre palabras y letras para separarlas sin partirlas a partir de los contornos detectados al principio del *pipeline*.

3.5.1.3. Rendimiento y resultados

Para medir el rendimiento del sistema se han evaluado ambas partes del mismo teniendo en cuenta también la velocidad de procesado. En este caso, la velocidad de procesado, veinte fotos por minuto, es elevada como se esperaba a juzgar por la arquitectura del detector. Para evaluar el sistema se han realizado diferentes pruebas con imágenes con diferentes grados de dificultad:

En las imágenes más sencillas solo hay una sección con texto y esta es bastante grande por lo que la detección tiene que ser correcta para que este sistema pueda ser tomado en cuenta como sistema definitivo.

(a) (b)

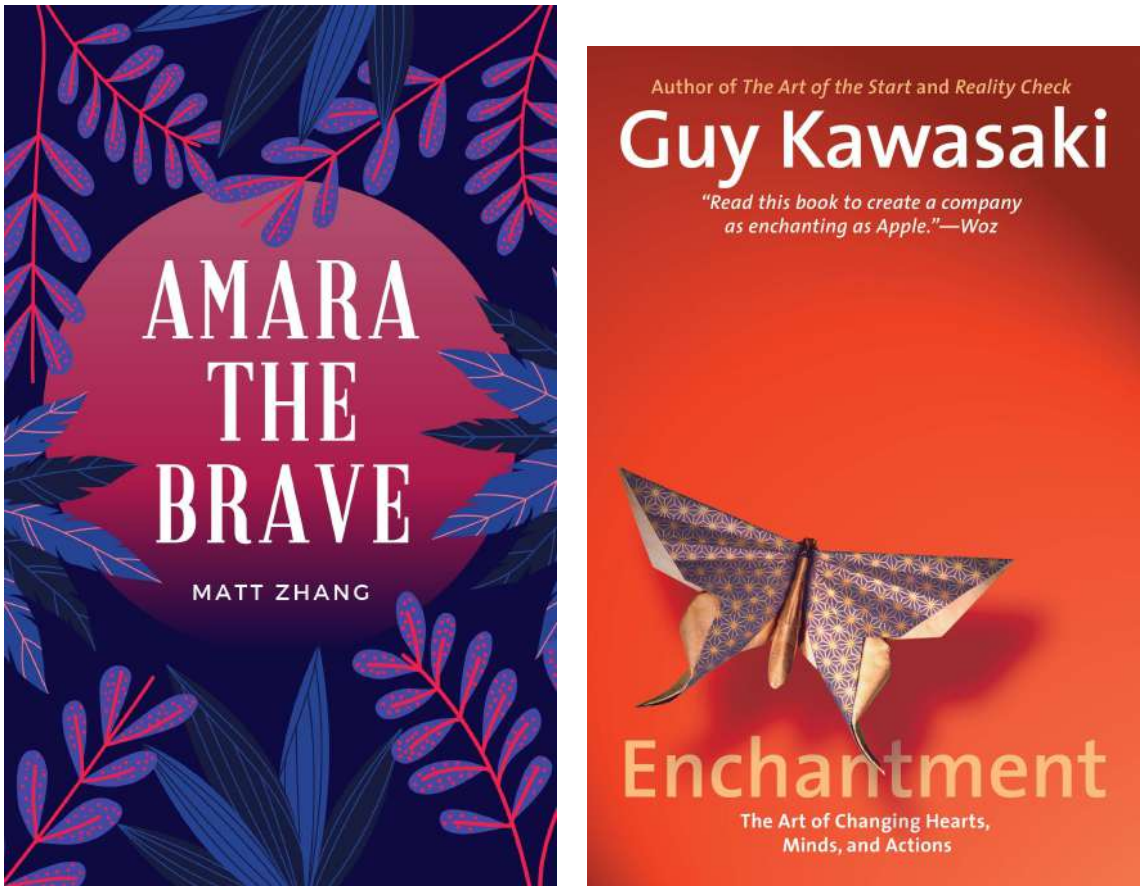
Figura 3.5: Imágenes sencillas, los textos de las matriculas deberían detectarse fácilmente. Fotos de [9]

(a) (b)

Figura 3.6: Resultados del procesado de las imágenes sencillas. Fotos de [9]

Los resultados no son satisfactorios ya que los cuadros de texto no son capaces de rodear todo el texto pese a que el texto está casi alineado con la horizontal.

El segundo grupo de imágenes esta compuesto por portadas de libros los cuales tienen varias secciones de texto. Es importante que el sistema sea capaz de detectar múltiples textos ya que en las fotos de hangares o aparcamientos de aviones pueden aparecer varios aviones y por tanto varias matriculas.



(a) (b)

Figura 3.7: Portadas de libros. Fotos de ejemplo de [13]

(a) (b)

Figura 3.8: Portadas de libros procesadas. Fotos de ejemplo de [13]

Los resultados vuelven a no ser satisfactorios ya que en la imagen con menos texto (imagen (a)) detecta textos en la zona en la que hay hojas y en la imagen con mucho texto (imagen (b)) no es capaz de detectar todo el texto.

Visto que la detección no cumple los requisitos para poder ser considerado el método idóneo para desempeñar las dos funciones sobre las que trata el proyecto. No hace falta realizar más pruebas de rendimiento ya que el sistema de reconocimiento tampoco cumple con lo esperado. En las fotos de los aviones no es capaz de reconocer una sola letra bien y en las fotos de portadas de libros si que reconoce ciertas palabras correctamente como ”Amara” en la foto (a) pero añade muchos signos de interrogación que responden a ningún tipo de símbolo que se halle dentro del recuadro.

Con esto, el sistema ”Text Recognition EAST detection and Tesseract” queda descartado.

3.5.2. Keras-OCR

El segundo sistema a evaluar es el ”Keras-OCR”. Es un sistema que combina el modelo *Craft* de detección y el modelo *CRNN* de reconocimiento. Este sistema puede descargarse en GitHub: https://github.com/faustomorales/keras-ocr.

3.5.2.1. Detector CRAFT

El modelo de detección CRAFT (Character Region Awareness For Text Detection) destaca por su capacidad de obtener correctamente textos de varias longitudes y en varias orientaciones como horizontal, curva o arbitraria.

Este modelo de detección también se basa en una FCN parecida a la del sistema VGG-16 de clasificación y detección de objetos para obtener las características de bajo nivel de la imagen y calcular así las dos probabilidades clave: *region score* y *affinity score*.

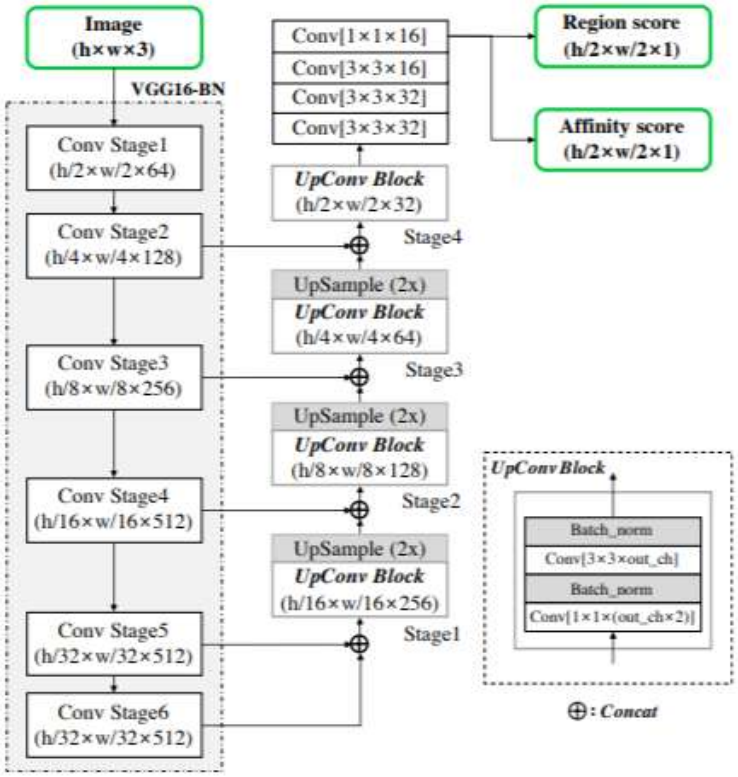


Figura 3.9: Representación de la arquitectura del sistema CRAFT. Fuente: [14]

La *region score* representa la probabilidad de que cierto píxel sea el centro de un símbolo, ya sea letra o número y la *affinity score* representa la probabilidad de que cierto píxel sea el centro del espacio entre dos símbolos.

Estas dos probabilidades no se miden de manera binaria, si no que se utiliza una función de probabilidad Gaussiana para representarlas. En la figura se puede observar el proceso de cálculo de ambas probabilidades. La red convolucional ofrece a su salida la imagen con las *character box* dibujadas, que son rectángulos que rodean a un símbolo. A partir de estos se generan las *affinity box* que representan la zona entre los centros

de los símbolos. Para calcular la posición de los vértices del rectángulo se calculan las diagonales de dos *character box* contiguas y tomando los triángulos superior e inferior generados por las diagonales, se calcula su punto central. Estos cuatro puntos centrales de los triángulos representan los cuatro vértices de la *affinity box*.

Para representar las probabilidades se toma una función Gaussiana isotrópica bidimensional y se adecua a la *character box* o *affinity box* que le corresponda. El proceso de adecuación es necesario ya que los rectángulos tienen diferentes inclinaciones dependiendo de la perspectiva de la imagen. Finalmente se podrán observar la *region score*, calculada a partir de *character box* y la *affinity score*, calculada a partir de *affinity box*

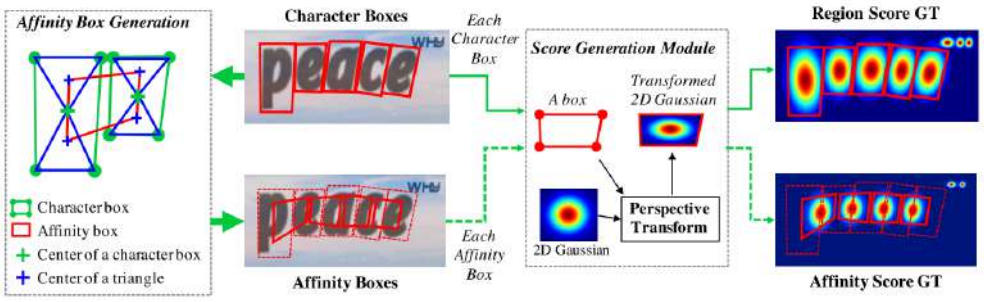


Figura 3.10: Representación del proceso de obtención de las *character box* y *affinity box*. Fuente: [14]

Este método de calcular la localización los diferentes elementos del texto por separado permite que dicho texto sea muy largo o se vaya curvando arbitrariamente ya que la probabilidad de afinidad se calcula a partir de los caracteres adyacentes.

El paso final una vez obtenidos los mapas de probabilidades consiste en crear un mapa final binario en el que se determina donde está el texto. Un algoritmo de *Connected Component Labelling* [37] se encarga de etiquetar las regiones de texto para que a continuación se calcule un rectángulo con el mínimo área posible para encuadrar cada palabra.

Los algoritmos de *Connected Component Labelling* sirven para buscar similitudes entre píxeles de una región para unir bajo una misma etiqueta a los que compartan propiedades como intensidad de color. En este caso, dicho algoritmo se encarga de decidir donde se acaba el texto para delimitar el área de los rectángulos que lo encuadran.

3.5.2.2. Reconocedor CRNN

La arquitectura del sistema de reconocimiento de texto CRNN esta compuesto por tres capas: Una primera capa convolucional, una capa recurrente y finalmente una capa de transcripción.

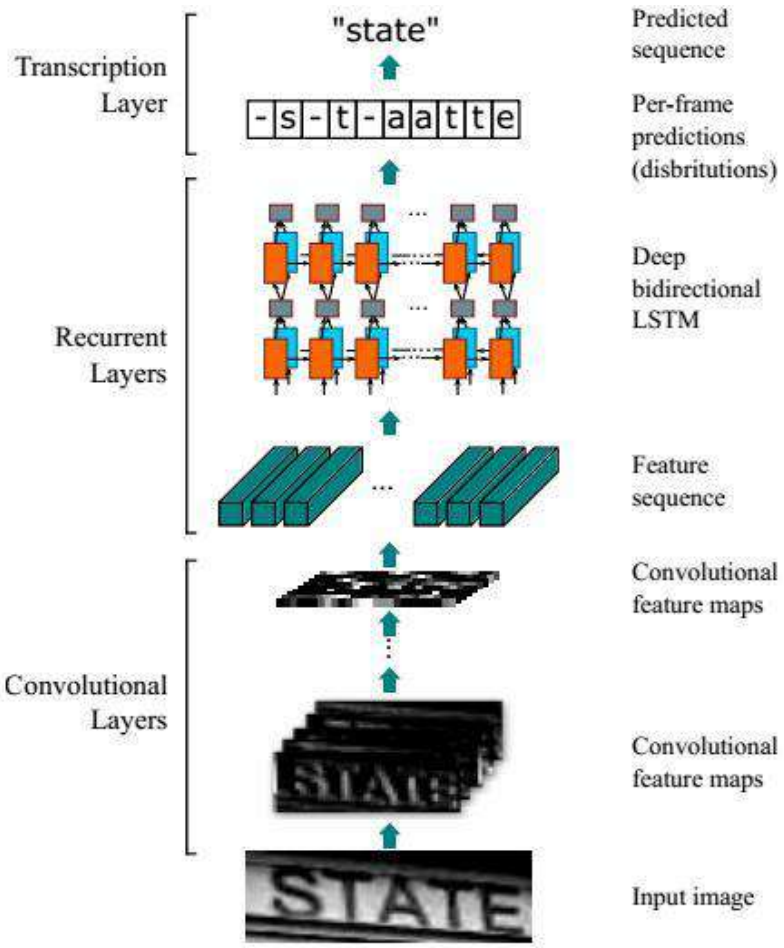


Figura 3.11: Representación de la arquitectura del sistema CRNN. Fuente: [15]

En la primera capa,la red convolucional, se usa una arquitectura parecida a los sistemas CNN (Convolutional Neural Network) con capas convolucionales y de max-pooling pero sin las capas totalmente conectadas. El objetivo de esta red es obtener las características de la imagen de entrada, estas tienen que ser escaladas a la misma altura antes de entrar en la red. Cada capa de convolución genera un mapa de convolución. El resultado de esta extracción de características es una secuencia de vectores de características. Cada uno de estos vectores es la concatenación de cada una de las columnas de todos los mapas generados. Habitualmente, dichas columnas son de un píxel de grosor.

En la segunda capa, la red neuronal recurrente se predicen etiquetas para cada *frame* de la secuencia de vectores de características. La red neuronal tiene tres principales ventajas, la primera es que tiene una gran capacidad de capturar información en su contexto. Esto quiere decir que es capaz de realizar el reconocimiento de una palabra sin tener que acudir a reconocer cada símbolo, gracias a esto es menos común que cometa errores de distinción entre símbolos parecidos como la ”i” y la ”l” ya que por comparación de alturas con el resto de letras del contexto puede decidir cual es la respuesta correcta. El segundo beneficio que aporta la red neuronal recurrente es su capacidad de propagar errores diferenciales hacia la capa anterior para que ambas sean entrenadas a la vez. Por último, es capaz de operar con secuencias de cualquier longitud.

En la capa de transcripción se realiza el proceso de convertir *frame* por *frame* las predicciones hechas por la red neuronal recurrente a una secuencia de etiquetas. Matemáticamente, se selecciona la predicción con más probabilidad de las etiquetas previstas por la capa anterior.

3.5.2.3. Rendimiento y resultados

Para medir el rendimiento del sistema se han evaluado ambas partes del mismo teniendo en cuenta también la velocidad de procesado. En este caso, la velocidad de procesado, tres imágenes por minuto, no es destacable pero si es suficiente para la aplicación a aeropuertos. Para evaluar el sistema se han utilizado tres grupos de imágenes:

En las imágenes más sencillas solo hay una sección con texto y esta es bastante grande por lo que la detección tiene que ser correcta para que este sistema pueda ser tomado en cuenta como sistema definitivo.

(a) (b)

Figura 3.12: Imágenes sencillas, los textos de las matriculas deberían detectarse fácilmente. Fotos de [9]

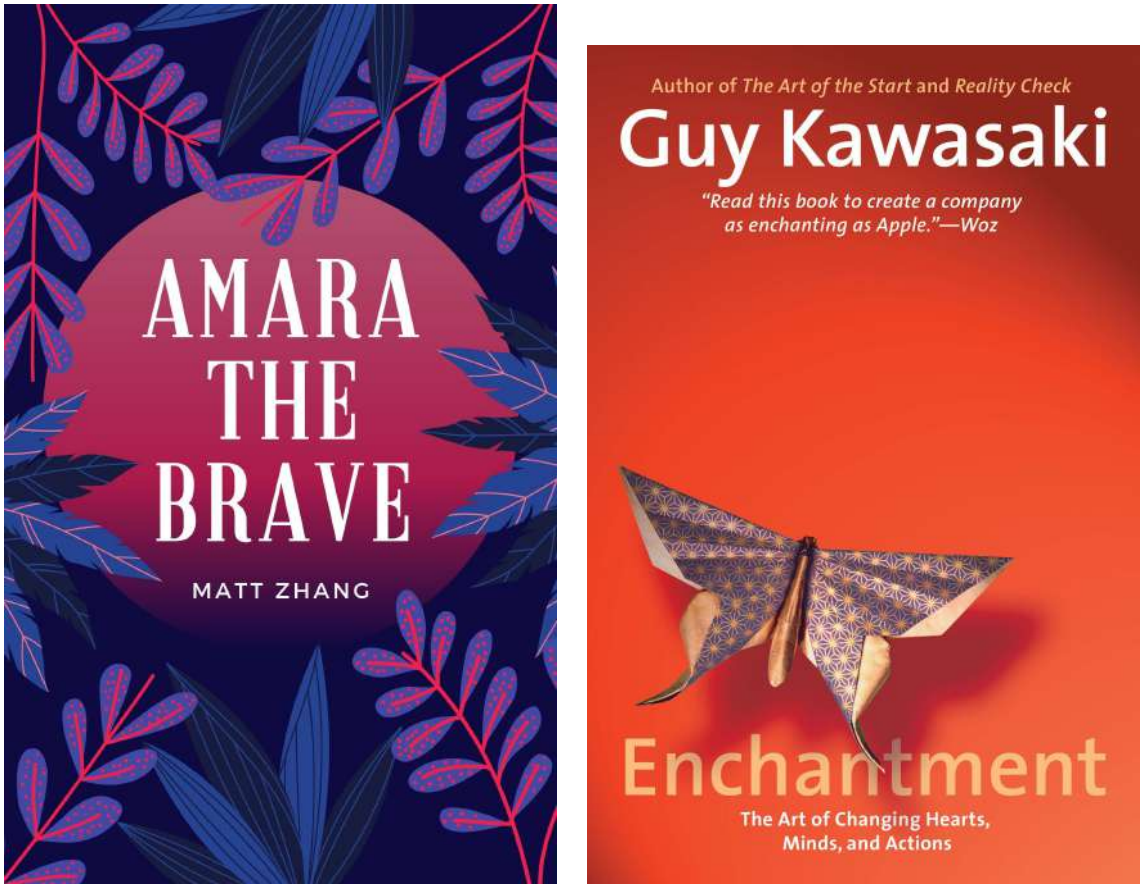
(a) (b)

Figura 3.13: Resultados del procesado de las imágenes sencillas. Fotos de [9]

Los resultados de la detección son satisfactorios ya que el detector detecta las matriculas y además detecta las marcas de agua con el nombre de la web de la cual se han obtenido las fotos.

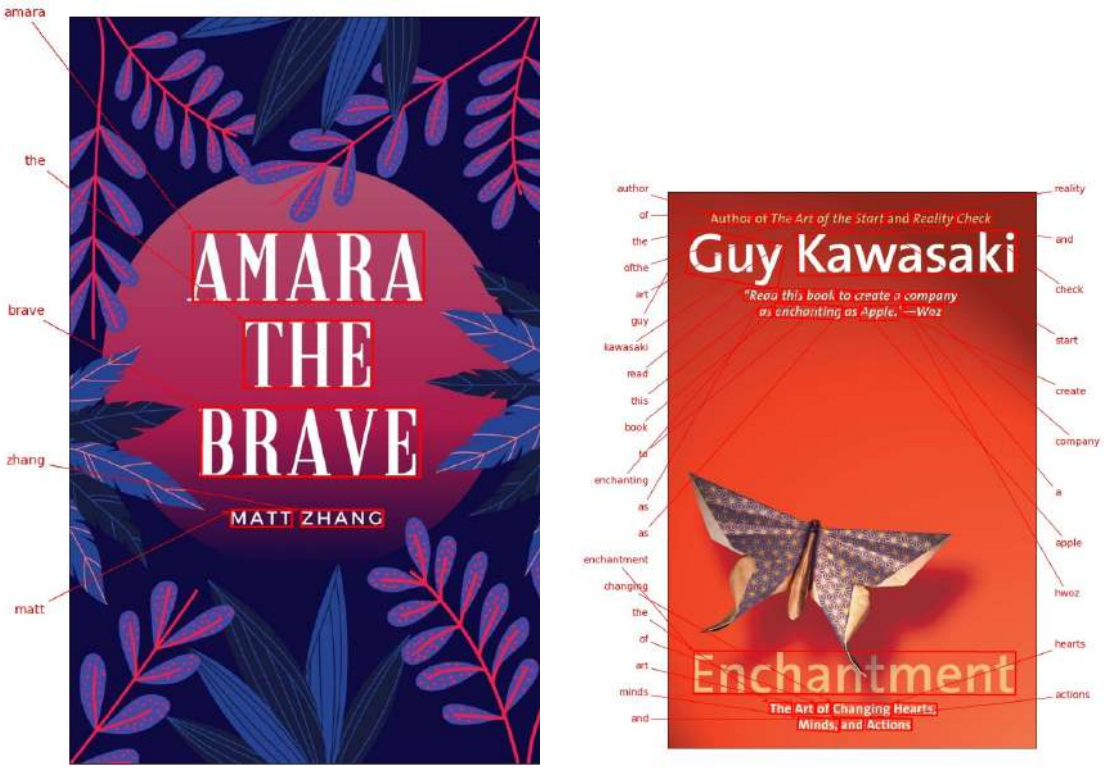
En cuanto al reconocimiento del texto de las matriculas, el resultado también es bastante bueno: En el caso de la foto (a), en vez de detectar ”N21286”, el sistema detecta ”nzizbb”. El proceso de reconocimiento genera bastantes errores como confundir ”2” con ”z”, ”1” con ”i” y ”86” con ”bb”. Todos los errores excepto el último son subsanables con un buen evaluador, por lo tanto, no se puede considerar que los resultados del reconocimiento son positivos para esta foto. En la imagen (b) el texto detectado para la matricula ”CC-PVS” el sistema detecta ”ccpvs”. La única discrepancia es la ausencia del guión lo cual se considera un error menor debido a que todas las letras de la matricula son correctas.

El segundo grupo de imágenes esta compuesto por portadas de libros los cuales tienen varias secciones de texto. Es importante que el sistema sea capaz de detectar múltiples textos ya que en las fotos de hangares o aparcamientos de aviones pueden aparecer varios aviones y por tanto varias matriculas.



(a) (b)

Figura 3.14: Portadas de libros. Fotos de ejemplo de [13]



(a) (b)

Figura 3.15: Portadas de libros procesadas. Fotos de ejemplo de [13]

Los resultados de detección son muy buenos, el sistema ha detectado todos los textos de ambas imágenes.

En cuanto al reconocimiento, la evaluación tiene que ser más minuciosa. En la imagen (a), el sistema reconoce correctamente todos los textos, tanto el titulo del libro como el autor del mismo.

La imagen (b) tiene muchas secciones de texto por lo que es muy probable que el sistema de reconocimiento cometa algún error. Sin embargo, el reconocedor solo ha cometido un error a la hora de reconocer el nombre del apellido del autor de la frase citada en el subtitulo. El texto original es ”-Woz” y se ha obtenido ”hwoz”. Es un problema poco relevante porque es fácilmente rectificable con un evaluador.

El resultado del procesado de estas fotos es altamente satisfactorio.

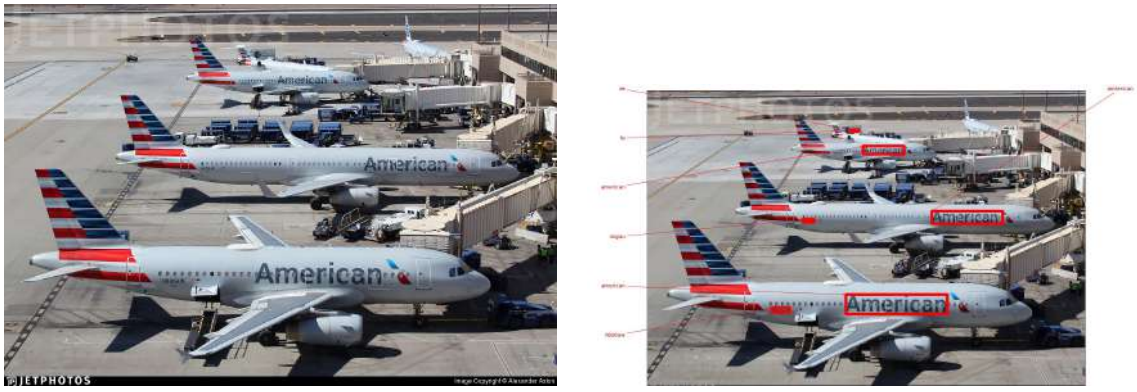
Para terminar, el tercer grupo de imágenes esta formado por dos imágenes que tienen dos dificultades particulares: La primera tiene una combinación de colores desfavorable al correcto reconocimiento y la segunda tiene muchos aviones en diferentes planos por lo que los textos son de varios tamaños.

(a) (b)

Figura 3.16: Comparación entre la imagen original (a) y la imagen procesada (b). Fotos de [9]

En el caso de la última foto, la detección ha sido correcta: Se ha detectado la matricula y además se ha localizado un texto cercano a la matricula que debido a su reducido tamaño era difícil de detectar. El reconocimiento del texto de la matricula, sin embargo, no cumple con las expectativas y reconoce ”srn” en vez de ”G-RANE”. La diferencia entre ambos texto es demasiado grande como para ser corregida por lo que se puede calificar el reconocimiento como incorrecto.

La última foto a analizar es de un aeropuerto en el que hay tres aviones aparcados uno delante de otro que como se ha mencionado anteriormente, representan la dificultad de la foto que radica en la variedad de planos en los que se encuentran los aviones.



(a) (b)

Figura 3.17: Comparación entre la imagen original (a) y la imagen procesada (b). Fotos de [9]

El detector ha sido capaz de detectar dos de las tres matriculas que aparecen en la imagen, la tercera que no ha sido capaz de detectar no supone un problema ya que ocupa una región diminuta de la imagen. La matricula más cercana,”N806AW”, ha sido detectada y reconocida correctamente, pero la más lejana que ha detectado no ha sido correctamente reconocida ya que la matricula es ”N989AU” y el sistema la ha reconocido ”nsgau”. La predicción es razonablemente parecida a la matrícula por lo que puede que el evaluador sea capaz de corregir el error.

Como indicador de rendimiento, se puede ver que ha detectado las letras de la aerolínea

”American” en los tres aviones pero el reconocimiento de estas ha fallado en la matricula del avión intermedio en la que se ha detectado ”amterican”.

Los resultados generales del sistema ”Keras-OCR” son positivos y hacen pensar que se trata de un sistema capaz de realizar las tareas de detección de matriculas que se requieren en el proyecto.

3.5.3. Easy-OCR

El último sistema que se ha probado es el llamado ”Easy-OCR”, se trata de un sistema que al igual que el anterior usa el modelo *Craft* para la detección y el modelo *CRNN* para el reconocimiento. Vistos los esperanzadores resultados del ”Keras-OCR” se decide probar con otro sistema similar para ver si es capaz de mejorar al anterior. El sistema se puede encontrar en GitHub: https://github.com/JaidedAI/EasyOCR.

Ya que comparte arquitectura con el modelo anterior, a continuación se presentarán los resultados obtenidos. La explicación teórica de los modelos se encuentra en los capítulos 3.5.2.1 y 3.5.2.2.

3.5.3.1. Rendimiento y resultados

Para medir el rendimiento del sistema se han evaluado ambas partes del mismo teniendo en cuenta también la velocidad de procesado. En este caso, la velocidad de procesado, tres imágenes por minuto, no es destacable pero si es suficiente para la aplicación a aeropuertos. Para evaluar el sistema se han utilizado tres grupos de imágenes:

En las imágenes más sencillas solo hay una sección con texto y esta es bastante grande por lo que la detección tiene que ser correcta para que este sistema pueda ser tomado en cuenta como sistema definitivo.

(a) (b)

Figura 3.18: Imágenes sencillas con los textos muy visibles. Fotos de [9]

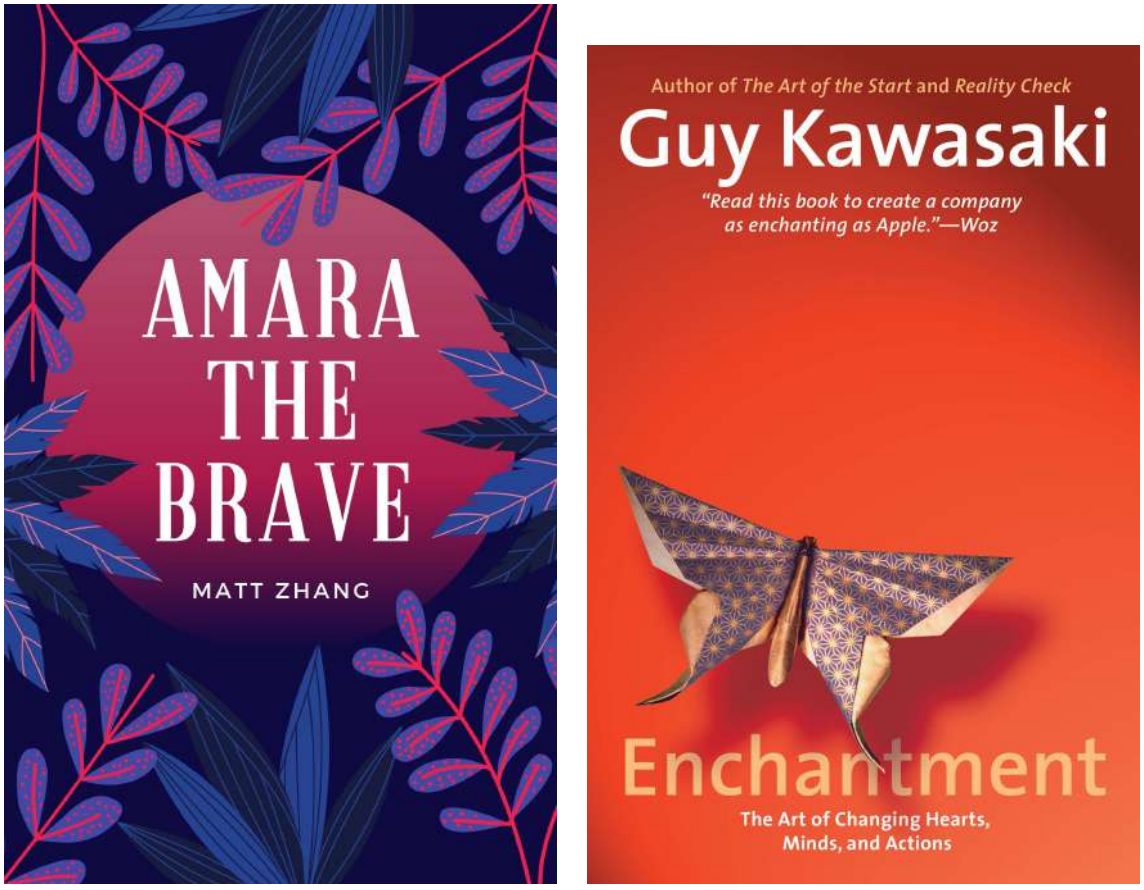
(a) (b)

Figura 3.19: Resultados del procesado de las imágenes sencillas. Fotos de [9]

Los resultados de la detección son positivos porque no solo detecta las matriculas, sino que también detecta la marca de agua de la web e incluso el nombre del autor de la foto que se encuentra en la barra negra que hay en la parte inferior de la imagen.

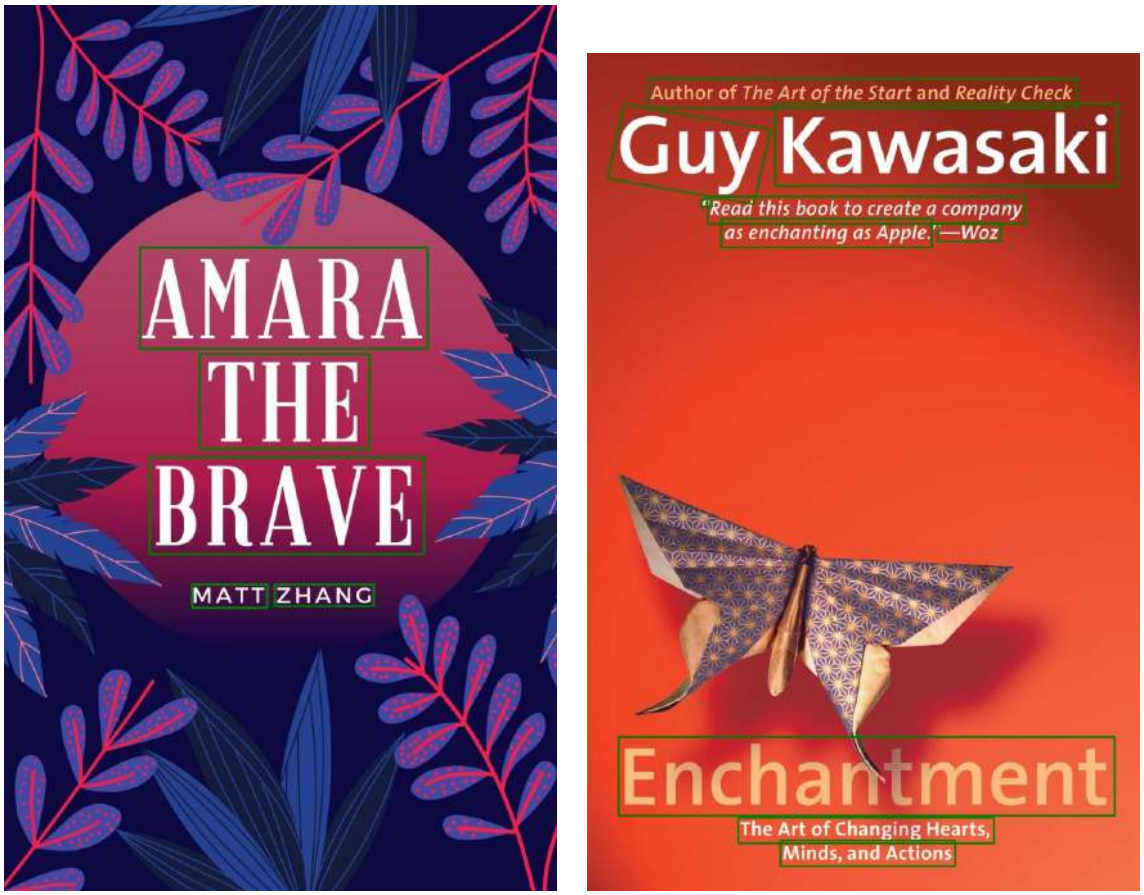
En cuanto al reconocimiento del texto de las matriculas, el resultado también es bastante bueno: En el caso de la foto (a), en vez de detectar ”N21286”, el sistema detecta ’n2/286’. El único error es confundir ”1” con ”/”, es un error fácilmente enmendable ya que son dos símbolos fáciles de confundir y que al ser un único error un evaluador bien preparado sería capaz de asociar el resultado a la matricula correcta. En la imagen (b) el texto detectado para la matricula ”CC-PVS” el sistema detecta ”cc pvs”. La única discrepancia es la ausencia del guión lo cual se considera un error menor debido a que todas las letras de la matricula son correctas.

El segundo grupo de imágenes esta compuesto por portadas de libros los cuales tienen varias secciones de texto. Es importante que el sistema sea capaz de detectar múltiples textos ya que en las fotos de hangares o aparcamientos de aviones pueden aparecer varios aviones y por tanto varias matriculas.



(a) (b)

Figura 3.20: Portadas de libros. Fotos de ejemplo de [13]



(a) (b)

Figura 3.21: Portadas de libros procesadas. Fotos de ejemplo de [13]

Los resultados de detección son muy buenos, el sistema ha detectado todos los textos de ambas imágenes.

En cuanto al reconocimiento, la evaluación tiene que ser más minuciosa. En la imagen (a), el titulo del libro es ”Amara the brave” pero el sistema ha reconocido ”Amara ihil brave” el confundir ”the” con ”ihil” es un error considerable porque el nulo parecido entre ambas palabras hace muy difícil que un evaluador pueda corregir este error. El autor del libro es ”Matt Zhang” y el sistema ha reconocido ”Matt Zhanc”, en este caso confundir una ”g” con una ”c” no supone un gran problema y es fácilmente detectable por un evaluador.

La imagen (b) tiene muchas secciones de texto por lo que es muy probable que el sistema de reconocimiento cometa algún error.

Hay cinco secciones de texto, la primera es ”Author of the art of the start and reality check” y el resultado del procesamiento es ”Author of the art ofthe start and keality check”, al realizar la comparación se aprecian dos diferencias: El sistema ha juntado las palabras ”of” y ”the” y ha confundido las letras ”r” y ”k”. Ninguna de las dos discrepancias son relevantes al ser fácilmente resolubles por un sistema evaluador.

El nombre del autor ”Guy Kawasaki” ha sido correctamente detectado por el sistema al igual que la frase de subtítulo ”Read this book to create a company as enchanting as Apple”. La discrepancia se encuentra en el autor de la frase del subtítulo, en vez de ”-Woz” ha detectado ”zwoz”. Una vez más el problema es insignificante por lo fácil que resultaría corregirlo con un evaluador.

Las secciones de texto localizados en la parte inferior de la portada han sido correctamente detectados por el sistema.

Para terminar, el tercer grupo de imágenes esta formado por dos imágenes que tienen dos dificultades particulares: La primera tiene una combinación de colores desfavorable al correcto reconocimiento y la segunda tiene muchos aviones en diferentes planos por lo que los textos son de varios tamaños.

(a) (b)

Figura 3.22: Comparación entre la imagen original (a) y la imagen procesada (b). Fotos de [9]

En este caso, el análisis de la foto es satisfactorio ya que se ha detectado correctamente la matricula y además el reconocimiento ha sido correcto, ha detectado ”G-RANE” que es exactamente la matricula. Además de la matricula, puede apreciarse la detección de un pequeño texto cerca de la matricula, este texto no supone un problema porque es fácilmente descartable en el postprocesado.

La última foto a analizar es de un aeropuerto en el que hay tres aviones aparcados uno delante de otro que como se ha mencionado anteriormente, representan la dificultad de la foto que radica en la variedad de planos en los que se encuentran los aviones.

(a) (b)

Figura 3.23: Comparación entre la imagen original (a) y la imagen procesada (b). Fotos de [9]

El detector ha sido capaz de detectar dos de las tres matriculas que aparecen en la imagen, la tercera que no ha sido capaz de detectar no supone un problema ya que ocupa una

región diminuta de la imagen. La matricula más cercana,”N806AW”, ha sido detectada y reconocida correctamente, pero la más lejana que ha detectado no ha sido correctamente reconocida ya que la matricula es ”N989AU” y el sistema la ha reconocido como letras en tailandés.

Como indicador de rendimiento, se puede ver que ha detectado las letras de la aerolínea ”American” en los tres aviones pero el reconocimiento de estas ha fallado en la matricula más lejana en la que se ha detectado ”anjerican”.

Los resultados generales del sistema ”Easy-OCR” son positivos y hacen pensar que se trata de un sistema capaz de realizar las tareas de detección de matriculas que se explican en el proyecto.

3.6. Selección del sistema OCR

Comparando los resultados obtenidos con los dos sistemas catalogados como eficaces, se puede observar que el sistema ”Easy-OCR” es ligeramente mejor ala hora de detectar textos que se hallan en el primer plano, pero a la hora de detectar múltiples textos el ”Keras-OCR” ha obtenido mejores resultados. Para poder decidir entre ambos sistemas, habrá que esperar a comparar los sistemas después del postprocesado y el evaluador. Se seleccionará el sistema que proporcione mayor porcentaje de acierto.

3.7. Postprocesado

El cometido de un sistema OCR es encontrar las regiones de texto de una imagen y traducir los píxeles a letras para luego mostrar las regiones encontradas y los textos reconocidos. Por lo tanto, a la salida de dichos sistemas el usuario se encuentra con un vector en el que cada coordenada contiene los datos que definen cada región de texto encontrada: Coordenadas del *bounding box*, texto reconocido y en el caso del ”Easy-OCR” la probabilidad asociada al reconocimiento.

El objetivo del postprocesado es filtrar en medida de lo posible los elementos del vector para intentar aislar el texto que corresponde a la matricula. Para ello se han desarrollado dos tipos de filtro que se distinguen por la característica que usan para discriminar entre resultados útiles e inútiles. Los filtros de dimensiones filtran usando las coordenadas de los *bounding box* y los filtros de contenido se fijan en el texto reconocido.

El postprocesado de las imágenes es imprescindible ya que los sistemas OCR se limitan a detectar y reconocer texto y para poder usar estos sistemas para aplicaciones como esta es necesario descartar resultados innecesarios.

3.7.1. filtros de dimensiones

Los sistemas ”EASY-OCR” y ”Keras-OCR” proporcionan las coordenadas de los *bounding box* en el mismo formato: Un vector en el que cada elemento son las coordenadas en ejes ”x” e ”y” de una de las esquinas del rectángulo. El problema es que el orden en el que se sitúan los puntos del rectángulo no es constante, es decir, el primer punto puede ser la esquina superior izquierda o la inferior derecha etc. Esto provoca que haya que ordenar las coordenadas para poder aplicar los filtros3.24. Para ordenar las coordenadas se ha decidido que el orden de las esquinas será el siguiente: Superior izquierda, superior derecha, inferior derecha e inferior izquierda, tal y como muestra la figura. En el caso de los rectángulos girados se mantendrá el orden elegido.

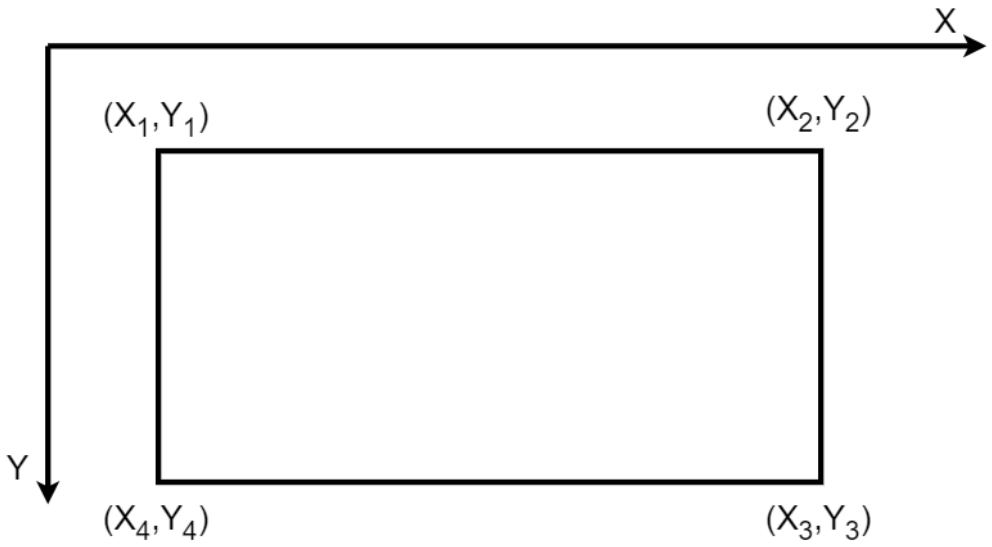


Figura 3.24: Orden correcto de los puntos

Una vez formateados los *bounding box*, se pueden aplicar los filtros diseñados para eliminar textos no correspondientes a matriculas. A partir de ahora, se nombrará cada punto por su índice siendo por ejemplo la esquina superior izquierda el punto uno.

- El primer filtro actúa en función del ángulo de inclinación del rectángulo. Las fotos que se van a procesar serán siempre de aviones que se encuentren en tierra por lo que las matriculas siempre se encontrarán alineadas con el eje horizontal del suelo, sin embargo existen textos que se encuentran en zonas como el timón vertical los cuales están inclinados respecto al suelo y no interesan ya que no son matriculas.

  Para realizar el filtrado se compararán la longitud del lado que forma la unión del los puntos uno y cuatro (altura del rectángulo y h_{lado} en 3.25) con la diferencia en el eje "y" de los puntos tres y cuatro(h_{imagen} en 3.25. Si dicha diferencia en el eje "y" es 1,5 veces mayor que la altura del rectángulo (1,5$h_{lado} < h_{imagen}$), se considerará que el rectángulo esta excesivamente rotado para poder tratarse de una matricula y será descartado.

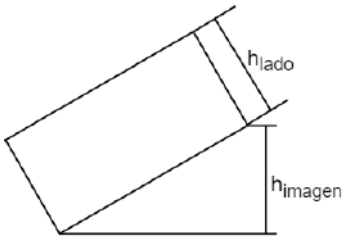


Figura 3.25: Distancias a tener en cuenta para filtrar por ángulo

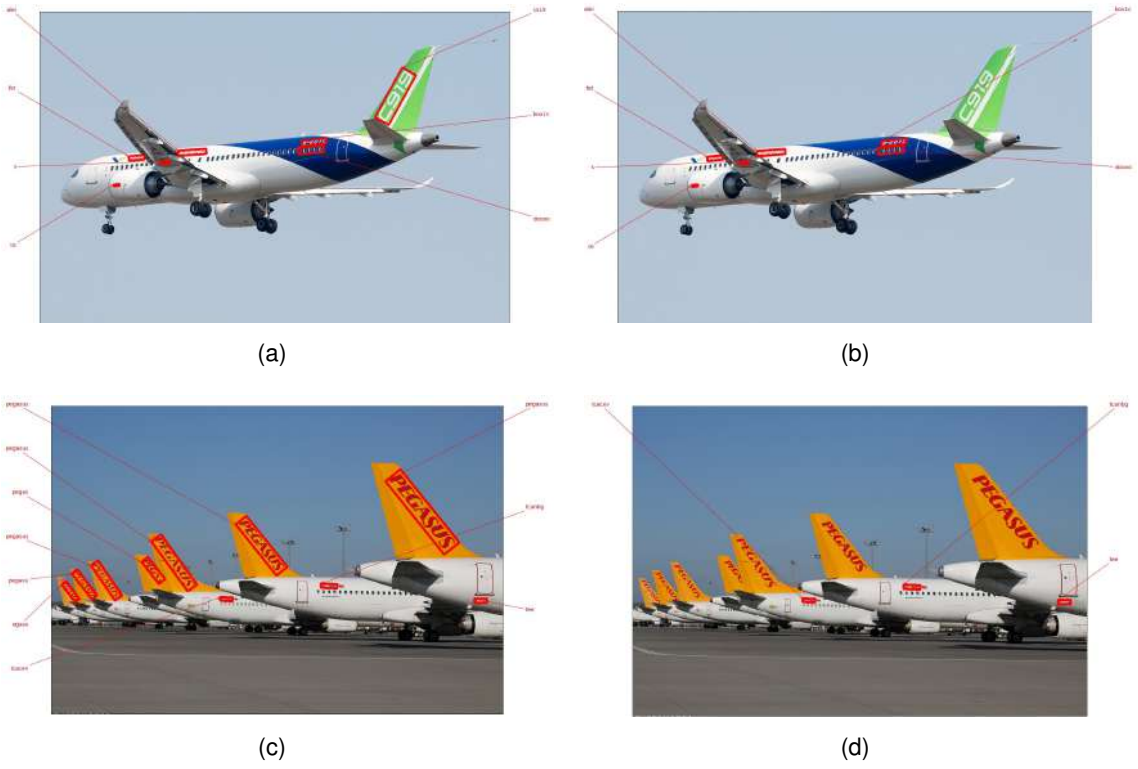


Figura 3.26: Ejemplos de filtrado de palabras que están demasiado inclinadas: Izquierda sin filtrar, derecha filtrado. Fotos de [9]

La figura 3.26 muestra claramente el motivo por el cual se aplica este filtro, los textos de cola quedan completamente filtrados.

- El segundo filtro discrimina por la relación de aspecto del rectángulo. Las matriculas acostumbran a cumplir cierta uniformidad en cuanto a la relación de aspecto de los *bounding box* que las encuadran por lo que todos los textos que superen un umbral serán descartados. La relación de aspecto más habitual para una matricula es de 3:1 pero dependiendo de la distancia del avión a la cámara y del tipo de avión la relación de aspecto puede aumentar hasta 7,5:1. Por lo tanto, se ha decidido poner el umbral en 7,75:1, este es suficientemente generoso como para asegurar que no filtrará por error ninguna matricula.

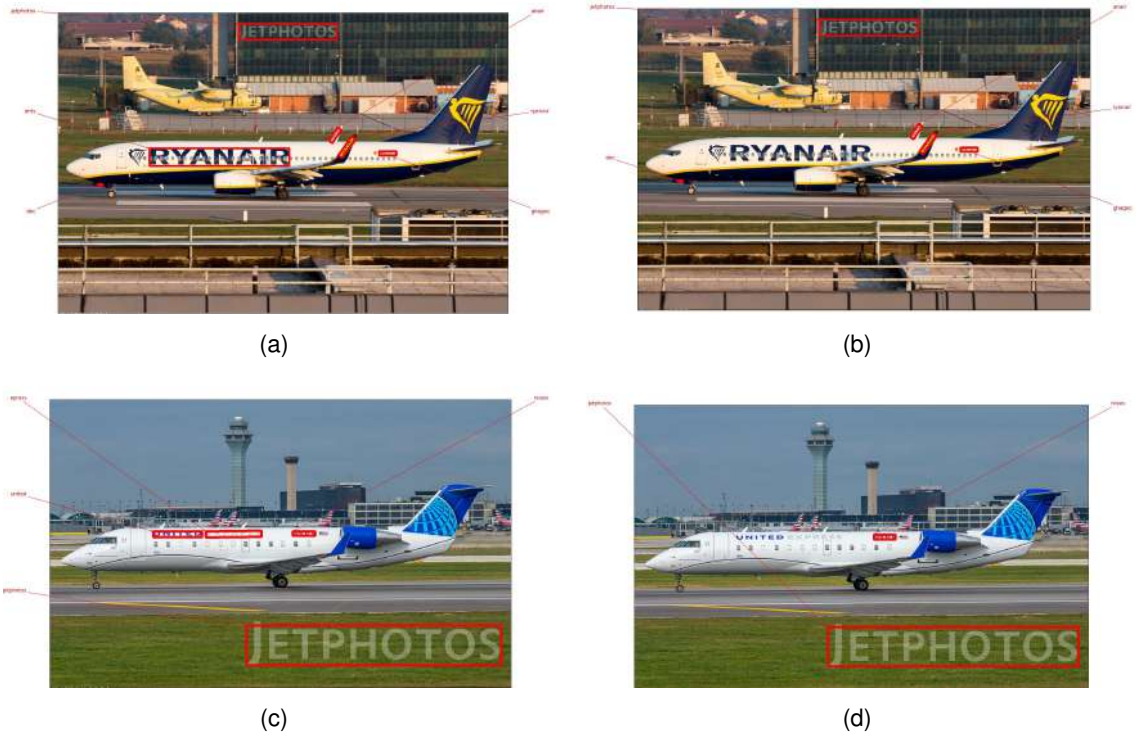
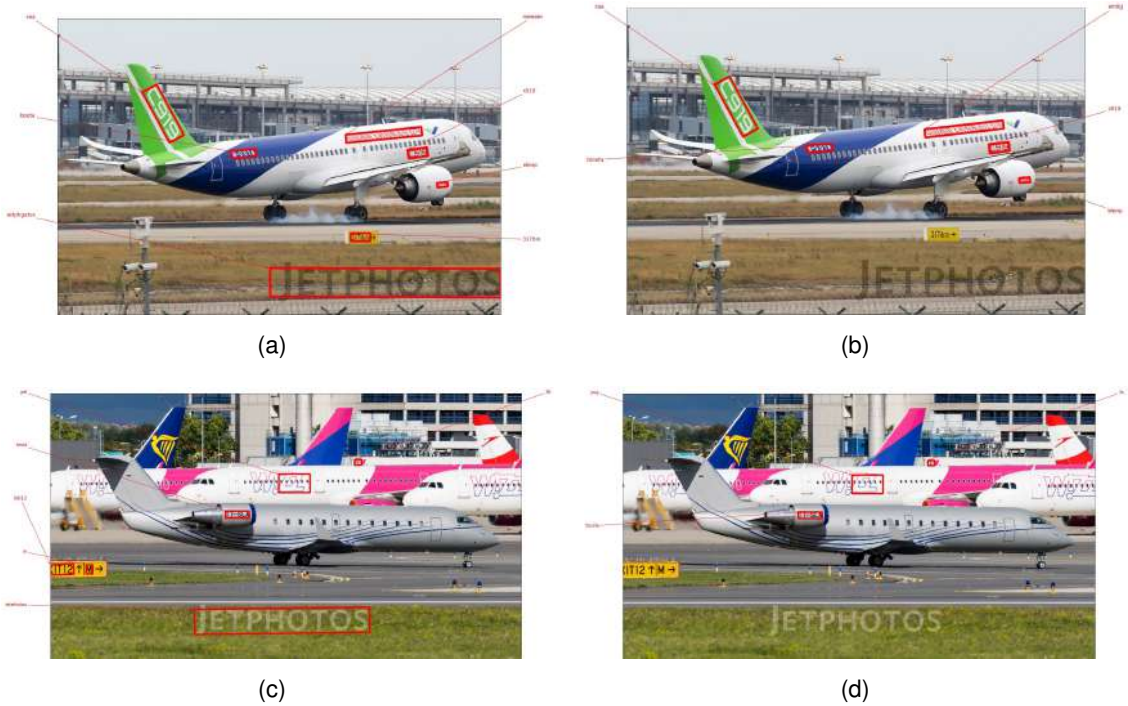


Figura 3.27: Ejemplos de filtrado de palabras que tienen una relación de aspecto demasiado grande: Izquierda sin filtrar, derecha filtrado. Fotos de [9]

La figura 3.27 muestra los principales objetivos del filtro, textos grandes que no son matriculas. En este caso, los textos filtrados de ambas imágenes son textos que de ser correctamente reconocidos los habría filtrado otro filtro pero como no se han ”leido” bien por parte del OCR, este filtro se encarga de eliminarlos.

- El tercer y último filtro de este grupo afecta a la posición del rectángulo en la imagen. Utilizando los datos ofrecidos por el detector de objetos, se comprueba si el centro del rectángulo está dentro del rectángulo que encuadra al avión o no, en caso negativo se descarta el texto. Este filtro está pensado para entornos en los que pueda haber letreros con texto o similares en segundo plano que puedan ser detectados.



(a) (b)

(c) (d)

Figura 3.28: Ejemplos de filtrado de palabras que no están en el avión: Izquierda sin filtrar, derecha filtrado. Fotos de [9]

En la figura 3.28 se puede ver que en ambas imágenes de ejemplo se eliminan los textos correspondientes a carteles a pie de pista y marcas de agua de la web propietaria de las fotos como en este caso es ”Jetphotos”

3.7.2. filtro de contenido

Los filtros de contenido afectan a los textos reconocidos por el sistema. Pese a que las matriculas no sean palabras que podamos buscar en un diccionario, cumplen ciertas normas en cuanto a longitud que se pueden utilizar para filtrar textos que no corresponden a matriculas.

- Por ello el primer filtro se encarga de descartar todos los textos que tengan menos de tres letras o más de ocho. Estas medidas responden a que las matriculas con menos letras o números tienen cuatro por lo que tres responde a la posibilidad de que el reconocedor no detecte un símbolo. En el otro extremo, las matriculas no superan los siete símbolos de longitud excepto en casos muy concretos en los que alcanzan los ocho, así que con estas longitudes se garantiza que la matricula no queda descartada a no ser que el sistema reconozca dos símbolos o menos.

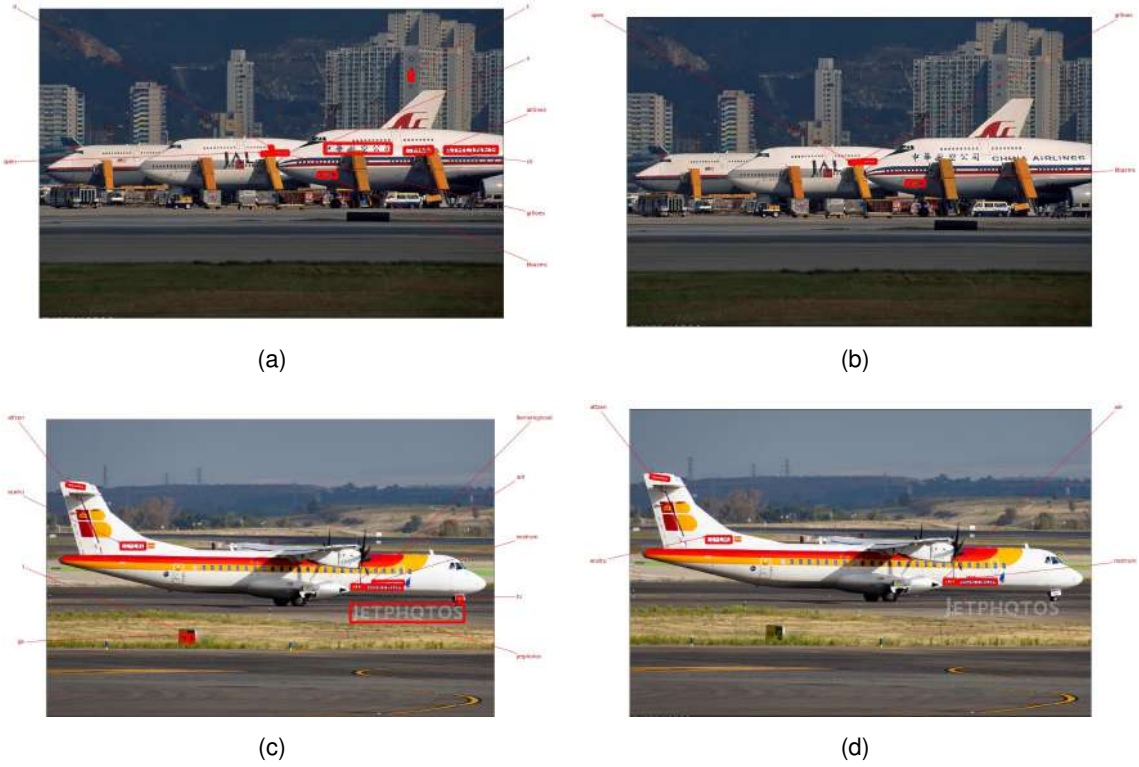


Figura 3.29: Ejemplos de filtrado de palabras cortas o largas: Izquierda sin filtrar, derecha filtrado. Fotos de [9]

La figura 3.29 ejemplifica el funcionamiento del filtro, en la primera imagen (a) se puede ver como filtra textos cortos como el del edificio que está en segundo plano o el del segundo avión. En el caso de la segunda foto (c) se observa como filtra los textos cortos de los carteles de cerca de pista y varios textos largos y cortos del avión.

- Un error recurrente de los sistemas OCR es confundir las ventanas de los aviones con textos en los que pone ”ooooo”, para evitar que estos se cuelen, se ha añadido un filtro el cual elimina el texto cuando este tiene la cadena ”ooo”.

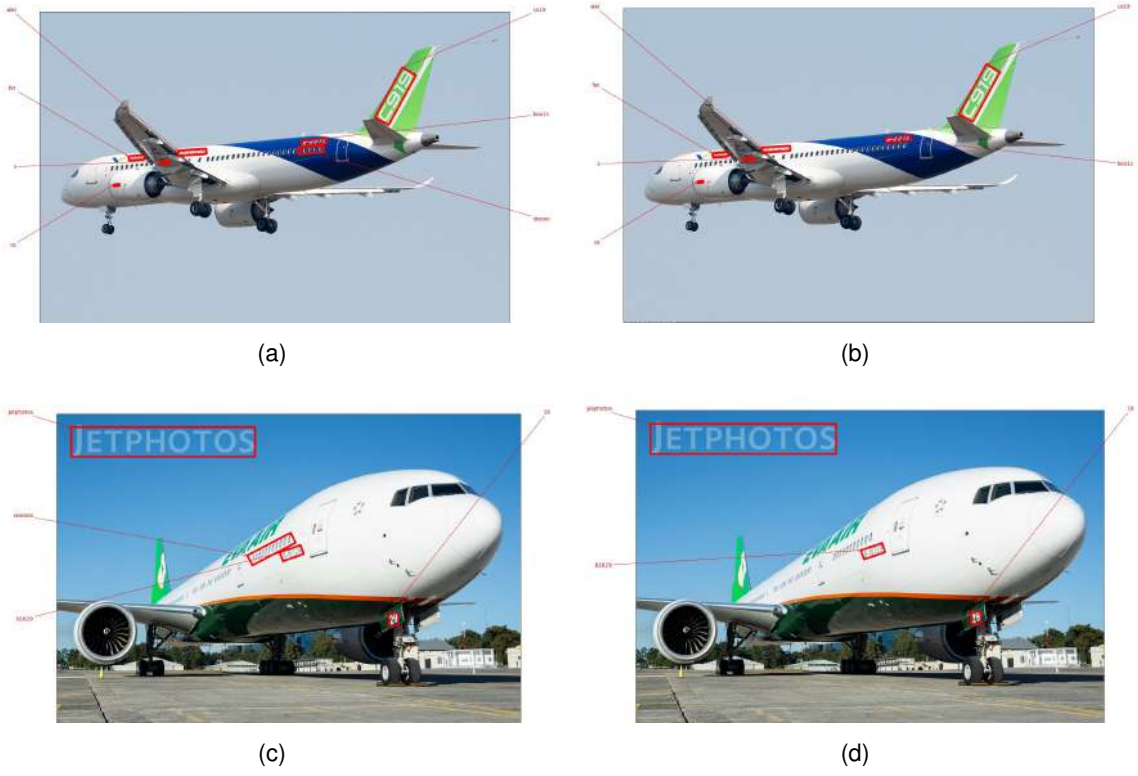
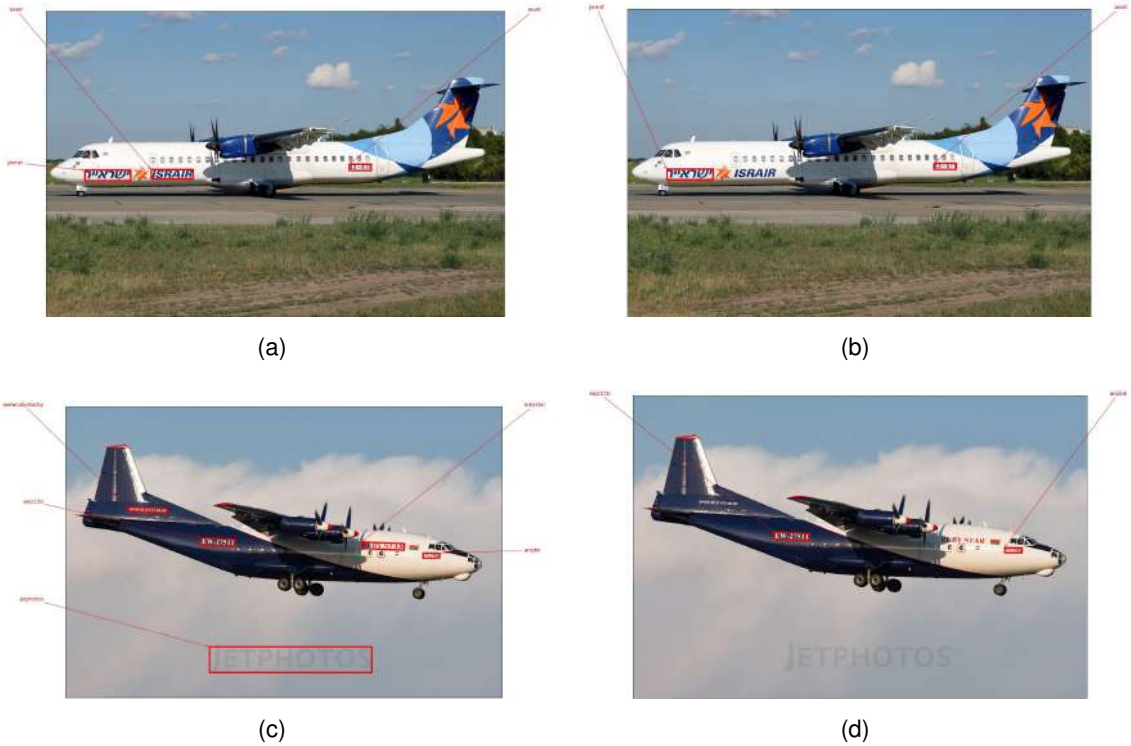


Figura 3.30: Ejemplos de filtrado de ventanas: Izquierda sin filtrar, derecha filtrado. Fotos de [9]

Como se puede ver en las imágenes de la figura 3.30, elimina una sección de ventanas que el sistema detecta como texto por imagen. En el caso de la imagen (a) la sección de ventanas se encuentra debajo de la matricula y en la imagen (c) las sección está encima de la matricula.

- Los aviones suelen tener el nombre de la aerolínea escrito en el fuselaje y es común que el texto pase todos los filtros. Para evitar que textos que claramente no se refieren a matriculas pasen sin ser filtrados se han recopilado más de 5000 nombres de aerolíneas. Cada vez que un texto contenga el nombre de una aerolínea será automáticamente descartado por este filtro.



(a) (b)

(c) (d)

Figura 3.31: Ejemplos de filtrado de palabras clave: Izquierda sin filtrar, derecha filtrado. Fotos de [9]

Como se puede observar, en la primera foto (a) de la figura 3.31filtra el nombre de la aerolínea ”Israir” y en la segunda foto (c) filtra el nombre de la aerolínea ”Ruby star” y la marca de agua de la web propietaria de la foto ”Jetphotos”

CAPÍTULO 4. CADENA DE PROCESOS

Los dos capítulos anteriores han consistido en explicaciones teóricas sobre el funcionamiento de los dos sistemas de *deep learning* que forman parte de la cadena de procesos que permiten traducir una foto en una matricula de manera eficaz. En este capítulo se expone el funcionamiento de la cadena y en los próximos se analizarán algunos de sus componentes en profundidad. Cabe precisar que la cadena que se va a explicar a continuación es la correspondiente al sistema que con una cámara monitorizando el acceso a la pista lleva el control de los despegues y los aterrizajes.

Antes de comenzar a explicar cada etapa de la cadena, conviene tener una visión global de esta: La cadena comienza con la cámara, encargada de tomar las fotos necesarias y subirlas a un servidor. Las imágenes las descarga un cliente a su PC para comenzar el procesado de la misma. EL primer paso del procesado es el algoritmo de detección de objetos, después el sistema OCR y para terminar el sistema de evaluación decide cual es la matricula que aparece en la imagen.

Figura 4.1: Representación esquemática de la cadena de procesado

4.1. De la cámara al servidor

El primer sistema que interviene en la cadena es la cámara, que se encarga de tomar las fotos en los momentos adecuados y de subirlas con un formato determinado para facilitar luego su obtención y procesado. Los pormenores de cómo toma las fotos la cámara están expuestos en el siguiente capítulo.

La cámara toma varias fotos a cada avión que pasa por delante de la cámara asegurando que en más de una foto aparezca el avión entero. Una vez salga el avión del plano de cámara, esta agrupa todas las fotos correspondientes al último avión y las sube al servidor creando en este una carpeta. Los nombres de las fotos corresponden a la fecha y hora en la cual han sido tomadas y el de la carpeta es igual al de la primera foto.

El servidor elegido para subir las fotos es una web llamada ”Cloudinary” la cual tiene una interfaz de programación de aplicaciones muy sencilla que permite subir fotos de manera automática fácilmente. La web provee de tres credenciales únicas a cada usuario que hay que introducir en el código para poder acceder a la ”nube” y subir con comandos sencillos las fotos como `cloudinary.uploader.upload()`.

Para realizar esta operación es imprescindible que la Raspberry Pi tenga acceso a internet, ya sea por la red Wi-Fi del aeropuerto o a través de una red móvil.

4.2. Del servidor al Keras-Retinanet

El servidor de Cloudinary cumple la función de ”puente” entre la cámara que esta situada en la plataforma del aeródromo y el PC situado en la oficina del mismo que es el encargado del procesado para la obtención de la matricula. Para descargar las imágenes del servidor Cloudinary no ofrece las mismas facilidades que para subirlas, no dispone de métodos capaces de descargar en su api.

Antes de comenzar a descargar las fotos que no han sido procesadas, el cliente comprueba que fotos han sido procesadas para no repetir procesos. La comprobación se realiza comparando la lista de carpetas que hay en la nube con la lista de carpetas procesadas que tiene el cliente. La lista de carpetas en la nube es fácil de obtener con la api de Cloudinary mediante el método `cloudinary.api.root_folders()`. La lista de carpetas procesadas se obtiene mediante una búsqueda en la base de datos del cliente.

La base de datos del cliente esta compuesta por dos tablas: La tabla FOTOS almacena datos sobre todas las fotos que se han descargado para ser procesadas y la tabla ENCONTRADAS guarda solo datos de las fotos seleccionadas como se explica en el apartado 4.4 de este capítulo. La tabla FOTOS guarda el nombre de la carpeta en la que está la foto en Cloudinary, el nombre de la foto, la relación entre el área del avión de la imagen con el área de la imagen, el numero de aviones que hay en la foto, la localización de dichos aviones y la matricula detectada.

El comando ejecutado para crear la tabla FOTOS:

```
CREATE TABLE FOTOS (Nombre_carpeta VARCHAR(50), Nombre_foto VARCHAR(50),
Rel_areas VARCHAR(50), Numero_aviones int, Localizacion VARCHAR(100),
Matricula VARCHAR(50))
```

EL resultado de la comparación de listas da como resultado otra lista con los nombres de las carpetas que están en la nube pero que no están en la BBDD y por tanto no han sido procesadas.

A continuación, el cliente recorre la lista de carpetas y realiza la siguiente secuencia de operaciones para cada una de ellas:

- Crea un directorio homonimo en el PC del cliente.
- Ejecuta el método `cloudinary.Search()` para obtener una lista con los nombres de todas las imágenes de la carpeta.
- Usando el método `cloudinary.utils.cloudinary_url()` obtiene la dirección de cada foto para posteriormente descargarla mediante un ”HTTP request”.
- Finalmente se guardan las imágenes en la nueva carpeta y se introducen los datos de nombre de carpeta y de foto en la BBDD para cada una de ellas.

4.3. Selección de fotos

Una vez todas las fotos han sido descargadas y almacenadas en sus correspondiente carpeta, se procesan todas con el algoritmo ”Keras-Retinanet” para detectar objetos y descubrir que objetos hay en las fotos.

En caso de que no haya ningún avión en la carpeta, el cliente elimina la carpeta del PC pero mantiene las filas correspondientes en la BBDD para evitar volver a procesar las fotos.

En caso contrario, se actualiza la BBDD con los datos obtenidos en el procesado: Relación entre el área de cada avión y el área de la imagen, el número de aviones que hay en la foto y la localización de los mismos. La localización viene dada por un vector de cuatro elementos que representan las coordenadas de los vértices superior izquierdo y superior derecho.

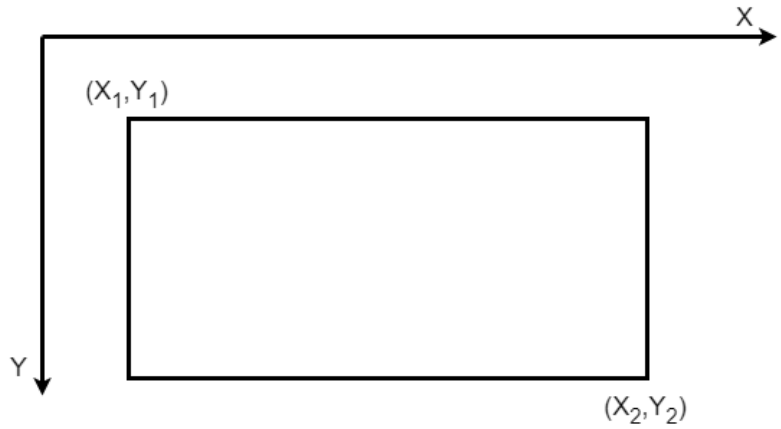
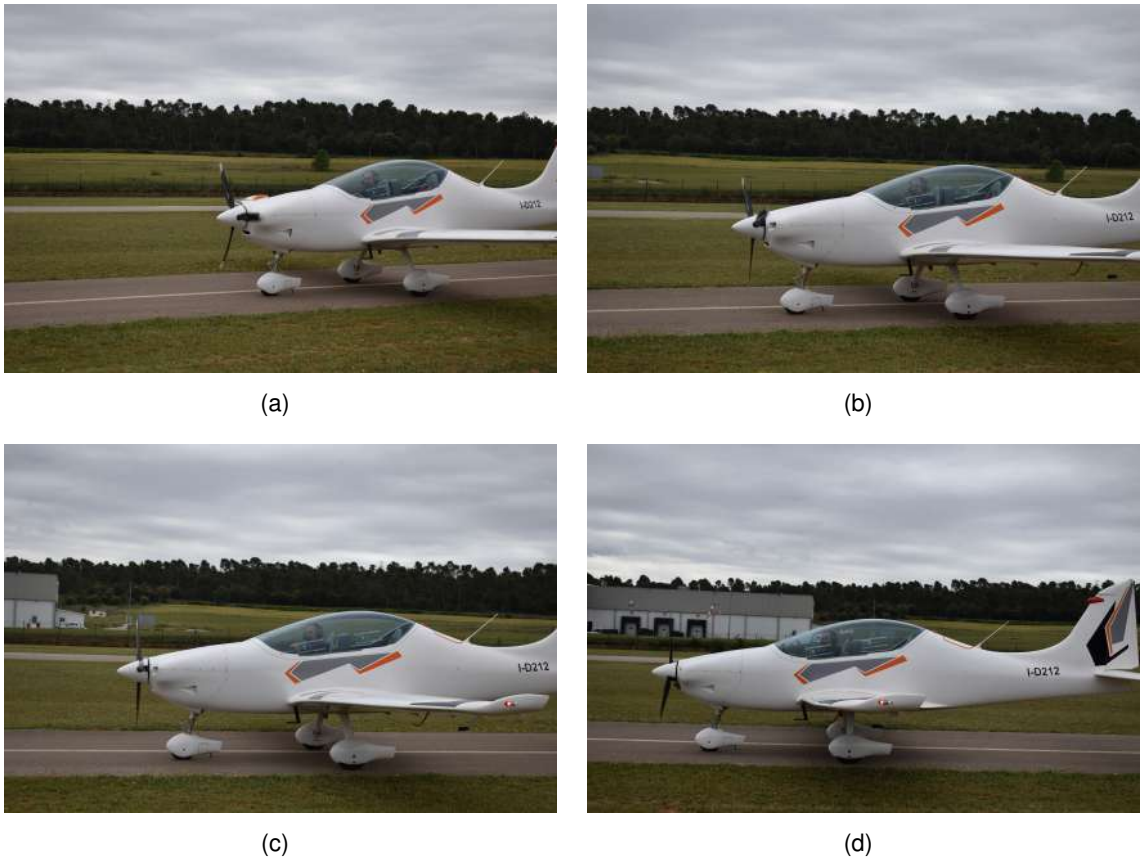


Figura 4.2: El vector que indica las coordenadas siguie el orden [x1,y1,x2,y2]

Después de guardar los datos, se realiza una consulta a la BBDD para obtener una lista con los nombres de las fotos, el número de aviones y la relación de áreas de todos ellos. El objetivo de esta consulta es encontrar las fotos en las que el avión se encuentra centrado y por tanto con su matricula visible. Se modifica la lista para obtener una en la que cada elemento contenga un nombre de foto y una relación de áreas. A continuación se ordena la lista en función de la relación de áreas para seleccionar las tres fotos que tengan el avión más centrado.

Las siguientes sirven de ejemplo del proceso de selección mencionado.



(a) (b)

(c) (d)

Figura 4.3: Entre las cuatro fotos el sistema selecciona la foto (d) como la más adecuada

4.4. Fin del proceso

Seleccionadas las tres mejores fotos, el siguiente paso es que el sistema OCR las procese en busca de secciones de texto. Los resultados arrojados por el OCR los recoge el evaluador y asocia una matricula a cada imagen, además de una puntuación que mide la fiabilidad de la asociación. Se guardan en la tabla FOTOS las matriculas encontradas en las tres fotos y se selecciona la foto con mejor puntuación como resultado definitivo de la predicción.

La segunda tabla de la BBDD ENCONTRADAS sirve para almacenar los resultados definitivos de todo este proceso. La tabla solo contiene dos elementos: El nombre de la foto y la matricula asociada. El proceso termina cuando después de que se guarden el nombre de la mejor foto y su matricula asociada en la tabla ENCONTRADAS la foto ganadora se almacene en el directorio ”matriculas-encontradas” del cliente y se elimine la carpeta descargada.

El comando ejecutado para crear la tabla ENCONTRADAS:

```
CREATE TABLE ENCONTRADAS (foto VARCHAR(50),matricula VARCHAR(50))
```

Se ha creado un método llamado `consulta_resultados()` que sirve para mostrar al usuario todas las matriculas detectadas aportando información sobre la foto en la que se basa la predicción y la fecha de detección obtenida gracias al nombre de la propia foto.

CAPÍTULO 5. OBTENCIÓN DE IMÁGENES

El primer elemento de la cadena de procesado es el que se encarga de tomar las imágenes que pasarán por todos los sistemas que componen la cadena. En este capitulo se expone el método usado para asegurar que la cámara tomará fotos de aviones evitando llenar el almacenamiento del servidor de fotos inservibles. Después, se calculará la posición óptima de la cámara y para terminar, se mostrarán los resultados obtenidos en un ejemplo real.

5.1. Detección de movimiento

La cámara encargada de tomar las fotos de los aviones estará conectada a un ordenador Rasbperry Pi que es el encargado de decidir cuando se toman las fotos y de enviarlas al servidor de *Cloudinary*. La manera que se ha seleccionado para accionar la cámara es el uso de un programa que permite detectar el movimiento.

Existen varios sistemas de detección de movimiento que funcionan con diferentes tecnologías. La mayoría se basan en detectar cambios en señales de microondas o en detectar temperaturas diferentes a las ambientales usando ondas infrarrojas. Incluso se pueden programar usando ultrasonidos midiendo los tiempos de retorno.

Al tratarse de un proyecto de procesado de imágenes, se ha decidido hacer la detección de movimiento a través de la comparación de imágenes basado en el sistema ”pi-motion-lite” [38]. Esta técnica consiste en que la cámara tome una foto que servirá de referencia y luego siga tomando fotos periódicamente para compararlas con la de referencia. La foto de referencia se restablece cada vez que el sistema detecta movimiento.

Para comparar, se recorrerá la imagen píxel por píxel buscando discrepancias. Para que una imagen pueda ser considerada diferente a la de referencia tendrá que cumplir una condición básica: Tener un número determinado de píxeles diferentes. El umbral que marca el número de píxeles necesarios se llama sensibilidad y es un número arbitrario que el usuario tiene que introducir en función de las necesidades.

Existe un segundo umbral que decide cuanto tiene que cambiar un píxel para que este sea considerado diferente. Se trata de otro número arbitrario que el usuario decide. También hay que seleccionar en que componente de la imagen se quiere hacer la comparación: Rojo, Verde o Azul.

Evidentemente, los valores de estos umbrales tienen que tener en cuenta el número de píxeles que hay en las imágenes que toma la cámara y que los posibles valores del píxel van de 0 a 255.

Una vez el sistema detecta movimiento, es decir el número de píxeles que han cambiado su valor más que el umbral es mayor a la sensibilidad, se guarda la imagen que ha cambiado y se continua comparando las imágenes con la de referencia. Esto quiere decir que en caso de que el causante del movimiento sea un avión de camino a la pista, este aparecerá en varias fotos ya que el sistema detectará movimiento varias veces mientras el avión pasa por delante de la cámara.

Como se ha explicado en el párrafo anterior, cada avión que pasa por delante de la cámara hace que la cámara le tome varias fotos. Para agrupar las fotos de un mismo avión y diferenciar este grupo de otros posteriores, se ha empleado un contador que se activa tras tomar una foto. Si dicho contador alcanza el valor de segundos que el usuario ha decidido, se considera que el avión ha pasado y guarda todas las fotos en una carpeta que subirá al servidor. En caso de que la cámara saque otra foto antes de llegar al limite del contador, este vuelve a cero y comienza la cuenta de nuevo. De esta manera, el cliente al descargarse una carpeta del servidor sabe que esta obteniendo todas las fotos de un mismo avión. Como se ha explicado en este capítulo, la foto de referencia se restablece cada vez que se sube una carpeta al servidor.

Resumiendo, se trata de una manera sencilla y adaptable de detectar cambios en el entorno ofreciendo la posibilidad de ajustar cuanto tiene que cambiar un píxel, cuantos píxeles tienen que cambiar y cuanto tiempo dura un cambio.

5.2. Localización de la cámara

Para obtener las fotos más convenientes es crucial colocar la cámara en una posición que permita fotografiar al avión completo. Para lograrlo hay que colocarla a la distancia adecuada de una calle por la que pasen los aviones que van a la pista. Es importante que la cámara este apuntando a una calle de rodadura porque de esta manera se asegura que el avión no se esta moviendo a velocidades excesivas para ser capturado en una foto.

A continuación se expone un ejemplo de cómo decidir la ubicación de la cámara, se ha tomado como ejemplo el aeródromo de Sallent-Pla de Bages.

La cámara que se utiliza para este ejemplo es la ”Raspberry Pi camera module v2” que tiene 62,2º de visión horizontal. Teniendo en cuenta esto y que la mayoría de los aviones que usan el aeropuerto miden como mucho 9 metros de largo, se puede obtener la distancia necesaria a la que tiene que colocarse la cámara para obtener un plano en el que se vea al avión entero. En este caso, no interesa apurar las medidas para que el avión quepa justo en el plano, si no que conviene dejar un margen para que el sistema de detección de movimiento sea capaz de tomar al menos una foto del avión entero. El margen es necesario porque el avión esta en movimiento y en caso de no tener margen, el sistema no tiene margen de error a la hora de temporizar el momento exacto en el que tomar la foto.

Por lo tanto, el calculo de la distancia a la que colocar la cámara se hará de tal manera que entre en el campo de visión una distancia de 1.5 veces la longitud máxima, es decir 13,5 metros. La distancia entre el avión y la cámara resultante es de 8,8 metros. La ubicación de la cámara, como se puede ver en la figura, es la idónea para fotografiar a todos los aviones que acuden a la pista.

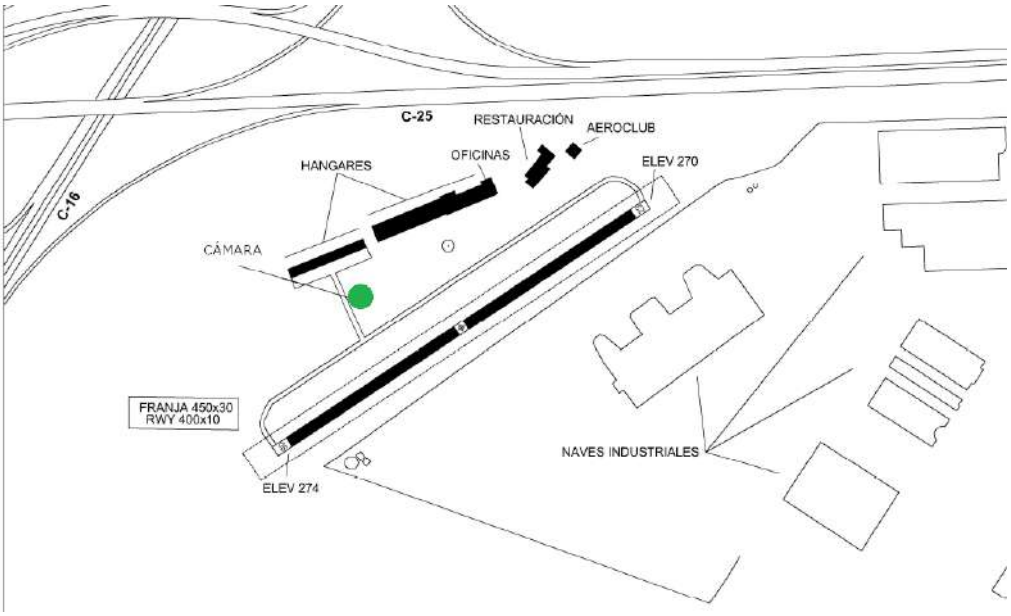


Figura 5.1: Ubicación de la cámara en el aeródromo. Fuente: [25]

5.3. Pruebas de funcionamiento

Para probar el funcionamiento del detector de movimiento y la cámara, se han realizado varias pruebas en el aeródromo de Sallent-Pla de Bages.

Para realizar estas pruebas se han decidido tomar un umbral de cambio de nivel del componente verde de cada píxel de 15 y una sensibilidad de 150 píxeles. Puede parecer que una sensibilidad de 150 en una foto con 786432 píxeles es un umbral bajo pero después de hacer unas pruebas con una sensibilidad de 300 se decidió que estos valores eran los más adecuados. También se ajustó el tiempo es espera entre objetos a diez segundos pese a que este parámetro no tuviera importancia por reducido volumen de tráfico del aeródromo.

La cámara escogida para estas pruebas es la ”Raspberry Pi camera module v2”, se ha escogido por ser la mas adecuada para ser usada con el ordenador encargado de ejecutar el detector de movimiento y por su pequeño tamaño. Se trata de una cámara con 8 Mpíxeles de resolución de fácil conexión al ordenador.

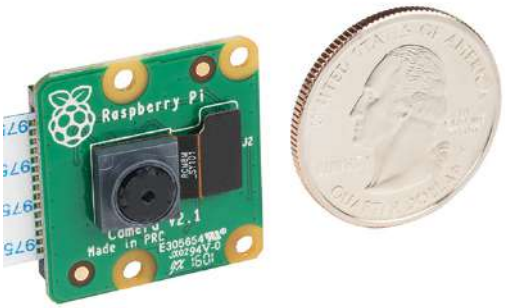


Figura 5.2: Comparación entre la cámara ”Raspberry Pi camera module v2” y una moneda de 0.25 dolares la cual tiene un diámetro que es un milímetro mayor a la de un euro. Fuente: [17]

La prueba consistió en dejar la cámara enfocando a la calle de acceso a la pista para que tomara fotos de las avionetas que pasasen por delante. A continuación se pueden ver dos fotos tomadas a avionetas aparcadas:

Figura 5.3: Avioneta con matricula ”EC-FN1”

Figura 5.4: Avioneta con matricula ”I-D212”

Se muestran fotos de las avionetas en estático porque la cámara utilizada no estaba correctamente configurada para tomar fotos de las avionetas en movimiento. Se puede afirmar que la cámara seleccionada es capaz de tomar fotos con la calidad requerida en el caso de que esté correctamente configurada gracias a los ejemplos de fotos que hay en internet.

A continuación se muestran las fotos defectuosas tomadas el día de la prueba ya que sirven para confirmar que el sistema detector de movimiento funciona correctamente y toma las fotos en el momento adecuado

(a) (b)

Figura 5.5: Las imágenes están completamente quemadas pero se puede apreciar que han sido tomadas en el momento adecuado.

Pese a la mala calidad de las imágenes se puede apreciar que el detector de movimiento funciona correctamente y es capaz de tomar las fotos en el momento adecuado. Ese mismo día además de usar la cámara de Raspberry, se sacaron fotos a los mismos aviones con otra cámara para después hacer comparaciones. Dicha cámara es la Nikon d3500 y el objetivo es el 18-55mm f 3.5:5.6. Se trata de productos de gama baja entre las cámaras réflex lo que demuestra que con una cámara de gama baja sería suficiente para tener un sistema totalmente operativo. Las fotos tomadas con la cámara Nikon se muestran a continuación:

(a) (b)

Figura 5.6: La calidad de las imágenes aumenta notablemente pese a que los aviones no esten quietos

CAPÍTULO 6. PREDICCIÓN DE MATRICULA

La ultima etapa de la cadena de procesado es el evaluador o predictor de matricula. El cometido de este sistema es decidir que matricula es la que hay en la foto a partir de las predicciones aportadas por el sistema OCR. Esta etapa es necesaria porque las predicciones no siempre son iguales a las matriculas y requieren de un sistema como este para que el texto incorrecto pueda ser asociado a la matricula de la imagen.

Para esto, el evaluador dispondrá de la lista de matriculas de los aviones que tienen plaza para estacionar en el aeropuerto y de los que van a realizar alguna operación en este. Así, el evaluador ”comparará” el texto de la predicción del OCR con cada matricula para determinar cual es la que más se parece.

6.1. Distancia entre cadenas de texto

Calcular la similitud de dos cadenas de caracteres es una operación que tiene varias posibles maneras de realizarse y por tanto muchos algoritmos que siguiendo procedimientos diferentes calculan distancias entre cadenas de texto obteniendo cada una su resultado. A continuación se exponen dos de los algoritmos más conocidos [18]:

6.1.1. Distancia de Hamming

El método propuesto por el matemático estadounidense Hamming es el más sencillo y por tanto el más limitado. El cálculo de la distancia consiste en tomar las dos cadenas de texto e ir comparándolas elemento por elemento, cada discrepancia suma un punto. El numero de puntos es la distancia entre las cadenas de texto. Es una manera sencilla de calcular la distancia ya que no requiere de mucha capacidad computacional, pero esta limitada a palabras de la misma longitud.

El hecho de que no pueda comparar palabras que no tengan la misma longitud es una característica que descarta el uso del algoritmo para este proyecto: Las predicciones que el OCR hace no siempre coinciden con la longitud de la matricula que se muestra en la imagen, ya sea porque omite algún símbolo como el guión o que simplemente comete algún error.

6.1.2. Distancia de Levenshtein

El matemático soviético Levenshtein propuso un método mas complejo para calcular las distancias. Al igual que el de Hamming, el algoritmo de Levenshtein se basa en detectar discrepancias entre las dos cadenas, pero a diferencia del primero, este tiene en cuenta operaciones como inserciones y eliminaciones lo cual permite comparar cadenas de varias longitudes. Por lo tanto, en vez de buscar discrepancias letra por letra, el algoritmo calcula cuantas operaciones de inserción o eliminación hay que realizar para transformar una cadena en la otra.

Por ejemplo, la distancia entre ”casa” y ”calle” es de tres ya que para transformar casa en calle hay que cambiar la ”s” por la ”l”, la ”a” por la ”e” e insertar una ”l” en la tercera

posición. Se considera que este método es la generalización del creado por Hamming y existe una relación entre ambos algoritmos: La distancia Levenshtein será siempre igual o menor a la Hamming.

La distancia de Levenshtein cumple con los requerimientos para comparar las predicciones del OCR con las matriculas pero su rendimiento no ha resultado satisfactorio.

6.2. Rendimiento de la distancia Levenshtein

Para medir la eficacia del algoritmo de Levenshtein, se ha seleccionado un grupo de 87 imágenes de aviones llamado ”real-images”. Para la comparación se generan dos listas: La primera contiene las 87 matriculas de los aviones y la segunda los resultados obtenidos tras procesar las 87, es decir, las predicciones que ha hecho el OCR. A continuación se compara cada predicción con todas las matriculas y se selecciona como correcta la que menos distancia tenga.

El resultado fue malo para ambos sistemas OCR, tanto ”Easy-OCR” como ”Keras-OCR” los cuales obtuvieron un porcentaje similar de acierto: 16/87 y 17/87 respectivamente. Son porcentajes de acierto menores al 20 % lo cual significa que no es una manera fiable de identificar aviones ya que hay mas errores que aciertos. Al realizar esta evaluación se observó que el sistema ”Keras-OCR” cometía errores de predicción muy parecidos en muchos casos y por ello se decidió crear un medidor de distancia a medida para mitigar los errores mas habituales de este sistema. El sistema ”Easy-OCR” queda descartado porque sus errores no parecen seguir un patrón claro y esto dificulta la corrección de los mismos.

6.3. Distancia propia

Como se ha mencionado en el apartado anterior, este sistema de cálculo de distancias esta pensado para mejorar el rendimiento ofrecido por la distancia Levenshtein y corregir los errores habituales cometidos por ”Keras-OCR”.

En los ejemplos de postprocesado del OCR se ha podido ver que hay fotos en las que por mucho que se apliquen todos los filtros hay textos que pasan el corte y no son matriculas, esto provoca que una foto pueda tener varios textos asociados cuando llegue a la etapa del cálculo de distancias. Para ello se ha creado un sistema de dos etapas: En la primera se obtiene la matricula más parecida a cada texto de la imagen y en la segunda se decide cual de estas matriculas se parece más al texto obtenido por el OCR.

Para obtener la matricula más parecida a la predicción del OCR se envía esta junto a la lista de matriculas a una función llamada `arbol_distancias()` que como su nombre sugiere, se trata de un árbol de decisión en el cual dependiendo de factores como la longitud de la predicción, se selecciona el método con el que se calculará la distancia. Cada matricula tendrá una puntuación y la que tenga la menor de todas será la considerada correcta. Cabe destacar que en este algoritmo las puntuaciones no siempre representan el número de operaciones hay que realizar o el número de discrepancias existentes si no que tratan de crear una jerarquía entre los errores.

- La primera condición que evalúa el árbol de distancias es la más sencilla y obvia y esta pensada para ahorrar tiempo de computación: Si la predicción y la matricula son iguales, la puntuación será de -1.

- La segunda condición que se comprueba esta basada en un error que comete el sistema OCR: Al no estar entrenado para ello, ”Keras-OCR” no reconoce los guiones los cuales son un elemento bastante común en las matriculas de aviones. Por ello, tras verificar que las dos cadenas de caracteres no son iguales se verifica si la predicción y la matricula que se están comparando son iguales en el caso de que se elimine el guión que puede tener la matricula. Si efectivamente la matricula sin guión y la predicción son iguales, la puntuación será de -1. Se le otorga la misma jerarquía ya que se trata de un error menor y que el resto de los elementos son iguales en el mismo orden.

- Una vez se ha verificado que la predicción no es igual a la matricula, se mira si las longitudes de las dos cadenas de caracteres son iguales. En caso afirmativo, la distancia se calcula con una función llamada `distancia_simple()`. Esta función es una adaptación de la distancia de Hamming a la comparación de matriculas. Es una adaptación porque depende de la discrepancia la función suma 0, 1 o 1,1 al computo de distancia entre las dos cadenas. Si la discrepancia es causada por un guión, no se sumará nada a la distancia. Si la discrepancia la crea una de las siguientes parejas de caracteres tampoco se suma nada a la distancia ya que se trata de errores comunes: ”0” y ”o”, ”a” y ”4”, ”z” y ”7”, ”l” y ”i”, ”g” y ”9”, ”s” y ”5”, ”i” y ”1” o ”b” y ”8”. Por último, si la discrepancia se da en el primer o último elemento de la cadena se suma 1.1 ya que se ha detectado que en la mayoría de los casos las primeras y últimas letras coinciden en las matriculas correctas. La función realiza este procedimiento con las cadenas tal y como están y del revés y devuelve la menor de las distancias.

  Una vez calculada la distancia con `distancia_simple()`, se devuelve como puntuación el valor obtenido menos 0.5 para bonificar los casos en los que las longitudes coinciden.

- Si la longitud de la cadena de la matricula es igual que la de la predicción más uno, se abre un abanico de tres posibilidades para calcular la distancia. En estos casos la puntuación es igual a la distancia calculada:

  - Si la matricula contiene un guión, la distancia entre las cadenas de caracteres será la que ofrezca `distancia_simple()`.
  - Si la cadena de la predicción esta incluida en la matricula con el mismo orden, la distancia será de 1. Un ejemplo de esto sería si la matricula es ”568ACJ” y la predicción ”68ACJ”.
  - Por último, se devolverá la menor de las distancias que se obtengan al comparar la predicción con la matricula recortada en dos maneras. La primera comparación será entre la predicción y la matricula sin su última letra y la segunda entre la predicción y la matricula sin su primera letra. Dichas comparaciones se harán con la función `distancia_simple()`.

- Si la cadena de caracteres de la matricula es 2 o más elementos más larga que la predicción se llega a esta condición en la cual se devuelve una puntuación de 1 en

caso de que la predicción este contenida en la matricula de la misma manera que se ha expuesto en la condición anterior.

- En caso de que no se haya cumplido ninguna de las condiciones que se han explicado hasta el momento, la puntuación se calculará mediante la función `coincidencia()`, la cual ofrece tres maneras diferentes de obtener la puntuación según unos requisitos que se desglosan a continuación:
  - La primera condición es triple y con que ambos textos cumplan una de ellas basta. La primera es que los dos elementos iniciales y finales de las dos cadenas coincidan, la segunda es que los cuatro primeros elementos coincidan y la tercera es que los cuatro últimos coincidan. Si se cumple cualquiera de estos tres casos, la puntuación será de 1,75.
  - La segunda condición es parecida a la primera: En caso de que los m elementos iniciales o finales de las cadenas coincidan, la puntuación será de 1,75, donde m es la longitud de la cadena de caracteres de la predicción.
  - Por último, si no se ha encontrado ninguna similitud entre ambas cadenas, la puntuación devuelta será de 10.

Normalmente, la puntuación obtenida por la matricula correcta es menor de 2,5. El conjunto de condiciones expuestas se ha confeccionado usando *fine tuning* y un conjunto de 350 predicciones y matriculas.

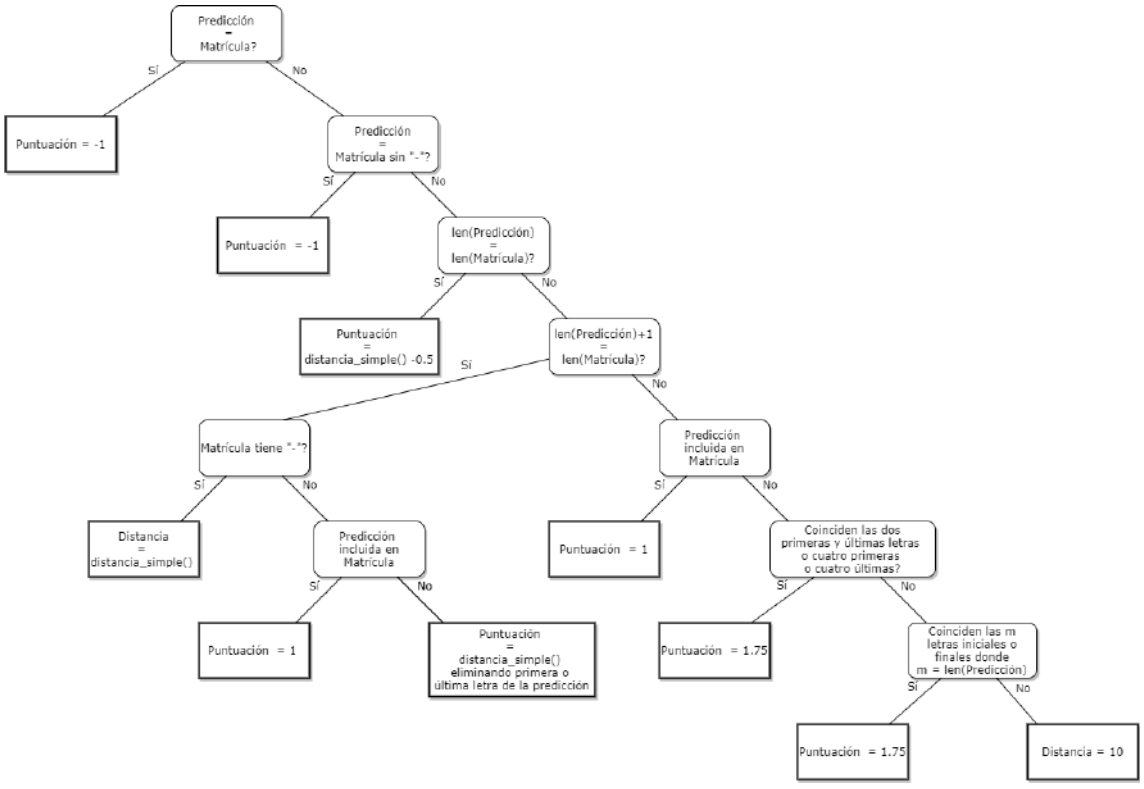


Figura 6.1: Orden correcto de los puntos

6.4. Rendimiento de la distancia propia

El rendimiento de la función de distancia ha ido mejorando a medida que se le han añadido nuevas condiciones. Para el grupo ”real-images” la función `distancia_simple()` tuvo un acierto de 30/87 o lo que es lo mismo, de un 34 % mejorando notablemente la distancia de Levenshtein.

Tras obtener el 34 % se decidió entrenar el sistema ”Keras-OCR” utilizando el código que el autor ofrece para que fuera capaz de reconocer los guiones y así mejorar los resultados. Pero para poder reconocer los guiones hay que entrenar tanto el detector como el reconocedor lo cual requiere de una capacidad de computación que no esta al alcance de este proyecto. Pese a disponer de los ordenadores que ofrece Google a través de Colab, estos solo permiten ejecuciones de 12h y para poder entrenar el OCR un número de ciclos adecuado harían falta demasiadas ejecuciones.

Visto este problema, se decidió re-entrenar el sistema de reconocimiento para obtener mejor rendimiento a la hora de reconocer los diferentes elementos que no sean guiones que componen la matricula. Pero pese a entrenarlo por más de 24h, la eficacia de la nueva red neuronal fue muy inferior a la predeterminada: En ese momento, se habían hecho mejoras en el evaluador que permitieron obtener 61/87 asociaciones correctas con la red predeterminada y con la entrenada se obtuvieron 20/87 con la mejor de las cinco redes diferentes que se probaron a entrenar.

Para aumentar la casuística, se creó un nuevo grupo de 356 imágenes llamado ”Demo” en el que estaban incluidas las de ”real-imagenes”. Gracias a los nuevos tipos de errores mostrados por ”Demo” se afinó el sistema para pasar de un 58,5 % inicial a un 82,3 % o lo que es lo mismo, 293/356. Las 63 predicciones que no se han asignado correctamente son en su mayoría textos que nada tienen que ver con la matricula y que resulta imposible asociar a ninguna de ellas siguiendo un proceso lógico.

Para terminar, se creó un ultimo grupo de imágenes el cual esta compuesto íntegramente por fotos tomadas en el aeródromo de Igualada. Para este grupo de imágenes el resultado fue altamente satisfactorio ya que el sistema asoció correctamente el 90,9 % de las 66 fotos que componen el grupo.

CAPÍTULO 7. IMPACTO

7.1. Impacto aeronáutico

En los capítulos anteriores se ha detallado el funcionamiento del detector automático de matriculas y solo por encima se han mencionado las razones de ser y los efectos que provoca la existencia de un sistema como este. El capítulo de impacto trata de exponer cómo y dónde sería adecuado el uso del sistema y para respaldar las propuestas se proporcionarán los datos pertinentes.

Desde su creación a principios del siglo XX la aviación comercial y recreativa ha ido aumentando el número de pasajeros y/o aficionados año a año gracias al avance de la tecnología y al abaratamiento de los precios. Este aumento ha provocado que la red de aeropuertos y aeródromos de España este formada por 51 aeropuertos y alrededor de 350 aeródromos.

En la mayoría de los aeropuertos los vuelos son controlados lo que quiere decir que desde la torre de control se dirige el tráfico y a través de los transpondedores y la comunicación por radio los aviones se identifican aportando entre otros datos su matricula. En los aeropuertos y aeródromos no controlados la identificación de los aviones no es automática y en algunos casos ni siquiera se controla la matricula de los aviones que aterrizan o despegan.

A esos aeródromos/aeropuertos va dirigido el proyecto cuyo objetivo es ayudar a identificar a los aviones que usan la pista y los que están estacionados sin necesidad que un miembro del personal este continuamente observando la pista o recorriendo los hangares para tomar nota de los aviones que realizan las diferentes actividades que ofrece la infraestructura. El hecho de conocer que aviones son los que usan las instalaciones no solo puede aportar beneficio económico al administrador a través de cuotas por uso, si no que al crear un registro de actividades, facilita el proceso de localizar el avión en caso de accidente.

De todos los aeródromos que hay en España se ha seleccionado el de Teruel como el más adecuado, pero además de este se expondrán más casos:

7.1.1. Aeropuerto de Teruel

El aeropuerto privado de Teruel, inaugurado en 2012, es el ejemplo perfecto de aeropuerto no controlado con gran capacidad de aparcamiento para aviones. Tanto es así que el aeropuerto de Teruel es ya el aeropuerto de Europa con mayor capacidad de aparcamiento para aviones con 130 plazas las cuales tiene comprometidas ya que en el mes de mayo de 2021 se ha autorizado el proyecto de ampliación de la campa que sirve como aparcamiento para poder alojar hasta 180 aviones de todos los tamaños. Esta ampliación de la campa es la tercera que se realiza en los últimos tres meses lo que indica que la demanda para aparcar en Teruel es alta debido principalmente a la ubicación y al clima seco y frío que favorece la conservación correcta del avión durante la estancia. Según explica a el ”Heraldo de Aragón” el director del aeropuerto Alejandro Ibrahim, la duración de la estancia de un avión en el aeropuerto es de uno a dos años [20].

Figura 7.1: Aparcamiento de aviones de Teruel. Fuente: [19]

Las estadísticas de uso del aeropuerto no son públicas lo que dificulta el análisis y cuantificación del número de matriculas que se pueden detectar en las múltiples funciones que puede realizar un sistema así en un aeropuerto como este.

Al ser un aeropuerto no controlado y ya que las operaciones de despegue y aterrizaje no son la principal actividad del aeropuerto, el uso del sistema para llevar el control de las operaciones ahorraría tiempo y trabajo para la gestión del aparcamiento. Además de controlar las operaciones de pista, se pueden colocar cámaras en diferentes puntos de la plataforma para registrar los movimientos que se dan en esta como por ejemplo controlar el acceso a hangares. Resulta especialmente importante en un aeropuerto con un aparcamiento tan extenso el controlar que avión se encuentra en el hangar reparándose o repostando. Por último, el control de la campa de aparcamiento se podría realizar mediante la colocación de postes con cámaras que permitan detectar varios aviones a la vez.

El aeropuerto de Teruel tiene plazas de aparcamiento para aviones de todos los tamaños, esto implica la cámara no podrá estar situada tan cerca de la calle como en la estimación realizada para el aeródromo de Sallent-Pla de Bages. El aumento de la distancia entre la cámara y los aviones exige que la calidad de esta sea lo suficientemente buena como para poder fotografiar las matriculas más pequeñas.

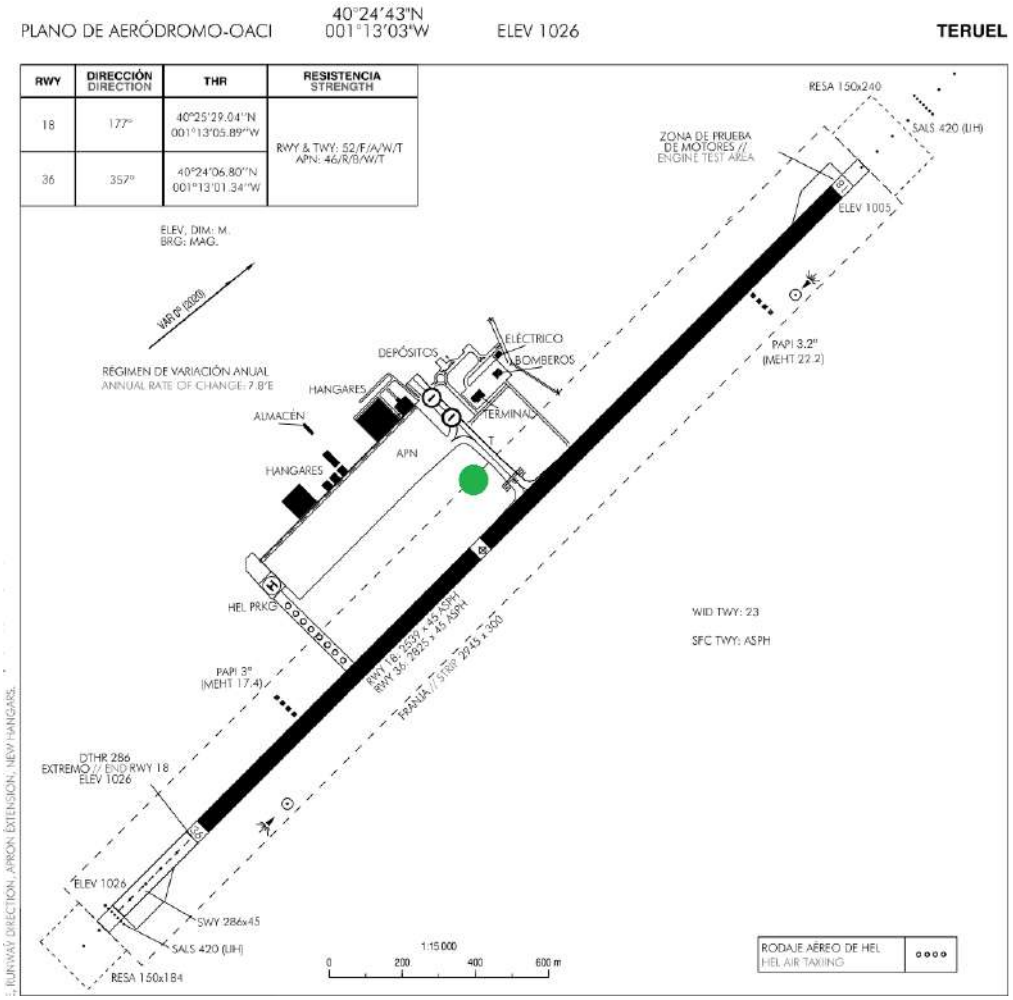


Figura 7.2: Ejemplo de localización de la cámara que controla las operaciones en el aeropuerto de Teruel indicado con un punto verde. Fuente: [16]

7.1.2. Otros ejemplos

Existen otros aeropuertos en España que también son adecuados para el uso de este sistema por ser no controlados y por que como el de Teruel sirven de estacionamiento para los aviones.

7.1.2.1. Aeropuerto de Andorra-La Seu

El aeropuerto de Andorra-La Seu es un aeropuerto no controlado en el cual los usuarios tienen que pagar una tasa por uso, por lo que actualmente se lleva el control de las operaciones. Se trata de un aeropuerto con trafico constante incluso en los meses afectados por la pandemia: Comparando con 2019, los únicos meses con menos operaciones en 2020 fueron marzo, mayo, septiembre y diciembre. Además de eso, el promedio de operaciones mensuales aumentó de 329,75 a 443,75 y en los primeros tres meses del año 2021, la media mensual ha aumentado hasta las 622,33 operaciones al mes. No cabe duda de que el aeropuerto de Andorra-La Seu es un aeropuerto pujante en el que el número de operaciones anuales no para de aumentar.

Cuadro 7.1: Número de operaciones por mes en Andorra La Seu. Fuente: [21]

Mes	**2021**	**2020**	**2019**
Enero	335	468	266
Febrero	759	541	263
Marzo	773	167	298
Abril		250	141
Mayo		240	405
Junio		719	302
Julio		765	393
Agosto		484	285
Septiembre		381	549
Octubre		510	494
Noviembre		587	320
Diciembre		213	241
Media	622	444	330

Vista la demanda cobra sentido el uso de un sistema automático de reconocimiento de matriculas para poder absorber la demanda creciente y seguir cobrando tasas por uso del aeropuerto. En el caso de este aeropuerto, la localización de la cámara es fácil de decidir ya que al tener una sola pista conectada en un extremo con el aparcamiento y todos los servicios, basta con colocar una cámara en la calle de salida de ese extremo de pista para fotografiar a todos los aviones que realicen alguna operación.

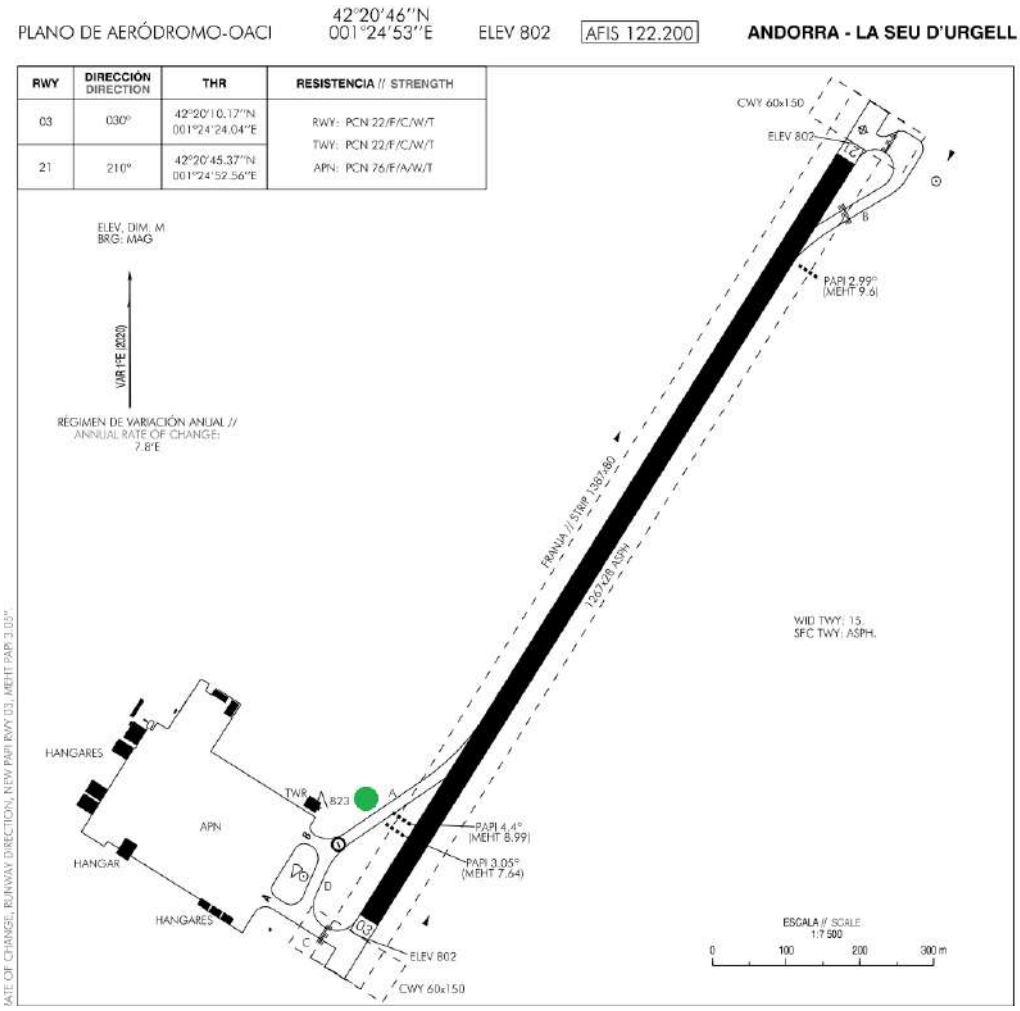


Figura 7.3: Ejemplo de localización de la cámara que controla las operaciones en el aeropuerto de Andorra La Seu indicado con un punto verde. Fuente: [16]

7.1.2.2. Aeropuerto de Castelló

El aeropuerto de Castellón es un caso intermedio entre los dos explicados hasta ahora, se trata de un aeropuerto con gran volumen de tráfico para tratarse de uno no controlado y una capacidad de aparcamiento de 71 aviones. En este caso, los usuarios no pagan tasas por uso. El número de operaciones siguen una clara curva ascendente la cual no se ha visto afectada para nada por la pandemia: Pasando de las 542 operaciones mensuales en 2019 a las 646 en 2020.

Cuadro 7.2: Número de operaciones por mes en Castelló. Fuente: [22]

Mes	**2020**	**2019**	**2018**
Enero	1141	249	100
Febrero	1564	297	72
Marzo	640	371	88
Abril	16	385	128
Mayo	16	832	186
Junio	143	504	232
Julio	791	843	164
Agosto	705	856	242
Septiembre	872	649	339
Octubre	671	701	393
Noviembre	685	591	221
Diciembre	506	537	280
Media	646	542	203

Igual que en el aeropuerto de Andorra-La Seu, se cree conveniente la instalación de un sistema de reconocimiento automático en la calle de acceso que conecta la plataforma con la única pista del aeropuerto.

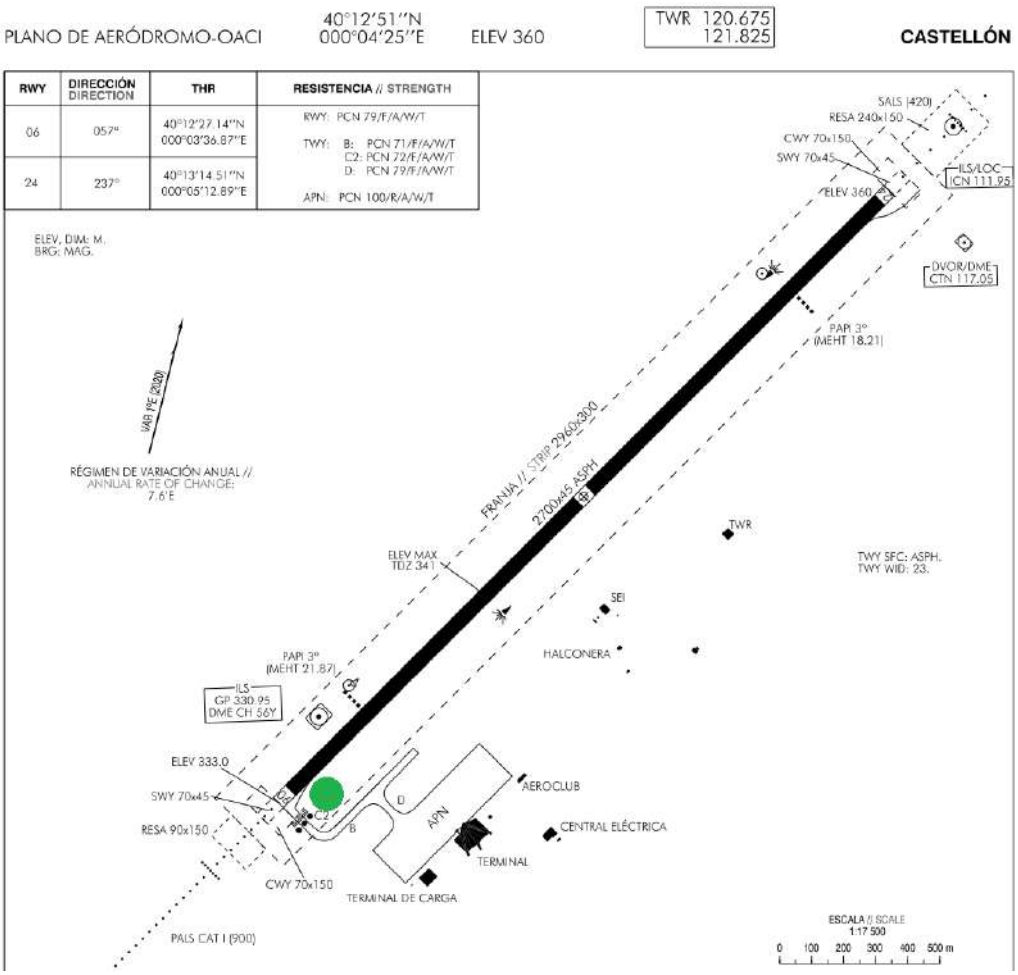


Figura 7.4: Ejemplo de localización de la cámara que controla las operaciones en el aeropuerto de Castelló indicado con un punto verde. Fuente: [16]

7.2. Impacto medioambiental

Es importante evaluar el sistema desde el punto de vista medioambiental ya que este es un aspecto casi tan crucial como el propio correcto funcionamiento debido a la situación actual del planeta. El sistema tiene varias maneras de reducir su impacto medioambiental las cuales se pueden agrupar en dos bloques: Fuentes de energía y ahorro de energía.

En cuanto al ahorro de energía, la principal ventaja que presenta el sistema es su rápida inicialización que permite apagar y encender la raspberry pi en los periodos en los que no haya tráfico. Por ejemplo, en los aeródromos pequeños como el de Sallent-Pla de Bages no hay trafico por la noche porque el aeródromo permanece cerrado por lo que se puede mantener el sistema apagado mientras el aeródromo este cerrado. En aeropuertos como el de Teruel que no tienen gran volumen de tráfico también se podría apagar el sistema para reducir gasto energético en las horas en las que no se prevea ninguna operación.

Por otro lado, la raspberry pi es un ordenador que requiere 5V y 3A para poder funcionar, lo cual permite que sea alimentado por baterías portátiles como si de un teléfono móvil se tratara. Aprovechando la baja demanda energética del la raspberry pi se pueden plantear dos maneras de alimentar el sistema mediante energía renovable. La primera manera sería usando paneles solares y baterías para sacar provecho de las horas de sol en las zonas en las que el clima lo permita. La alternativa de la energía solar es viable gracias a que en la plataforma de los aeropuertos no suele haber objetos que puedan hacer sombra a los paneles. La segunda manera es la más viable: Alimentar la raspberry pi utilizando energía eólica. La principal ventaja de esta manera de obtener la energía es que los aeropuertos están situados en zonas ventosas en las que el viento suele soplar en una o dos direcciones claramente predominantes.

En resumen, si se alimenta con energía de fuente renovable y se mantiene encendido solo en los periodos necesarios, el sistema tiene un impacto mínimo en cuanto a consumo. Además, si se protege correctamente de las inclemencias climáticas, la durabilidad del sistema es alta.

CAPÍTULO 8. CONCLUSIONES Y MEJORAS

La conclusión principal que se extrae de este proyecto es que el reconocimiento automático de matrículas de aviones a través del procesado de imágenes es la manera más eficaz y sencilla de controlar los movimientos de un aeropuerto. Se llega a esta conclusión por los diversos factores que hacen de este sistema el mejor comparado con los transpondedores que se usan hoy en día en la aviación.

El primero de los factores es el equipamiento necesario. Con el sistema actual, los aviones tienen que tener incorporado un transpondedor para transmitir su matricula al aeropuerto y a su vez, el aeropuerto tiene que estar provisto de la infraestructura necesaria para poder recibir y procesar las señales. En la detección por imágenes, la única infraestructura necesaria son las cámaras que se colocan para tomar las fotos. El resto del equipamiento necesario para hacer funcionar el sistema es un ordenador y un servidor, el ordenador lo tienen todos los aeropuertos y como servidor se puede usar una web de almacenamiento de archivos en la nube por lo que no hace falta comprar físicamente uno.

Otro factor clave es el sencillo mantenimiento, en caso de que algo falle en el sistema, solo hay que acudir al punto de la plataforma donde se ha situado la cámara y solucionarlo. Para añadir redundancia, se podrían situar dos cámaras una a cada lado de la calle que se quiera fotografiar. El hecho de que todo el sistema se encuentre permanentemente en tierra facilita enormemente cualquier tipo de reparación sin causar retrasos.

Pero la mayor ventaja de la que dispone este sistema es su amplio margen de mejora. Los sistemas de *deep learning* son cada vez mas populares lo que hace pensar que dentro de unos años la eficacia de los mismos habrá aumentado considerablemente. A continuación se hace mención de posibles mejoras concretas de este sistema.

La primera posible mejora ya se ha mencionado en el capítulo de detección de movimiento: Pese a que la cámara de la Raspbrry Pi correctamente configurada pueda dar buenos resultados, en casos en los que la cámara tiene que estar más alejada del lugar de paso de los aviones es probable que se requiera de una mejor cámara. El uso de una buena cámara es crucial, porque es la manera en la que se obtienen los elementos que el sistema tiene que usar para obtener un resultado.

Tener servidores propios sería una mejora importante, así en vez de depender de unos servidores ajenos se gestionarían los datos de la manera mas adecuada para el sistema. Tener un servidor propio evita tener que pagar por capacidad de almacenamiento a un tercero y permitiría tener una base de datos centralizada en la que guardar todos los registros.

Pero sin duda la mejora más trascendente se tiene que dar en los algoritmos. El primer paso sería poder disponer de ordenadores con la suficiente capacidad como para entrenar el sistema actual para que pueda detectar guiones y mejore su capacidad de reconocimiento. A partir de ahí todas las mejoras que pueda experimentar el campo del *deep learning* será implementada. Una de las mejoras a futuro que más beneficio podría traer sería el hecho de que los algoritmos pudieran ser más ligeros en cuanto a uso de memoria y poder ser ejecutados en las Raspberry-pi que hacen funcionar a la cámara.

La gran versatilidad del sistema hace pensar que puede ser usado en otras situaciones si se realizan las adaptaciones pertinentes. Si se adaptan los filtros del postprocesado y se prepara al sistema de detección de objetos para detectar barcos, se podría utilizar para controlar el acceso a un puerto, por ejemplo.

Todo el código puede ser descargado en https://github.com/MartinMarcellan/Reconoce_matriculas

BIBLIOGRAFÍA

[1] Real academia española, ”definición algoritmo”
https://dle.rae.es/algoritmo 3

[2] F.Chollet, ”What is deep learning?”. *Deep learning with python*. (Manning. Shelter island. 2018): 1–25. xiii, 3

[3] F.Tarres, ”Point Transforms”.*Image Processing and Computer Vision*
https://atenea.upc.edu 7

[4] O’Reilly, ”Detection or localization and segmentation”
https://www.oreilly.com/library/view/deep-learning-for/9781788295628/4fe36c40-7612-44b8-8846-43c0c4e64157.xhtml xiii, 10

[5] datascience, ”Un recorrido por AlexNet”
https://datascience.eu/es/programacion/un-recorrido-por-alexnet/ xiii, 11

[6] computersciencewiki, ”Max-pooling / Pooling”
https://computersciencewiki.org/index.php/Max-pooling_/_Pooling xiii, 12

[7] ArcGIS Developers, ”How RetinaNet works?”
https://developers.arcgis.com/python/guide/how-retinanet-works/ xiii, 13, 14

[8] GitHub, ”Keras-retinanet”
https://github.com/fizyr/keras-retinanet xiii, 15, 16

[9] Jetphotos
https://www.jetphotos.com/ xiii, xiv, 16, 17, 18, 25, 31, 34, 36, 39, 42, 43, 44, 45, 46, 47

[10] X.Zhou,C.Yao,H.Wen,Y.Wang,S.Zhou,W.He y J.Linag, ”EAST: An Efficient and Accurate Scene Text Detector”, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 2642–2651, 2017. xiii, 21, 22, 23

[11] Medium, ”Non Max Suppression (NMS)”
https://medium.com/analytics-vidhya/non-max-suppression-nms-6623e6572536 xiii, 24

[12] R.Smith, ”An Overview of the Tesseract OCR Engine”,Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) xiii, 24

[13] GitHub, ”Text-Recognition-EAST-detection-and-Tesseract”
https://github.com/pranavstha11/Text-Recognition-EAST-detection-and-Tesseract- xiii, xiv, 26, 27, 32, 33, 37, 38

[14] Y.Baek, B.Lee, D.Han, S.Yun y H.Lee, ”Character Region Awareness for Text Detection”, Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, pp. 9357–9366, 2019. xiii, 21, 28, 29

[15] B-Shi, X.Bai y C.Yao, ”An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition”, IEEE Transactions on Pattern Analysis and Machine Intelligence PP(99) 2015 xiii, 30

[16] Enaire, ”Aip españa”
https://aip.enaire.es/AIP/ xiv, 69, 71, 73

[17] Sparkfun, ”Raspberry Pi Camera Module V2”
https://www.sparkfun.com/products/14028 xiv, 58

[18] baeldung, ”String Similarity Metrics – Edit Distance”
https://www.baeldung.com/cs/string-similarity-edit-distance 61

[19] ABC, ”El avión de pasajeros más grande del mundo aparca en el aeropuerto de Teruel”
https://www.abc.es/viajar/noticias/avion-pasajeros-mas-grande-mundo-aparcara-aeropuerto-teruel xiv, 68

[20] Heraldo de Aragón, ”El aeropuerto de Teruel se consolida como el mayor aparcamiento de aviones de Europa”
https://www.heraldo.es/noticias/aragon/teruel/2020/12/26/el-aeropuerto-de-teruel 67

[21] Aeropuerto Andorra la seu
https://aeroportandorralaseu.cat xv, 70

[22] Aeropuerto de Castelló
https://www.aeroportcastello.com xv, 72

[23] F.Tarrés, P.Gerd y R. Quijada, ”Sistema de reconocimiento de productos” y ”Detección y reconocimiento de texto”. *Virtualmarket*, Tecnocim. (Barcelona. 2020): 28–124. xiii, 13

[24] StackOverflow
https://stackoverflow.com/

[25] Aeròdrom Sallent Pla de Bages https://aerodrom-barcelona-bages.com xiv, 57

[26] A.Krizhevsky, I.Sutskever y G.Hinton, ”ImageNet Classification with Deep Convolutional Neural Network”, Neural Inf. Process. Syst. Found. 2012 Conf., pp. 1–9, 2012. 11

[27] M.Zeiler y R.Fergus, ”Visualizing and Understanding Convolutional Networks”, Comput. Vision–ECCV 2014, vol. 8689, pp. 818–833, 2014.. 12

[28] K. Simonyan y A. Zisserman, ”Very Deep Convolutional Networks for Large-Scale Image Recognition”, Int. Conf. Learn. Represent., pp. 1–14, 2015. 12

[29] C. Szegedy, W. Liu, Y. Jia, y P. Sermanet, ”Going deeper with convolutions”, arXiv Prepr. arXiv 1409.4842, pp. 1–9, 2014. 12

[30] K. He, X. Zhang, S. Ren, y J. Sun, ”Deep Residual Learning for Image Recognition”, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 12

[31] Y. Jiang et al., ”R2CNN: Rotational Region CNN for Orientation Robust Scene Text Detection”, vol. 1, no. c, pp. 1–8, 2017. 21

[32] B. Shi, X. Bai, y S. Belongie, ”Detecting oriented text in natural images by linking segments”, Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 3482–3490, 2017. 21

[33] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, y C. Yao, ”TextSnake: A Flexible Representation for Detecting Text of Arbitrary Shapes”, Lect. Notes Comput. Sci. (including Software Tècnic Tecnocim, SL Todos los derechos reservados Pág. 154 de 160 Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 11206 LNCS, pp. 19–35, 2018. 21

[34] X. Rong, C. Yi, y Y. Tian, ”Unambiguous text localization and retrieval for cluttered scenes”, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 21

[35] A.Graves,S.Fernández1,F.Gomez y J.Schmidhuber, ”Connectionist Temporal Classification: Labelling UnsegmentedSequence Data with Recurrent Neural Networks”, Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006. 21

[36] Y.Gao, Z.Huang, Y.Dai, C.Xu, K.Chen, J.Guo, ”DSAN: Double Supervised Network with Attention Mechanism for Scene Text Recognition”, 2019 IEEE Visual Communications and Image Processing (VCIP). 21

[37] K.Schwenk y F.Huber ”Connected Component Labeling Algorithm for very complex and high resolution images on an FPGA platform”, SPIE Remote Sensing. International Society for Optics and Photonics, 2015. 29

[38] GitHub, ”pi-motion-lite”
https://github.com/pageauc/pi-motion-lite 55