UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería Informática

# Estudio sobre la mejora en la detección de objetos pequeños mediante técnicas de procesamiento de imágenes y super-resolución.

# Study about the improvement of small objects detection using image processing and super-resolution techniques.


Realizado por

Daniel Castillo Conesa

Tutorizado por

Enrique Domínguez Merino
Iván García Aguilar

Departamento

Lenguajes y Ciencias de la Computación


MÁLAGA, Diciembre de 2022

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

# Estudio sobre la mejora en la detección de objetos pequeños mediante técnicas de procesamiento de imágenes y super-resolución

# Study about the improvement of small objects detection using image processing and super-resolution techniques.


Realizado por
**Daniel Castillo Conesa**

Tutorizado por
**Enrique Domínguez Merino**
**Iván García Aguilar**

Departamento
**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA



MÁLAGA, DICIEMBRE DE 2022

Fecha defensa: Diciembre de 2022

# Resumen

Los drones son un tipo de vehículo aéreo no tripulado (UAV) los cuales en los últimos años se han convertido en una herramienta indispensable en multitud de tareas debido a su enorme utilidad, bajo coste y facilidad de uso. La popularización de estos aparatos ha traído consigo un aumento en la toma de imágenes aéreas. En este tipo de imágenes normalmente aparecen objetos de pequeño tamaño. Detectar a los mismos suele ser un paso fundamental para extraer información de acuerdo con el ámbito de aplicación. Existen múltiples modelos entrenados mediante algoritmos de aprendizaje profundo o *Deep Learning* que son capaces de detectar de manera automática objetos en una imagen. No obstante, estos modelos han sido entrenados para identificar objetos de tamaño grande o mediano principalmente. Por tanto, su tasa de detección o *accuracy* desciende conforme los objetos en la imagen se hacen más pequeños, especialmente en imágenes de baja calidad o con un fondo complejo. Las soluciones disponibles para la detección de este tipo de elementos a día de hoy son limitadas. Por ello, es necesario la investigación y aporte de nuevas soluciones dentro de este campo.

En este trabajo se plantea un estudio empírico sobre la mejora en la detección de objetos pequeños mediante la aplicación de redes neuronales convolucionales (CNN), en conjunto con distintas técnicas de procesamiento de la imagen y super-resolución durante la fase de inferencia. Este tipo de técnicas son conocidas como *Test Time Augmentation* (TTA) y tienen como objetivo presentar al modelo de detección la misma imagen de entrada con una serie de transformaciones para así intentar aumentar la tasa de detección de los objetos en el fotograma sobre el cual se desea inferir, y con ello, se logre aumentar el *accuracy* del modelo sin necesidad de re-entrenar el mismo, ya que este es un proceso costoso computacionalmente, quedando fuera de los límites de este trabajo.

**Palabras clave:** Aprendizaje Profundo, Redes Neuronales Convolucionales, Visión por Computador, Detección Objetos Pequeños, Super-resolución, Aumento en Fase de Pruebas.

# Abstract

Drones, also referred to as Unmanned Aerial Vehicles (UAVs), are increasingly being used in many tasks as they become cheaper, easier to use, and more efficient. This growing popularity brings an increment in aerial imagery from these devices. Such images often contain small objects and detecting them can be a necessary step to extract useful information. There are multiple detection models pre-trained with Deep Learning algorithms that can automatically detect objects in an image. However, these models have been mostly trained to detect large or medium-sized objects, and thus their performance or accuracy drops lower as the object size in the image becomes smaller, especially in low-resolution images or images with a complex background. The solutions available for detecting this type of element are currently limited. This is why research and new solutions are needed in this field.

In this paper we present an empirical study about the improvement of small object detection by applying, during the inference step, different techniques of image processing and super-resolution, also known as Test Time Augmentation (TTA), with the goal of improving the detection rate of small objects as we present them to the detection model in different ways, and therefore achieving a better accuracy without having to retrain the model, since this is a computationally expensive process, which is out of scope in this work.

**Keywords:** Deep Learning, Convolutional Neural Networks, Computer Vision, Small object detection, Super-resolution, Test Time Augmentation.

# Índice

# Siglas

AP . . . . . Average Precision

BN . . . . . Batch Normalization

BSRGAN . . . . . Blind Super-Resolution Generative Adversarial Network

CLAHE . . . . . Contrast Limited Adaptive Histogram Equalization

CNN . . . . . Convolutional Neural Network

DOTA . . . . . Dataset of Object detection in Aerial images

EDSR . . . . . Enhanced Deep Super-Resolution network

EEGAN . . . . . Edge-Enhanced Generative Adversarial Network

ESRGAN . . . . . Enhanced Super-Resolution Generative Adversarial Network

Faster R-CNN . . . . . Faster region-based convolutional neural network

FeMaSR . . . . . Feature Matching Super-Resolution

FSRCNN . . . . . Fast Super-Resolution Convolutional Neural Network

GAN . . . . . Generative Adversarial Network

HR . . . . . High-resolution

IoU . . . . . Intersection over Union

JPEG . . . . . Joint Photographic Experts Group

LapSRN . . . . . Laplacian Pyramid Super-Resolution Network

LR . . . . . Low-Resolution

LUT . . . . . Look-Up Table

mAP . . . . . mean Average Precision

MS COCO . . . . . MicroSoft Common Objects in Context

NMS . . . . . Non-Maximum Suppression

PSNR . . . . . Peak Signal-To-Noise Ratio

R-CNN . . . . . Regions with CNN features

ResNet . . . . . Residual Networks

SISR . . . . . Single Image Super-Resolution

SODA . . . . . Small Object Detection dAtasets

SR . . . . . Super-Resolution

SRCNN . . . . . Super-Resolution Convolutional Neural Network

SRGAN . . . . . Super-Resolution Generative Adversarial Network

SRResNet . . . . . Super-Resolution Residual Network

SSD . . . . . Single-Shot Detector

SSIM . . . . . Structural Similarity Index measure

TTA . . . . . Test Time Augmentation

UAV . . . . . Unmanned Aerial Vehicle

VAID . . . . . Vehicle Aerial Imaging from Drone

YOLO . . . . . You Only Look Once

# 1
# Introducción

## 1.1. Motivación

La detección de objetos es una de las aplicaciones más importantes dentro de la visión por computador. Esta tarea consiste en localizar objetos en una imagen así como clasificarlos según la clase a la que pertenecen. La localización de los objetos en la imagen se obtiene principalmente de tres maneras, bien a partir de un recinto delimitador (*bounding box*), a través de un conjunto de puntos de interés (*keypoints*) o mediante la segmentación de su máscara (*segmentation*). La clase a la que pertenecen los objetos detectados se obtiene mediante una etiqueta (*label*) asociada a una clase determinada. El dataset MS COCO [1] define un total de 80 categorías de objetos o clases que han sido identificadas y etiquetadas. Este *dataset* corresponde con uno de los más utilizados dentro del campo por la gran variedad de objetos etiquetados que dispone, así como la variabilidad de escenas y contextos. Sin embargo, este tipo de conjunto de datos presentan un problema relacionado con el tamaño de los objetos que albergan las imágenes. Modelos pre-entrenados haciendo uso de estos tendrán un desempeño aceptable en multitud de escenas, sin embargo, el mismo se verá afectado considerablemente cuando el tamaño de estos elementos vaya disminuyendo.

Un objeto puede ser clasificado de diversas formas atendiendo al número de píxeles al cuadrado que conforman al mismo. En la tabla 1 se definen los tamaños establecidos para cada grupo.

| Tamaño | Área (en píxeles al cuadrado) |
|---|---|
| Pequeño | área <32x32 |
| Mediano | 32x32 <área <96x96 |
| Grande | 96x96 <área |

Tabla 1: Tamaño de los objetos de acuerdo al área que lo conforman en píxeles al cuadrado.

Esta definición está pensada para imágenes de un tamaño o resolución común dentro de los datasets de detección (640x480 px). Pero si nos referimos a imágenes con mayor resolución,

de formato HD (1280x720 px) o superior, esta definición no es del todo precisa, considerando por ello un objeto pequeño como aquel que ocupa menos de un 1 % de la superficie total de la imagen.

La detección de objetos pequeños en imágenes digitales tiene múltiples aplicaciones en una gran variedad de ámbitos, como son por ejemplo el análisis de imágenes satelitales, que sirve para la monitorización del clima, los ecosistemas y las catástrofes medioambientales. El análisis de imágenes médicas se utiliza en muchas áreas de la medicina, y sirve tanto para acelerar el proceso de diagnóstico de enfermedades como para el desarrollo de nuevos medicamentos. Por otro lado, la detección de objetos pequeños desde drones o aviones de reconocimiento es ya una herramienta fundamental dentro de la guerra y el espionaje moderno. Otra área de aplicación corresponde con el cultivo de forma autónoma con el fin de determinar el estado de los mismos, así como el de los animales en la agricultura y la ganadería. Otros ejemplos de su uso son la detección de peatones y señalización en vehículos autónomos, la monitorización del tráfico, el control del urbanismo, la detección y el conteo de personas en grandes aglomeraciones y eventos de todo tipo.

De entre todas estas tareas, este trabajo se centra principalmente en la detección de vehículos circulando por diversas vías. Estas imágenes han sido tomadas desde drones en entornos urbanos. La detección de objetos en este tipo de imágenes presenta dificultades añadidas como son las variaciones en la altitud del aparato, el ángulo de visión de la cámara que incide sobre el terreno, las diferentes condiciones climatológicas y de luminosidad. A esto se le suma el hecho de que a día de hoy aún no se dispone de un dataset de referencia para objetos pequeños al nivel de *MS COCO* [1] o *ImageNet* [2], dificultando por ello el desarrollo de nuevos algoritmos específicos de *Deep Learning* para este propósito, pues el entrenamiento de estas redes neuronales profundas depende directamente de los objetos que componen el dataset. Existen, no obstante, algunos datasets [[3]-[11]] que permiten abordar el problema de la detección de objetos pequeños, pero cuentan en general con muchas menos imágenes y además no suelen guardar relación entre ellos, dificultando por ello la posible reutilización de un nuevo modelo entre los distintos datasets.

## 1.2. Objetivos

Los mejores modelos de detección actuales obtienen muy buenos resultados a la hora de detectar en una imagen objetos de tamaño mediano y grande, pero su eficacia se reduce bastante cuando tienen que detectar objetos pequeños. El TFG consistirá en el desarrollo de una metdología que permita mejorar la detección de un modelo previamente re-entrenado, sin necesidad de modificar la arquitectura del mismo o realizar *Fine-Tuning*. La precisión de modelos pre-entrenados decae en más de 50 puntos porcentuales al pasar de detectar objetos grandes a pequeños. Las redes neuronales convolucionales de aprendizaje profundo cuentan con muchas capas de profundidad, una profundidad que es necesaria para extraer las características y patrones más complejos dentro la imagen, unas características que se obtienen en las capas más profundas de la red. Estos modelos están a su vez diseñados de tal manera que conforme avanzamos por las distintas capas de convolución y *pooling* la entrada o *tensor* se va haciendo cada vez más pequeña. Esto por una parte hace que se mantengan las características principales de la imagen y a su vez simplifica enormemente el coste computacional del entrenamiento pero con la contrapartida de que los detalles y los objetos de menor tamaño se van diluyendo y, o bien no logran mantener suficiente información para su correcta detección o bien directamente desaparecen antes de llegar al final de la red donde finalmente se produce la detección de los objetos. Este tipo de arquitectura perjudica por tanto a los objetos más pequeños, objetos que se componen de menos píxeles y por tanto que contienen menos información, siendo esta necesaria para la extracción de características o *features* que posibilitan tanto su detección en la imagen como su corrrecta clasificación.

En imágenes aéreas los objetos pueden llegar a tener un tamaño muy pequeño y a la vez haber un gran número de estos repartidos por diversas localizaciones de la imagen. Esto complica aún más su detección ya que a los modelos actuales les resulta más difícil diferenciarlos del fondo del fotograma o de otros objetos de distinto tamaño y forma similar [12]. Adicionalmente, aparece el problema del solapamiento de los objetos si hay una gran cantidad de ellos en un mismo espacio, algo que ocurre en el dataset de *VisDrone* [3], el cual ha sido el seleccionado para realizar las pruebas de la metodología propuesta en dicho trabajo. Los modelos estudiados han sido pre-entrenados con *MS COCO* [1], un dataset donde suele haber pocos objetos por imagen en comparación a otros datasets de imágenes aéreas como VisDrone [3], que llega a

albergar un máximo de 902 objetos en una misma imagen, o DOTA [4], que llega a contar con más de 1000 instancias de objetos en algunas imágenes. Finalmente, otro factor a tener en cuenta es el cambio de escala de los objetos en la imagen. *Jingkai Zhou et al.* [13] apuntan en su trabajo que la varianza de la clase Coche en VisDrone [3] es cinco veces superior a la encontrada en MS COCO [1], esto se debe a las distintas variaciones en la altitud de los drones durante la toma de imágenes desde el aire.

A continuación, se enuncian los requerimentos necesarios para cubrir los objetivos planteados en dicho trabajo:

- Elección del conjunto de datos adecuados al campo de aplicación relacionado con la detección de vehículos.
- Estudio y selección de las redes neuronales convolucionales pre-entrenadas existentes a partir de su precisión.
- Evaluación de diversos modelos de super-resolución y selección del más apropiado de acuerdo a la calidad de imagen resultante.
- Metodología basada en la re-inferencia de imágenes super-resueltas con el objetivo de aumentar la precisión en elementos pequeños, así como obtener detecciones mucho más confiables del resto de elementos presentes en la escena.
- Conjunto de pruebas que avalen la mejora tanto cuantitativa como cualitativa obtenida tras aplicar la metodología propuesta.

La metodología propuesta en dicho trabajo de fin de grado se define como precedente para la mejora en la detección de vehículos de tamaño reducido, con la posible implicación que pueda tener en aplicaciones tales como el control de densidades enunciados anteriormente entre otros.

## 1.3. Tecnologías usadas

A lo largo del proyecto, se han utilizado diversas tecnologías, las cuales se enuncian a continuación:

- Python: Lenguaje que facilita el ámbito del *Deep Learning* así como técnicas de preprocesado de la imagen, debido principalmente a la gran cantidad de librerías disponibles.
- Tensorflow 2: Un *Framework* que permite hacer uso de diversos modelos pre-entrenados.
- Google Colab Pro+: Entorno sobre el cual se ha desarrollado la metodología presentada en este trabajo, así como las pruebas pertinentes para avalar el correcto funcionamiento de la misma. Permite la ejecución de código desde el navegador, pues está alojado en la nube. Ádemás, la versión pro facilita la realización de tareas costosas computacionalmente pues disponen de tarjetas gráficas (*GPU*) destinadas para este tipo de ámbitos.
- GitHub: Repositorio en el cual se almacenará el código de la metodología propuesta.
- Latex: Sistema de composición de textos.
- Overleaf: Editor de textos basado en *LaTeX* utilizado para la elaboración de dicha memoria.

## 1.4. Estructura del documento

La memoria está estructurada en una serie de capítulos, los cualrs recogen los pasos llevados a cabo a lo largo del proyecto:

**CAPÍTULO 2 - TRABAJOS PREVIOS:** En este capítulo se describe de forma detallada la fase del estado del arte, incluyendo por ello las tecnologías y trabajos destacables propuesto por otros autores en el ámbito de la detección de objetos.

**CAPÍTULO 3 - METODOLOGÍA DE TRABAJO:** A lo largo de dicho capítulo se indica qué metodología se ha seguido para el diseño e implementación de este proyecto software.

**CAPÍTULO 4 - DATASETS:** En este capítulo se detalla una serie de *datasets* destacables en el ámbito de aplicación de objetos. Posteriormente se enuncia cual de ellos ha sido seleccionado para la realización de las pruebas pertienentes.

**CAPÍTULO 5 - ESTUDIO Y SELECCIÓN DE MODELOS:** En este capítulo se analizan los diversos modelos existentes, tanto dentro del campo de la detección de objetos a partir de

redes neuronales convolucionales, así como para la aplicación de super-resolución. Por ello, se realiza un estudio de acuerdo a la precisión y calidad de imagen respectivamente con el objetivo de seleccionar el más adecuado para la metodología desarrollada.

**CAPÍTULO 6 - MÉTODO PROPUESTO:** En este capítulo se detalla el desarrollo del proyecto, describiendo cada una de las fases que compone dicha metodología.

**CAPÍTULO 7 - RESULTADOS:** En este capítulo, se analizarán los resultados obtenidos utilizando la implementación detallada en el capítulo anterior, aportando por ello una serie de pruebas tanto cuantitativas como cualitativas.

**CAPÍTULO 8 - CONCLUSIONES Y LÍNEAS FUTURAS:** Por último, se exponen una serie de conclusiones tras la finalización de dicho proyecto. Además se definen las líneas futuras sobre las cuales proseguir con dicho trabajo.

# 2

# Trabajos Previos

Una de las premisas de la metodología desarrollada está basada en la ampliación de la imagen de entrada, con el fin de aumentar también el tamaño de todos los objetos contenidos en la misma, incluídos los objetos pequeños. Este proceso se realiza antes de pasar la imagen por el modelo de detección de objetos. El objetivo es que los objetos pequeños aumenten su tamaño y su área en píxeles para con ello aumentar sus posibilidades de ser detectados. El método empleado principalmente para ampliar la imagen de entrada es la Super-Resolución (SR), que no es más que otro tipo de algoritmo basado en redes neuronales convolucionales, pero que devuelve en este caso una imagen ampliada o super-resuelta como salida.

Un ejemplo de aplicación de un modelo de SR para mejorar la detección lo encontramos en [14], donde se propone una red *SRCNN* para mejorar la detección en imágenes remotas multi-espectrales. Otro caso de aplicación de un modelo de *SR* lo encontramos en [15], donde se propone hacer *fine-tuning* a una red conformada por la unión de dos modelos distintos ya pre-entrenados, uno de super-resolución y otro de detección, obteniendo con ello un incremento en la tasa de detección respecto al modelo de detección por separado sobre un conjunto de imágenes de baja resolución.

*Jiang et al.* [16] advierten en su trabajo de que los modelos tradicionales de super-resolución fallan al generar imágenes ampliadas con detalles en los bordes de imágenes contaminadas por ruido, dificultando por ello la tarea de la detección de objetos en imágenes satelitales. Para intentar resolver este problema, proponen una nueva arquitectura de *SR* basada en redes *GAN*, llamada *EEGAN*, la cual sea invariante al ruido. Para ello, emplean dos sub-redes en la parte generadora *G*, una se encarga de generar la imagen de alta resolución (*HR*) y la otra se encarga de generar bordes bien definidos y sin ruido en la imagen final super-resuelta. Otro ejemplo de utilización de super-resolución basada en redes *GAN* se establece en [13], donde se propone un nuevo método de aumento multi-escala conocido como *Scale Adaptative Image Cropping* (*SAIC*), que consiste en recortar la imagen de entrada de distintas maneras para

aplicar super-resolución a distintas escalas según el tamaño estimado del objeto con el objetivo de mejorar las detecciones en imágenes aéreas de baja resolución obtenidas desde UAVs. Más recientemente, *Iván García Aguilar et al.* [17], en su trabajo para mejorar la detección de vehículos en secuencias de imágenes de carreteras, aplica super-resolución a una escala x2 alrededor de objetos ya detectados previamente con el objetivo de que aparezcan nuevas detecciones de objetos pequeños en una segunda pasada del modelo de detección. La premisa de su desarrollo se basa en utilizar las deteciones definidas por el modelo inicialmente, con el fin de utilizarla como zonas tentativas sobre las cuales re-inferir. Con ello, consigue unos buenos resultados, incrementando la precisión con respecto al modelo de detección original sin necesidad de re-entrenarlo o modificar el mismo.

Un caso distinto de re-entrenamiento sobre un modelo ya existente lo tenemos en [18], donde se estudia mejorar la detección de objetos pequeños aumentando artificialmente la presencia de objetos en distintas partes de la imagen. Para ello, se realiza un *oversampling* o sobremuestreo durante la fase de re-entrenamiento del modelo de detección, una técnica conocida como *data augmentation*. En su trabajo, observan que los modelos entrenados con el dataset *MS COCO* [1] están sesgados en el sentido que prestan menos atención a los objetos pequeños en la imagen, ya que la mayor parte de las imágenes con las que han sido entrenados contienen una mayor proporción de objetos grandes y medianos que de pequeños. Por tanto aplican este *data augmentation* sobre las imágenes del dataset que contengan elementos pequeños, sobremuestreando el número de imágenes que los contienen y multiplicando sus apariciones dentro de la misma imagen, con el objetivo de que los modelos de detección presten más atención a estos objetos pequeños durante la fase de entrenamiento.

*Tayara et al.* [19] proponen en su trabajo un nuevo modelo llamado *Fully Convolutional Regression Network* (*FCRN*). Constituye una red neuronal convolucional con la se que aplica regresión espacial sobre mapas de densidad de probabilidad para la detección y conteo de vehículos en imágenes aéreas de alta resolución. Se trata de un nuevo modelo enfocado a este tipo de problema, pero que parte de un modelo base o *backbone* (*VGG-16*) entrenado con el *dataset ImageNet* [2].

Otro tipo de enfoque se centra en realizar un aumento durante la fase de inferencia, algo que

se conoce como *test time augmentation* (*TTA*). Un ejemplo de ello es [20], donde se emplean diferentes técnicas como rotación y volteo de la imagen en imágenes médicas de tejidos para la segmentación precisa de los núcleos dentro de las células en la imagen. Unas técnicas que tienen bastante sentido aplicar ya que las células se encuentran flotando en el líquido y por tanto pueden aparecer de distintas formas en la imagen. Además, el etiquetado en este tipo de imágenes médicas es un proceso bastante costoso y exhaustivo, por lo que aplicar en este caso *TTA* se ofrece como una excelente alternativa para mejorar su tasa de detección.

En [21], van más allá y proponen utilizar ambas técnicas de aumento a la vez, *data augmentation* durante la fase de re-entrenamiento de un modelo basado en *CNN*, y *test time augmentation* durante la fase de inferencia. Su trabajo trata sobre la identificación de distintos tipos de insectos en plantas para el control de plagas, para ello dividen la imagen de entrada en una malla o *grid* de sub-imágenes que luego amplían a distintas escalas así como aplican rotaciones de distinto ángulo a cada una de ellas. Esto permite presentar los insectos en las plantas de múltiples formas y tamaños al modelo de detección consiguiendo mejorar el *accuracy* respecto al modelo original.

Cuando hablamos de detectar objetos pequeños, dadas las problemáticas mencionadas anteriormente, se tiende a sacrificar la velocidad del modelo por un mayor *accuracy*. No obstante, para ciertas aplicaciones es indispensable que el algoritmo se pueda ejecutar en tiempo real, y en este sentido el modelo de detección *YOLO* [22] es la opción más utilizada actualmente. Las primeras versiones de YOLO tenían dificultades para detectar los objetos pequeños en la imagen, pero con las versiones siguientes han ido mejorando en este aspecto. La versión más reciente, *YOLO v7* [23] es considerada el nuevo estado del arte en la detección de objetos en tiempo real tanto a nivel de rendimiento y *accuracy* en la detección, como de velocidad del modelo en la fase de inferencia.

Inspirados por los excelentes resultados de las redes *EEGAN* [16] y *ESRGAN* [24] aplicadas a la tarea de la super-resolución, *Jakaria Rabbi et al.* [25], quienes estudian en su trabajo la detección de objetos pequeños en imágenes satelitales, proponen una nueva arquitectura, llamada *EESRGAN* (*edge-enhanced SRGAN*), la cual funciona tanto para detectores de una sola fase como para detectores en dos fases. El objetivo de aplicar este modelos de *SR* es obtener

imágenes de una mayor calidad, con unos bordes claros y bien definidos, que no se vean afectados por el ruido, consiguiendo así una mejor tasa de detección de los objetos en la imagen, independientemente del tamaño.

Recientemente se ha publicado un trabajo [26] que propone un nuevo *framework*, denominado *Focus-And-Detect*, para la detección de objetos pequeños en imágenes aéreas grabadas desde drones. Este framework realiza dos pasadas sobre la imagen de entrada, en una primera pasada de encarga de proponer las regiones, *focal regions*, de la imagen donde es probable que haya más objetos y en una segunda pasada se realiza la detección de los objetos en las mismas. Finalmente aplican un mecanismo llamado *Incomplete Box Suppression* (*IBS*) para unificar las *bounding boxes* de los objetos detectados en regiones focales que se solapan entre sí. Con su propuesta alcanzan un 42.06 de *mAP* sobre el dataset de validación de *VisDrone* [3] sin aplicar ninguna técnica de *TTA* sobre la imagen.

# 3

# Metodología de Trabajo

El desarrollo de este proyecto se ha dividido en una serie de fases o hitos y se ha fundamentado en la metodología ágil de tipo *Scrum*, elegida con el objetivo de mejorar la adaptación a los cambios imprevistos durante el desarrollo incremental del proyecto. Se ha elegido esta metodología porque, al ser un trabajo basado en un estudio sobre cómo mejorar la detección de objetos pequeños en imágenes, en un primer momento no estaban correctamente definidos todos los requisitos al no conocer con total seguridad las herramientas con las que se iban a trabajar. Por ello, dicha metodología ha facilitado la compaginación de las fases que conforman el estudio del *background*, los trabajos previos y los modelos, así como la detección y super-resolución para el desarrollo del código.

Cada fase del proyecto se ha dividido a su vez en varias tareas o hitos más pequeños que han permitido trabajar conjuntamente, de forma ágil y eficiente, en los objetivos propuestos durante cada fase del proyecto, así como facilitar y mejorar la organización de éste, simplificando a su vez su desarrollo.

Para llevar un control más activo del proyecto, detectar de forma temprana errores y evitar dejarlos para fases más tardías del proyecto, se ha trabajado mediante sprints y reuniones semanales (*weekly meetings*) donde se ha comentado el trabajo realizado durante cada sprint, los problemas encontrados, soluciones para tratar estos problemas y los objetivos de cara al siguiente *sprint*. Las comprobaciones continuas y las revisiones que se han ido realizando durante cada fase han reducido el número de problemas y fallos encontrados conforme se ha ido avanzando en el desarrollo del proyecto.

# 4

# Datasets

Antes de comenzar a abordar el estudio y selección de modelos para llevar a cabo la metodología descrita, es conviente mencionar brevemente algunos de los datasets que se han explorado durante la realización de este trabajo.

*MS COCO* [1] es uno de los datasets de referencia para abordar tanto el problema de la detección de objetos como de la segmentación de la imagen. La mayor parte de los modelos de detección de objetos que se estudian en este trabajo han sido entrenados con este dataset. Actualmente cuenta con 300.000 imágenes de ámbito general registradas, de las cuales más de 200.000 están etiquetadas, con un total de un millón y medio de instancias de objetos catalogadas entre todas ellas. Por otra parte, cuenta con 80 categorías o clases de objetos. Para permitir el entrenamiento supervisado de los modelos de detección, han definido un estándar de etiquetado que permite asociar fácilmente cada objeto etiquetado en una imagen, conocido como *ground-truth*, dentro de un conjunto de anotaciones que luego pueden evaluarse de diversas maneras con respecto a las detecciones, siguiendo el mismo formato de las mismas.

*ImageNet* [2] es el otro dataset de referencia junto con *MS COCO* y el más grande por volumen de datos. A día de hoy, cuenta con 14.197.122 imágenes disponibles, de las cuales algo más de un millón están etiquetadas con *bounding boxes*. A su vez, cuenta con más de 20.000 categorías distintas de objetos. Suelen aparecer subconjuntos de este dataset en los *benchmarks* de prueba a la hora de evaluar modelos de detección, sean o no entrenados con este *dataset* y lleva celebrando anualmente desde 2010 un evento conocido como *ImageNet Large Scale Visual Recognition Challenge* (*ILSVRC*), donde compiten en distintos *benchmarks* los modelos de detección de objetos.

*VisDrone* [3] es un dataset creado por el equipo *AISKYEYE* y que contiene 288 secuencias de video con 261.908 frames y 10.209 imágenes estáticas grabadas desde drones en entornos urbanos. Dispone de un total de más de 2.6 millones de anotaciones y cuenta con 11 clases distintas,

como personas y varios tipos de vehículos. Es el dataset que se ha elegido para realizar el presente trabajo, pues permite abordar el problema de la detección de objetos pequeños, y a su vez las categorías que aparecen en sus imágenes pueden ser identificadas por los distintos modelos entrenados con *MS COCO* los cuales están disponibles en *Tensorflow Model Zoo* [0]. No obstante, es necesario aplicar una transformación de la *bounding box* de las anotaciones de *VisDrone* para poder evaluar los modelos de detección contra este *dataset*, ya que éstas no siguen el estándar de *COCO*. Todas las pruebas realizadas en este trabajo se han obtenido utilizando el dataset de detecciones de prueba de 2019, conocido como *VisDrone_2019-DET-test-dev*, que cuenta con un total de 1.610 imágenes. *VAID* [4] es otro dataset disponible para la detección de vehículos a partir de imágenes grabadas desde un dron. Consta de 6.000 imágenes aéreas de tamaño 1137x640 con distintos ángulos de incidencia sobre la carretera y en distintas condiciones lumínicas. Por su parte cuenta con siete categorías distintas de vehículos.

Figura 1: Ejemplo de fotogramas que componen el *dataset* de **Visdrone**.

*SODA* [5] es un dataset publicado recientemente para la detección de objetos pequeños y que se divide a su vez en dos partes o subconjuntos, pues está centrado en dos temáticas distintas. Por un lado está *SODA-D*, que se centra en la temática de conducción de vehículos por carretera, y contiene 24.704 imágenes de alta calidad que muestran escenarios típicos de conducción desde

[0] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

el punto de vista del vehículo y 277.596 instancias anotadas con *bounding boxes* de nueve categorías distintas de objetos comunes, que suelen aparecer en este tipo de escenarios, como son señales de tráfico, semáforos, peatones u otros vehículos. Por su parte, *SODA-A*, cuya temática son las imágenes aéreas, cuenta con 2.510 imágenes de alta resolución, 800.203 instancias de objetos anotadas con *bounding boxes* orientadas y 9 categorías de objetos.

*DOTA* [6] es un dataset para la detección de objetos pequeños en imágenes tomadas por satélite y obtenidas de distintas fuentes públicas. Su versión más reciente, *DOTA-v2.0*, cuenta con 11.268 imágenes y 1.793.658 instancias de objetos etiquetadas, así como un total de 18 clases o categorías de objetos distintas que pueden ser vistas desde un satélite. También dispone de imágenes tanto en color como en escala de grises. Este dataset también tiene la particularidad de que sus *bounding boxes* están rotadas, pues desde arriba a vista de satélite los objetos pueden tener cualquier orientación, y cuenta a su vez con una métrica llamada*Ground Sampling Distance* (*GSD*), que indica la distancia entre píxeles en metros de la Tierra. El problema encontrado con este dataset es que las imágenes que contiene (objetos muy pequeños vistos desde arriba), son demasiado distintas a las encontradas en *MS COCO*, y por tanto no se pueden utilizar de manera efectiva los modelos de detección de la librería *TensorFlow* basados en *COCO* para detectar objetos en imágenes de este dataset. Un trabajo derivado de *DOTA* es *iSAID* [7], un dataset centrado en la segmentación precisa de imágenes aéreas de alta resolución que contienen gran cantidad de objetos muy pequeños. A pesar de que cuenta con únicamente 2.806 imágenes, este dataset llega hasta las 655.451 instancias de objetos anotadas a nivel de píxel y dispone de 15 categorías de objetos distintas. Otro dataset de imágenes satelitales es *AID* [8], pero que en este caso trata el problema de la clasificación de la escena que aparece en la imagen. Entender la escena en una imagen es tarea más abstracta y compleja dentro de la visión por computador, desde el punto de vista semántico. Este dataset contiene más de 10.000 imágenes anotadas con 30 categorías semánticas distintas de escenas que pueden ser vistas desde un satélite.

Otros datasets encontrados que permiten abordar el problema de la detección de objetos pequeños pero con un propósito más específico y alejado de la temática de este trabajo son: *FLAME* [9], un dataset para la detección de incendios forestales, *Bee4Exp Honeybee Detection* [10], un dataset para la detección de abejas en imágenes tomadas desde drones o *NIH Chest X-*

*ray* [11], un dataset de imágenes en rayos X de pacientes para el diagnóstico de enfermedades del tórax.

# 5

# Estudio y selección de modelos

En este apartado, se enuncian los diferentes modelos que se han explorado durante la realización del presente trabajo. Cabe mencionar que, si bien existen más modelos de detección que los mencionados abajo, no todos ellos tienen compatibilidad directa con la librería *TensorFlow* con la que se ha trabajado. De los no incluídos, el más destacado es *YOLO* [22], que finalmente no ha sido incluido porque utiliza un entorno distinto (*Darknet*). Además, este presenta mayores dificultades para detectar objetos pequeños en una imagen.

## 5.1. Modelos de detección de objetos

Los modelos de detección de objetos los podemos dividir, según cómo procesan la imagen de entrada, en dos grupos. Por un lado están los que procesan la imagen en una sola pasada, los llamados *one-stage detectors*, y por otro lado los que lo hacen en dos fases, que se conocen como *two-stage detectors*. Los de este último grupo, hacen primero una pasada sobre toda la imagen para predecir las regiones de la imagen, regiones de interés o *regions of interest* (*RoI*), dónde es más probable que se encuentren los objetos. Luego en una segunda pasada detectan y clasifican los posibles objetos que se encuentren en esas regiones obtenidas en la pasada anterior. Por otra parte, los modelos de detección en una sola fase hacen, como su propio nombre indica, tanto la localización de candidatos como su detección y clasificación en una misma pasada.

Los detectores en dos fases alcanzan mejores resultados en los *benchmarks* de referencia. En general estos modelos tienen una arquitectura más compleja y un mayor número de capas, esto hace que suelan tener un *accuracy* más alto pero a costa de una velocidad menor, ya que proponen potencialmente más candidatos que los detectores en una fase, quienes, por su parte, intentan simplificar la estructura de la red y quitar los elementos menos eficientes computacionalmente. Esto hace que sean más rápidos pero también a costa de ser menos precisos que el otro grupo. De este último grupo sólo se ha incluído a la familia de modelos de *SSD*.

## 5.2. Modelos de detección en dos fases (*Two-stage detectors*)

La familia de modelos de *EfficientDet* [27], desarrollada por *Google Brain* y cuyo *backbone EfficientNet* ha sido pre-entrenado con el dataset *ImageNet* [2], incorpora cambios en la arquitectura de la red focalizados en la eficiencia del modelo como son:

*Bi-directional Feature Pyramid Network* (*BiFPN*), que permite la fusión ponderada de características de forma bidireccional y multiescala entre capas de una manera más eficiente y sencilla que sus predecesores. Estas características luego se envían por igual tanto a la red que se encarga de detectar la clase del objeto (*Class prediction net*) como a la que se encarga de detectar la *bounding box* del mismo (*Box prediction net*).

También proponen en una versión más reciente [28] un método de escalado compuesto de resolución, profundidad y anchura de la red que funciona uniformemente y de manera conjunta, que se ha mostrado como el método más eficiente en comparación con otros métodos de una única escala o de escalado limitado de la red, obteniendo con ello los mejores resultados hasta la fecha en el dataset *ImageNet* y otros benchmarks de referencia.

Los modelos de *CenterNet* [29] se caracterizan por emplear una arquitectura sin cajas, o *anchorless*, que permite la simplificación del modelo y ofrecen un tiempo de procesamiento más rápido que otras alternativas. Estos *anchors* o cajas son utilizados por los detectores para realizar las predicciones de donde es más probable que se encuentre el objeto detectado, y para ello se generan varios de estos por cada objeto. El problema de este enfoque es que es muy ineficiente a nivel computacional, especialmente en imágenes con muchos objetos donde suele haber solapamientos. Además, requiere de un preprocesado de todos los *anchors*, incluidos los que se van a descartar, antes de aplicar *NMS*. Por ello *CenterNet* apuesta por otro enfoque más simple basado en los *keypoints*. Para pasar de *keypoints* a *bounding boxes* utilizan la siguiente fórmula:

$$(x + \delta_x - w/2, y + \delta_y - h/2)$$

$$(x + \delta_x + w/2, y + \delta_y + h/2)$$

Donde $(x, y)$ es el centroide del objeto detectado, $(\delta_x, \delta_y)$ es una estimación del *offset* o desplazamiento del centroide y $(w, h)$ es una estimación de la anchura y altura del objeto detectado. El modelo realiza por tanto dos predicciones para detectar el objeto, por una parte predice el centroide a partir de una distribución de probabilidad a la que se aplica un filtro gaussiano para detectar los picos donde se encuentran los centroides, y por otra parte se encarga de predecir mediante regresión los valores $(\delta_x, \delta_y)$ y $(w, h)$ del centroide correspondiente. Este método por tanto ya no requiere de aplicar *NMS*, pues se eliminan las predicciones irrelevantes al devolver un único *keypoint* por pico máximo en el mapa de probabilidad.

De los modelos de detección en dos fases analizados, *CenterNet* es el que opera más rápido de todos. Dispone de dos variantes que son capaces de operar en tiempo real, una emplea *Resnet18* y funciona a 142 FPS con un 28.1 % de AP, la otra, que emplea *DLA34* como *backbone* alcanza un 37.4 % de AP a 52 FPS. Por otra parte, la variante que emplea *Hourglass-104* como *backbone* opera ya a sólo 1.4 FPS pero en cambio, aumenta el *AP* de *COCO* hasta un 45.1 %. Por ello, aunque *CenterNet* está originalmente pensado para detectar de manera rápida *keypoints* de los objetos, se ha decisido utilizar esta última variante, ya que genera un *bounding box* por cada detección y es la más precisa, para las pruebas. Ya que, cuando hablamos de detectar objetos pequeños en la imagen, es conveniente sacrificar velocidad por *accuracy*.

*CenterNet* también permite la detección de objetos con orientación, la estimación de *bounding boxes* en 3D, así como la estimación de poses humanas. Otros tipos de modelos que también detectan *keypoints* en una imagen son *CornerNet* y *ExtremeNet*, pero no se van a cubrir en este trabajo.

La familia de modelos basados en *R-CNN* [30] se caracterizan por la proposición de regiones previa detección de los objetos en la imagen. El *accuracy* de este tipo de modelos depende del buen funcionamiento del módulo que propone las regiones de interés o *regions of interest* (*RoI*), pues la tarea de proposición de regiones se encarga de decirle a la de detección de objetos dónde hay que mirar para detectarlos, con lo cual los objetos que queden fuera de estas regiones de interés no se llegan a detectar. *R-CNN* utiliza un método de búsqueda selectiva para localizar las RoI, que luego clasifica independientemente en dos grupos, categorías de objetos o fondo de la imagen. *Fast-RCNN* [31] mejora al *R-CNN* original añadiendo mejoras como extraer las

regiones de interés de los mapas de características y consiguiendo una mayor eficiencia y velocidad en la fase de inferencia que su predecesor.

*Faster R-CNN* [32] soluciona el problema de la lentidud en la parte de proposición de regiones introduciendo el concepto de *Region Proposal Networks* (*RPN*), una *fully convolutional network* (*FCN*) que actúa como puente entre el módulo de proposición de regiones y el de detección de objetos y que permite compartir información de las capas convolucionales entre ambos. Esto hace que el tiempo de cómputo necesario para predecir las *RoI* se reduzca drásticamente en comparación los modelos de detección anteriores basados en regiones, al poder realizar todo el cómputo desde la *GPU* dentro de la misma red (*end-to-end*). La *RPN* toma una imagen como entrada y devuelve un conjunto de rectángulos o regiones de la imagen, cada uno con una puntuación o *score* en función a la probabilidad de que esa región contenga objetos de alguna clase o de ser simplemente fondo de la imagen. Previamente ya se había investigado sobre este enfoque, un ejemplo de ello es la red de proposición *MultiBox* [33], que funciona únicamente para secciones de la imagen, a diferencia de *RPN*, que es *fully convolutional*. Además la red *Multibox* no comparte información entre los módulos de proposición y detección como sí hace *RPN*.

Dentro de esta familia de modelos, *Faster R-CNN* es considerado como el estado del arte, y empleando el *backbone VGG-16*, alcanza los 5-7 FPS en una *GPU* y un *mAP* de 42.7 sobre el dataset de prueba de *MS COCO*, en comparación con el 35.9 que obtiene *Fast R-CNN*.

## 5.3. Modelos de detección en una fase (*One-stage detectors*)

*SSD* [33] es un modelo que se compone básicamente de una *DNN* que funciona como base de la red, a la que se le ha quitado la última capa, la de clasificación, y en su lugar se han metido varias capas convolucionales de tamaño decreciente, lo que permite hacer predicciones para detectar objetos a múltiples escalas.

El modelo otorga una puntuación por cada recinto según estime la presencia de un objeto perteneciente a una clase u otra, y luego reajusta con mayor precisión el *bounding box* inicial, llamado *default box*, aplicando un *offset* relativo a cada punto de la caja o recinto que rodea al objeto detectado. Las detecciones finales se producen tras aplicar *NMS* de todas las cajas ge-

neradas. La red también combina predicciones utilizando mapas de características de distinto tamaño o resolución, lo que permite detectar objetos de distinto tamaño en la misma imagen, a diferencia de *YOLO* [22], que sólo opera a una misma escala.

La idea de *SSD* es predecir las puntuaciones de las detecciones así como el tamaño de las cajas o bounding boxes utilizando pequeños filtros convolucionales aplicados a los mapas de características a distintas escalas. *SSD* es más simple que los modelos en dos fases ya que no tiene que proponer regiones al módulo de detección de objetos. Y el hecho de ser un modelo más simple y compacto permite que sea más fácil de integrar y entrenar en sistemas que sólo requieren de un modelo de detección. También es un modelo que tiene un buen comportamiento en imágenes de baja resolución. No obstante, tiene un rendimiento menor que *Faster R-CNN* en cuanto a detección de objetos pequeños. Sin embargo *SSD* es capaz de ejecutarse en tiempo real, pues *SSD500* funciona a 19 FPS con una entrada de 512x512 y *SSD300* alcanza los 59 FPS con imágenes de 300x300. Dependiendo del modelo base empleado y del tamaño de imagen de entrada se puede alcanzar una mayor o menor velocidad de procesamiento sin repercutir demasiado en la precisión del modelo.

Antes de presentar la evaluación de los distintos modelos conviene explicar brevemente la terminología y fórmulas empleadas en dicha evaluación. Para medir lo correcta que es la detección de un objeto en una imagen, calculamos la *ratio* de solapamiento o intersección que existen entre la *bounding box* del objeto detectado y la *bounding box* de la solución previamente etiquetada (*ground-truth*) respecto de la unión de ambas.

$$IoU = \frac{\text{Área de intersección}}{\text{Área de unión}}$$

A esta fórmula se la conoce como *Intersection over Union* o *IoU*, y en la Figura 2 se muestra un ejemplo de su aplicación.

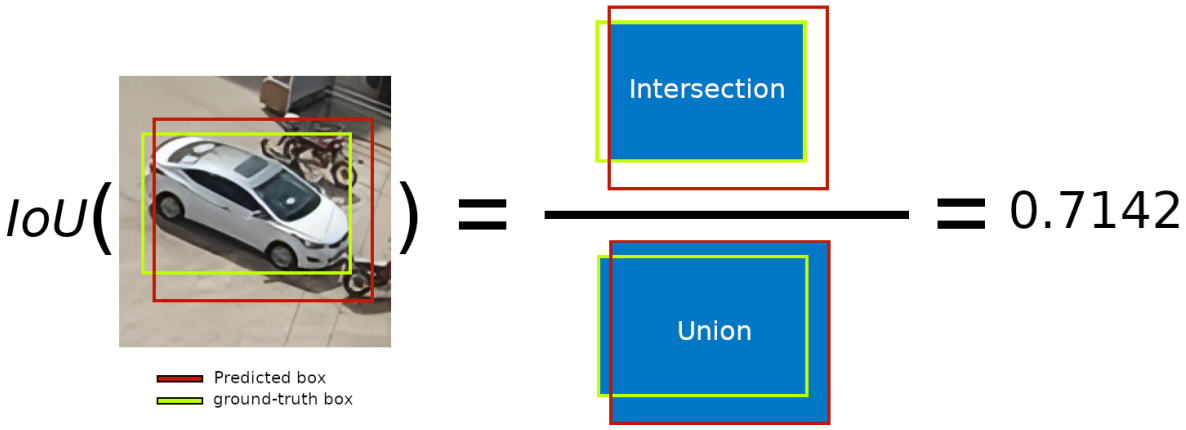


Figura 2: *IoU* de una detección.

Además del *IoU*, existen otros factores que entran en juego para determinar si una detección es válida o no. En este sentido, podemos agrupar todas las detecciones que genera un modelo de detección y su correspondiente solución o *ground-truth* en tres grupos:

1. Verdaderos positivos o *True Positives* (TP).
2. Falsos positivos o *False Positives* (FP).
3. Verdaderos negativos o *True Negatives* (TN).
4. Falsos negativos o *False Negatives* (FN).

Una detección es un verdadero positivo (*TP*) si cumple las siguientes condiciones: Su *IoU* es mayor o igual a un valor umbral preestablecido, al que llamamos $threshold_{IoU}$ , su puntuación o *score*, también conocida como *score objectness*, es mayor o igual a otro valor umbral que podemos denotar como $threshold_{score}$, y si la clase o *label* con la que ha sido etiquetado es correcta. Por otra parte, si la clase es incorrecta o si el *IoU* es inferior al $threshold_{IoU}$, bien porque la *bounding box* no está lo suficientemente solapada con el *ground-truth* o bien porque directamente la ha localizado en un lugar donde no hay nada, entonces tenemos un falso positivo (*FP*). Si en la solución existe un objeto pero el modelo no logra detectarlo, entonces es catalogado como un falso negativo (*FN*). Por último, los verdaderos negativos (*TN*) se dan cuando hay una detección con un *score* inferior al valor umbral y además el *IoU* es inferior

al $threshold_{IoU}$, esto ocurre cuando el modelo detector confunde por ejemplo el fondo de la imagen con un objeto, dándole una puntuación muy baja, cuando realmente no hay nada allí. Como estas detecciones se descartan no intervienen pues en la evaluación de un modelo.

Una vez definidos los *TP*, *TP*, *TN* y *FN*, podemos presentar los términos conocidos como *Precision* y *Recall*. El primero de ellos sirve para medir el porcentaje de las detecciones realizadas que son correctas respecto del total de detecciones realizadas en la imagen $I$. Dicho de otra forma, el número de verdaderos positivos entre la suma de los verdaderos positivos y los falsos positivos en dicha imagen.

$$\text{Precision}(I) = \frac{TP_I}{TP_I + FP_I}$$

El término que mide el porcentaje de detecciones sobre el total de objetos en una imagen, o *ground-truth* de la imagen $I$ (el número de verdaderos positivos dividido por la suma de los verdaderos positivos y los falsos negativos), se conoce como *Recall* (*R*).

$$\text{Recall}(I) = \frac{TP_I}{TP_I + FN_I}$$

Si representamos los dos términos anteriores en una gráfica, obtenemos la llamada curva *Precision-Recall* o *PR curve*. El área bajo dicha curva es lo que se conoce como *Average Precision* o *AP*, cuyo rango está entre 0 y 1.

$$AP = \int_{r=0}^{1} p(r)dr = \frac{1}{11}\sum_{r=0}^{1} p(r)$$

Esta integral en la práctica la podemos aproximar mediante una suma de rectángulos si discretizamos el intervalo de integración [0,1] en 11 partes de igual base con incrementos de 0.1 Por último tenemos el término de *mean Average Precision* o *mAP*, que es el promedio de *AP* entre todas la clases que aparecen en la evaluación. Esta es la métrica empleada por el evaluador de *COCO* y la que se muestra en la Tabla 3.

$$mAP = \frac{1}{C} \sum_{c}^{C} AP_c$$

A continuación, se presentan en la Tabla 3 los resultados de las pruebas realizadas con todos los modelos de detección. Se han seleccionado los tres modelos de cada familia que mejor se adaptan al método propuesto en términos del tamaño de la entrada. Las pruebas se han realizado sobre un subconjunto de 300 imágenes seleccionadas aleatoriamente del dataset de prueba de *VisDrone* y muestran el *mAP* según distintos intervalos de *IoU* y el área del objeto detectado. Estos resultados se han calculado empleando un umbral de detección ($threshold_{score}$) de 0.2.

| Familia | Versión del Modelo | IoU=[0.5:0.95] area = all | IoU > 0.5 area = all | IoU > 0.75 area = all | IoU=[0.5:0.95] area = small | IoU=[0.5:0.95] area = medium | IoU=[0.5:0.95] area = large |
|---|---|---|---|---|---|---|---|
| | EfficientDet3 | 0.159 | 0.260 | 0.170 | 0.039 | 0.284 | 0.536 |
| EfficientDet | EfficientDet4 | 0.204 | 0.322 | *0.221* | 0.072 | 0.340 | **0.628** |
| | EfficientDet5 | **0.246** | **0.397** | **0.267** | **0.123** | **0.376** | *0.572* |
| | CenterNet ResNet50 | 0.047 | 0.110 | 0.034 | 0.012 | 0.083 | 0.218 |
| CenterNet | CenterNet Hourglass104 | 0.129 | 0.243 | 0.123 | 0.062 | 0.200 | 0.392 |
| | CenterNet Hourglass104 | 0.193 | *0.341* | 0.193 | *0.113* | 0.276 | 0.455 |
| | faster_rcnn_resnet101_v1 | 0.076 | 0.155 | 0.067 | 0.025 | 0.129 | 0.230 |
| R-CNN | faster_rcnn_resnet152_v1 | 0.083 | 0.153 | 0.082 | 0.025 | 0.146 | 0.279 |
| | faster_rcnn_resnet_v2 | 0.141 | 0.273 | 0.132 | 0.063 | 0.231 | 0.321 |
| | ssd_resnet50_v1 | 0.123 | 0.211 | 0.129 | 0.051 | 0.206 | 0.289 |
| SSD | ssd_resnet101_v1 | 0.135 | 0.225 | 0.142 | 0.055 | 0.224 | 0.346 |
| | ssd_resnet152_v1 | 0.134 | 0.231 | 0.139 | 0.060 | 0.215 | 0.362 |

Tabla 3: Resultados de las pruebas de evaluación de los modelos de detección. El mejor resultado en cada métrica se muestra en negrita y el segundo mejor en cursiva.

Como vemos arriba, la familia de modelos de *EfficientDet* [27] es la que mejores resultados ha obtenido, y en concreto su versión *D5* es la que alcanza un mayor mAP en todas las columnas, incluida la de objetos pequeños, mientras que el *D4* gana en los objetos grandes. Aunque la idea es seleccionar modelo cuyo *mAP* sea el mayor, se ha optado en su lugar por utilizar la versión *EfficientDet4* ya que el tamaño de la entrada de este modelo (1024x1024 px) favorece en mayor medida la aplicación de nuestra propuesta que el modelo *EfficientDet5* (1280x1280 px). Durante la fase de pruebas se utilizaron ambas versiones por separado con la metodología desarrollada. Los resultados confirman que la versión *D4* obtiene los mejores resultados, motivo por el cual se ha seleccionado la misma para el resto de pruebas.

## 5.4. Modelos de Super-Resolución

La super-resolución o *SR* es una aplicación dentro del campo de procesamiento de imágenes que crea una imagen de alta resolución (*HR*) a partir de una imagen de baja resolución (*LR*). La super-resolución se ha convertido en una de las herramientas principales si no la más importante para mejorar la *ratio* de detección de objetos pequeños en imágenes por las razones que se han comentado anteriormente. Por ello, se realiza un estudio y una comparación de los diferentes modelos de *SR* que hay disponibles actualmente con el objetivo de seleccionar el modelo que mejor funcione y poder así utilizarlo en nuestra metodología con el fin de lograr los mejores resultados posibles.

*EDSR* [34] es un modelo de super-resolución basado en la arquitectura *ResNet* [35] o *Residual Network*. Mejora la misma realizando algunas optimizaciones en los parámetros de la red, así como eliminando las capas de *Batch Normalization* (*BN*), que no son necesarias para este tipo de arquitectura. Con ello, no sólo simplifica el modelo, sino que mejora además su rendimiento durante la fase del entrenamiento.

*ESPCN* [36] es un modelo de super-resolución basado en redes neuronales convolucionales (*SRCNN*) que propone como novedad realizar las tareas de extraer los mapas de características así como aplicar los complejos filtros de escala al principio del proceso, en el espacio de baja resolución (*LR*), en lugar de al final del proceso, en el espacio de alta resolución (*HR*), donde es más costoso computacionalmente. Además introducen un filtro de convolución a nivel de subpíxel más eficiente para escalar los mapas de características de *LR* a *HR*. Esto lo hace bastante más rápido que otros modelos y es apto para realizar *SR* en tiempo real sobre videos a 1080p.

*FSRCNN* [37] es otro modelo basado en *SRCNN* que propone una estructura de *CNN* en forma de reloj de arena con el objetivo de realizar la *SR* más rápido y con mejores resultados que sus predecesores. Esta forma de reloj de arena se debe a que, durante el mapeo de las características, que es la tarea más costosa computacionalmente de un modelo de *SR*, primero se reduce el tamaño de la entrada (*shrinking layer*), con el objetivo de disminuir la complejidad durante la propia fase de *mapping*. Tras esto, se vuelve a agrandar (*expanding layer*) con el objetivo de expandir la dimensión de las características y mantener un buen nivel de detalle en la salida. Esto hace al modelo bastante más rápido sin perjudicar a su *accuracy*. Este modelo es capaz de

ejecutarse en tiempo real en una *CPU* obteniendo buenos resultados, por lo que es la opción más adecuada si no se cuenta con una *GPU*.

*LapSRN* [38] es un modelo inspirado en *LAPGAN* [39] pero aplicado a la tarea de *SR*, para ello utiliza otro tipo de función de coste (*Charbonnier*) y cambia la arquitectura interna. En lugar de agrandar la imagen antes de cada capa de convolución, *LapSRN* opta en su lugar por extraer las características directamente del espacio de baja resolución (*LR space*) y sólo agranda la imagen al final de cada capa. También proponen compartir características de capas anteriores con capas más profundas con el objetivo de aprender patrones más complejos sin aumentar el coste computacional. Esto es posible porque el tamaño entre capas es similar, va creciendo progresivamente, lo que le da a la red una forma de pirámide.

*FeMaSR* [40] es un trabajo reciente el cual se diferencia de otros métodos de *SR* en que emplea el espacio de características de la imagen en lugar de su dominio espacial para generar la imagen super-resuelta. Este modelo mapea las características distorsionadas de la imagen de baja resolución (*LR*) con sus equivalentes de alta resolución (*HR*). Este mapeo se realiza mediante unos *priors* ya pre-entrenados que almacenan los mapas de características que contienen la información codificada así como su decodificador. Estos mapas han sido pre-entrenados mediante una red *GAN* de vectores cuantificados (*VQ-GAN*) e incorpora regularización semántica para mejorar la calidad de la imágen reconstruida. El inconveniente con este enfoque es que se necesitan imágenes *HR* de referencia para el entrenamiento de los *priors*, y por tanto el rendimiento del modelo depende del dominio de entrenamiento de la red *GAN*.

Las redes *GAN* [41] son un tipo de redes neuronales que están formadas por dos modelos distintos, un modelo generador (*G*) y otro discriminador (*D*). El modelo generador se entrena para aprender los patrones que le permitan generar nuevos datos que se parezcan lo más posible a la entrada con la que ha sido entrenado y que por tanto hagan pensar a la parte discriminadora que son datos reales en vez de artificiales. El modelo discriminador se entrena para distinguir los datos reales de los datos generados por *G*. En este sentido, ambos modelos compiten entre sí para ver quien supera al otro en rendimiento.

Desde su primera aparición en 2016 con *SRGAN* [42], los modelos de super-resolución basados en redes *GAN* han ido mejorando progresivamente y mostrado un rendimiento superior a los

modelos tradicionales de *SR* basados en *CNN*, siendo capaces de generar imágenes más realistas y con mayor nivel de detalle y definición en las texturas y formas de los objetos.

*ESRGAN* [24] es un modelo basado en *SRGAN* que trae consigo diversas mejoras sobre los factores claves de una red *GAN*. Estos son la arquitectura, la función adversaria y la función de percepción. Introducen como base de la arquitectura de la red los bloques residuales densos, *Residual-in-Residual Dense Block* (*RRDB*), este nuevo tipo de bloque es más profundo y complejo que los *Residual Block* (*RB*) que utiliza la *SRGAN* original, y tienen como objetivo aprender patrones más complejos. Se eliminan las capas de *BN*, algo que también hace *EDSR* y que hace que el modelo sea más eficiente durante la fase de entrenamiento. En lugar de la función *standard Discriminator* que utiliza la *SRGAN*, se utiliza una nueva función adversaria llamada *Relativistic average Discriminator* (*RaD*) para generar imágenes más realistas. Como función de percepción desarrollan una nueva alternativa más eficiente, que se enfoca más en las texturas de la imagen que en los objetos y obtiene resultados más consistentes en comparación con la imagen real a nivel de brillo medio.

Este modelo, al estar basado en las redes *GAN*, no está orientado tanto a optimizar el *PSNR* de la imagen súper resuelta, sino más bien a obtener una imagen más natural a la vista humana, con un nivel de detalle y textura más realista. Por ello, utilizan otra métrica llamada *perceptual index* o *PI* más asociada a la calidad de la percepción de la imagen reconstruida. El problema con este modelo es que sólo está disponible con un factor de ampliación de x4 en su versión original. No obstante, hay disponible una extensión más reciente llamada *Real-ESRGAN*, que sí permite realizar *SR* a escalas x2, x3 y x4, y es ésta la versión del modelo que se va a emplear durante la fase de estudio de *SR*.

Finalmente, *BSRGAN* [43] es un modelo basado en *ESRGAN* [24] pero cuya principal novedad es que ha sido entrenado con un modelo de degradación no conocido de antemano. Un modelo de degradación es aquel que realiza el proceso inverso de *SR*, esto se basa en, convertir una imagen *HR* a otra *LR*. En su trabajo parten de la premisa de que no se pueden obtener modelos de super-resolución realistas si la imagen de entrada no ha sido previamente degradada de forma efectiva. Por ello, el modelo de degradación juega un papel importante dentro del rendimiento de la *SISR*. Este nuevo enfoque, llamado *blind-SR* y de ahí su nombre, se diferencia de la

mayoría de modelos de *SR* disponibles, los cuales han sido entrenados utilizando básicamente interpolación bicúbica, a lo que se conoce como *non-blind SR*, pues se conoce de antemano el método de degradación. Por ello proponen un nuevo modelo de degradación que consiste en aplicar simultáneamente distintas transformaciones de forma aleatoria para conseguir una mayor variedad en las imágenes generadas que se utilizan para entrenar la red. Se aplican sobre la imagen original métodos de reducción de resolución, aumento de la difuminación y del ruido sobre la imagen de entrada. Para aumentar la borrosidad de la imagen emplean convoluciones con dos tipos de *kernel* gaussianos, uno isotópico y otro anisotrópico. Para los métodos de reducción se utilizan interpolación bilineal, bicúbica y *nearest*. Por último para el ruido se elige entre un ruido gaussiano en distinta intensidad, distintos factores de compresión JPEG y una simulación del ruido que se produce en las cámaras fotográficas.

Se han encontrado y probado varios métodos más de SR que finalmente no se han incluido en el trabajo debido a que o bien solo disponían de un factor de ampliación de x4 o bien no ha sido posible incorporarlos de manera efectiva durante la fase de estudio.

Para analizar el rendimiento de los distintos modelos de super-resolución existen dos métricas, *PSNR* y *SSIM* [44]. Se han incluido porque son las que tradicionalmente se utilizan en los trabajos de *SR* y son ampliamente conocidas dentro de la literatura.

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right)$$

Con $MAX_I$ se representa el valor de intensidad máximo $I$ mientras que con $MSE$ (*Mean Squared Error*) el error cuadrático medio de la imagen $I$.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) + (\sigma_x^2 + \sigma_y^2 + C_2)}$$

El *SSIM* se calcula en distintas ventanas de la imagen *I*, que denotamos x e y. Siendo $\mu$ el valor medio de los píxeles de la ventana, $\sigma$ la varianza de los píxeles de la ventana y $C_1$ y $C_2$ dos valores que evitan que el denominador tienda a cero.

| Método | Escala | PSNR | SSIM |
|---|---|---|---|
| BICUBIC | x2 | 30.76 | 0.9073 |
| | x4 | 26.22 | 0.7730 |
| EDSR | x2 | **32.01** | **0.9226** |
| | x4 | 26.32 | **0.7833** |
| ESPCN | x2 | 31.59 | 0.9163 |
| | x4 | 26.20 | 0.772 |
| FSRCNN | x2 | *31.66* | *0.9176* |
| | x4 | 26.12 | 0.7666 |
| LapSRN | x2 | 31.20 | 0.9124 |
| | x4 | **26.33** | *0.7777* |
| FeMaSR | x2 | 27.22 | 0.7778 |
| | x4 | 23.42 | 0.6727 |
| BSRGAN | x2 | 29.76 | 0.8864 |
| | x4 | 25.69 | 0.7492 |
| RealESRGAN | x2 | 28.93 | 0.8694 |
| | x4 | 25.42 | 0.7575 |

Tabla 4: Resultados de las pruebas de evaluación de los modelos de super-resolución. El mejor resultado en cada escala se muestra en negrita y el segundo mejor en cursiva.

No obstante, *Ledig et al.* [42] argumentan en su trabajo que el *PSNR* no es una buena métrica sobre la eficacia del modelo de *SR* debido a que una minimización en el *MSE* no implica que los detalles de la imagen se vean mejor. Por su parte, *Zhang et al.* [45] proponen una nueva métrica llamada *Feature Similarity Index Measure* o *FSIM* como alternativa a *PSNR* y *SSIM*. *Aljanabi et al.* [46] se basan en el concepto anterior de *FSIM* y desarrolla el *Information theoretic-based Statistic Similarity Measure* (*ISSM*) donde combinan teoría de la información (*Shannon entropy*), estadística (*SSIM*) y características estructurales a partir de los bordes con

el operador de Canny [47].


Figura 5: Imagen original con la zona ampliada en un recuadro verde.

Una vez descritos los modelos de super-resolución estudiados, se detallan los resultados en la Figura 4 a partir de un subconjunto aleatorio de 10 imágenes del dataset de prueba de *VisDrone*.

Si comparamos las Figuras 6 y 8 con respecto a sus imágenes originales 5 y 7 respectivamente, se puede apreciar una disminución importante en la calidad y la nitidez de la imagen generada mediante *SR* al pasar de la escala x2 a una escala x4, sobre todo en los detalles y formas más pequeñas. Por lo cual, a la vista de las pruebas realizadas, no se va a estudiar ampliar la imagen más allá de una escala de x4 en relación a la imagen original de entrada. Desde el punto de vista del ojo humano los 3 últimos modelos, que son los más recientes de los estudiados, han obtenido un mejor desempeño, sin embargo no han alcanzado valores muy altos en las métricas utilizadas, esto se debe a que, como se ha dicho arriba, no han sido entrenados para maximizarlos y utilizan en cambio sus propias métricas.

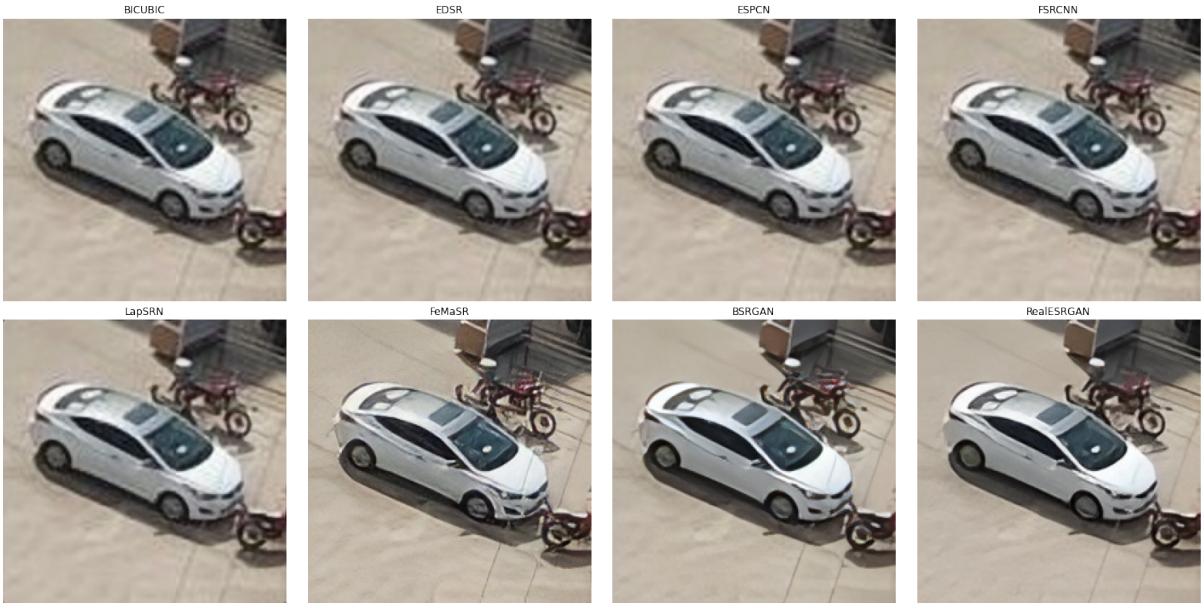


Figura 6: Imagen comparativa de los diferentes métodos de SR a escala x2.

Figura 7: Imagen original con la zona ampliada en un recuadro verde.

Por tanto, a la vista de los resultados obtenidos en las pruebas realizadas, se ha seleccionado a *Real-ESRGAN* como modelo de super-resolución para la propuesta. *BSRGAN*, que parece ofrecer incluso mejores resultados que *ESRGAN* en cuanto a detalles, se descartó en un primer momento al no ofrecer desde un principio facilidades para trabajar indistintamente a escala


Figura 8: Imagen comparativa de los diferentes métodos de SR a escala x4.

x2 y x4, y aunque se ha terminado integrado en el código generado durante la última fase del trabajo, no se han generado resultados para dicho modelo.

# 6

# Método propuesto

En este apartado, se describe el método propuesto por este trabajo para la mejora en la detección de objetos pequeños en imágenes aéreas grabadas desde drones. Como novedad, se incluye además de la aplicación de super-resolución propuesta otras técnicas de *TTA*, tales como las tablas de búsqueda o *Look-Up Tables* (*LUT*) de incremento de la luminosidad en la imagen. Durante la fase de estudio previa, se observó que el *dataset* utilizado para las pruebas contenía aproximadamente un 10 % de imágenes tomadas durante la noche, y esto afecta a la claridad con la que se muestran los objetos en el fotograma, siendo por ello más difíciles de distinguir del fondo en las imágenes nocturnas. Por otra parte se ha optado por aplicar super-resolución por partes a toda la imagen. *Iván García Aguilar* [17] la aplica únicamente en patches alrededor de objetos ya detectados previamente, este enfoque centra su atención en las zonas con mayor densidad de objetos en la imagen, pero tiene la contrapartida de que pueden quedar objetos pequeños en zonas marginales de la imagen sin ser ampliados, lo que imposibilita su detección en última instancia. En cuanto a otras técnicas de *TTA*, a diferencia de trabajos como [20] o [21], donde es posible encontrar objetos dentro de la imagen con cualquier orientación, sólo hemos incluido un *flip* o volteo horizontal a la imagen, debido a que no tiene sentido encontrarse un coche en una posición anómala. Por último, incluimos la super-resolución como herramienta principal para la mejora en la detección de objetos pequeños, donde se emplea un modelo más reciente que el observado en trabajos previos. No se han estudiado aplicar otras técnicas típicas del procesamiento de imágenes como por ejemplo el realzado de bordes o la eliminación del ruido en la imagen ya que éstas pueden llegar a afectar negativamente al rendimiento en las detecciones. El modelo de super-resolución empleado ya deja los objetos bien definidos y sin ruido.

El método presentado es visible en su totalidad en la Figura 9 y se puede acceder a través del siguiente repositorio de GitHub . Se parte de una imagen de entrada, a la que llamamos $\mathcal{I}$, de

[0] http://github.com/dancc7t/SAISOD

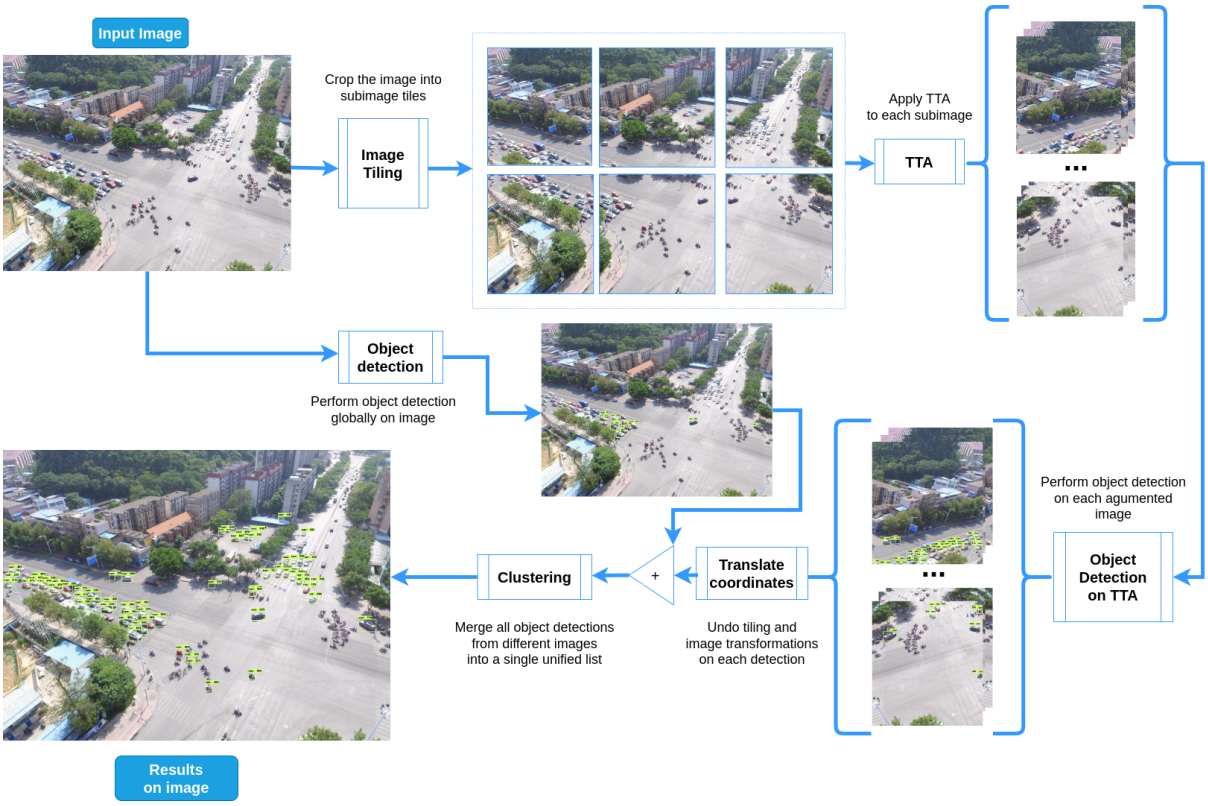


Figura 9: Diagrama del método propuesto.

un modelo detector denotado como $\mathcal{D}$, y de un modelo de super-resolución, establecido como $\mathcal{Z}$, que nos permiten trabajar sobre toda la imagen $\mathcal{D}$ a dos tamaños distintos, x1 y x2, debido a que en una misma imagen pueden aparecer objetos a múltiples escalas.

Inicialmente se pasa la imagen $\mathcal{I}$ por el modelo detector $\mathcal{D}$ ([*Object detection*] en la Figura 9), obteniendo así la primera lista de detecciones iniciales.

$$\mathcal{D}(\mathcal{I}) = \mathcal{L}_{ini}$$

Esta detección inicial se realiza con el fin de detectar los objetos de mayor tamaño en la imagen, que son identificables con mayor facilidad por el modelo $\mathcal{D}$ y que a su vez son los más propensos a ser troceados en distintos *tiles* o sub-imágenes.

Por otro lado, se divide la imagen $\mathcal{I}$ en varias sub-imágenes $\mathcal{S}_{\mathcal{I}}^{i}$ ([*Image Tiling*] en la Figura 9). Esta transformación se hace con el objetivo de que, tras aplicar super-resolución a cada

sub-imagen, $Z(\mathcal{S}_{\mathcal{I}}^{i})$, la imagen aumentada coincida lo mejor posible con el tamaño de entrada máximo permitido del modelo de detección, ya que $\mathcal{D}$ ajusta la imagen a su tamaño de entrada aplicando interpolación bicúbica, una técnica que, como hemos visto anteriormente, genera una imagen de peor calidad repercutiendo en las detecciones y por tanto a los resultados de nuestra propuesta.

$$\hat{\mathcal{S}}_{\mathcal{I}}^{i} = \arg\min_{\mathcal{S}_{\mathcal{I}}^{i}} \left[\text{size}(z \cdot \mathcal{S}_{\mathcal{I}}^{i}) - \text{size}(\mathcal{D})\right]$$

$z$ denota el factor de aumento a aplicar. A su vez, en este paso se guardará la posición relativa de cada subimagen $\mathcal{S}_{\mathcal{I}}^{i}$ respecto a la imagen original $\mathcal{I}$ de manera que luego se pueda deshacer la transformación en las detecciones de cada subimagen. Para evitar detectar un mismo objeto varias veces entre distintas sub-imágenes, se extiende cada una de ellas por los lados internos para así obtener un solapamiento de las *bounding boxes* entre objetos repartidos entre dos o más sub-imágenes y permitir así unificar las detecciones en un paso posterior.

Cabe decir que esta técnica es extrapolable a cualquier otro factor de aumento $z$. Aunque, como hemos mencionado antes, aplicar super-resolución más allá de una factor de ampliación de x4 no parece factible para este tipo de problema con los modelos actuales explorados.

Por cada una de las sub-imágenes generadas, se aplicarán hasta tres técnicas de *TTA* distintas, que son: super-resolución, *flip* horizontal y opcionalmente una transformación *LUT*, pues esta transformación sólo se aplica si el brillo medio de cada sub-imagen es inferior a un valor umbral establecido. El objetivo de este paso ([*TTA*] en la Figura 9) es, como comentamos previamente, presentar al modelo de detección la misma imagen pero de diferentes formas para así aumentar el número de detecciones en la imagen.

Por tanto, cada una de estas nuevas imágenes aumentadas se pasan por el detector de objetos $\mathcal{D}$ ([*Object Detection on TTA*] en la Figura 9), que generan a su vez tres listas de detecciones, a las que llamamos $\mathcal{L}_{SR}, \mathcal{L}_{FLIP}, \mathcal{L}_{LUT}$. Cada una de estas listas son de la forma:

$$\mathcal{L} = [a_1, a_2, ..., a_n]$$

Éstas contienen las anotaciones $a_i$ que guardan la información relativa a la detección realizada por el modelo $\mathcal{D}$. Siendo $a$ una terna de la forma:

$$a = (b, c, s)$$

$s$ representa el *score* o puntuación, también llamado *objectness score*, que otorga el modelo a la detección realizada, $c$ la clase estimada por el modelo y cuyo valor viene predeterminado por el estándar de *COCO* y por último con $b$ una lista que contiene a su vez cuatro elementos que se corresponden a las coordenadas de los cuatro vértices de la *bounding box* rectangular que localiza al objeto en la imagen:

$$b = [y_{min}, x_{min}, y_{max}, x_{max}]$$

A partir de estos datos de $b$ y de la posición relativa de cada tile o sub-imagen $\mathcal{S}_{\mathcal{I}}^{i}$ respecto de la imagen original $\mathcal{I}$ podemos localizar globalmente cada objeto detectado en la imagen si deshacemos la transformación o aumento aplicado, que en el caso de super-resolución quedaría de la siguiente forma:

$$b' = (x_i, y_i) + \frac{1}{z}b$$

Siendo $(x_i, y_i)$ las coordenadas de la esquina superior izquierda de la sub-imagen $\mathcal{S}_{\mathcal{I}}^{i}$ respecto de la imagen original y $z$ el factor de aumento aplicado, que en el caso de nuestra propuesta es 2.

Para el aumento de la *LUT* el cambio es el mismo ya que la posición de los objetos en la imagen coincide con la de la imagen aumentada con super-resolución pues sólo se ha aumentado el brillo medio de la imagen. El aumento de *flip* horizontal sí requiere deshacer el cambio de coordenadas en el eje $X$ de la imagen, antes de aplicar la transformación anterior, quedando $b$ de la siguiente manera:

$$b = [y_{min}, w - x_{max}, y_{max}, w - x_{min}]$$

Siendo $w$ la anchura de la sub-imagen $\mathcal{S}_{\mathcal{I}}^{i}$ multiplicada por el factor de aumento $z$. Una vez tenemos todas las anotaciones obtenidas en los pasos anteriores:

$$\mathcal{L} = \mathcal{L}_{ini} \cup \mathcal{L}_{SR} \cup \mathcal{L}_{FLIP} \cup \mathcal{L}_{LUT}$$

Ya agrupadas en el mismo sistema de coordenadas de la imagen $\mathcal{I}$, se procede a unificarlas en una lista final de anotaciones donde, idealmente, cada anotación $a_i$ se corresponde con únicamente un objeto detectado en la imagen de entrada $\mathcal{I}$. Para ello se emplea un método de clustering ([*Clustering*] en la Figura 9). Éste consiste en agrupar la lista de entrada $\mathcal{L}$ en una lista de *clusters* $\mathcal{L}_K = [k_1, k_2, ..., k_n]$, donde cada cluster contiene las anotaciones que son similares entre sí en el sentido que sus vectores de características $(b, c, s)$ son lo suficientemente parecidos entre sí como para ser considerados una misma detección de un mismo objeto.

$$k = \{a_i, a_j \in \mathcal{L} \ / \ IOU(b_i, b_j) \geq threshold_{IoU} \wedge c_i = c_j\}$$

El criterio de agrupación entre cada par de anotaciones $a_i, a_j$ se basa por tanto en la cercanía entre sus respectivas *bounding boxes*. Si el *IoU* de ambas es mayor que un valor umbral, al que llamamos $threshold_{IoU}$ y su clase es la misma entonces las agrupamos en un mismo *cluster*.

Como veremos más adelante, al presentar los resultados, establecer el valor más adecuado de $threshold_{IoU}$ no es una tarea fácil teniendo en cuenta que estamos trabajando con un dataset donde las imágenes han sido tomadas con distintos ángulos de incidencia sobre el terreno y donde pueden aparecer muchos objetos solapados entre sí en un mismo espacio. Por último se devuelve una lista definitiva de anotaciones donde se obtiene la anotación de cada cluster con el score más alto otorgado por el modelo de detección $\mathcal{L}_{out}$. Estas anotaciones obtenidas en la salida pueden utilizarse luego en otros métodos y en otras aplicaciones, ya dependiendo del uso que se les quiera dar, como por ejemplo mostrar los objetos detectados en la imagen entre otros.

Antes de presentar los resultados obtenidos durante la fase de pruebas se explica brevemente una de las técnicas de *TTA* aplicadas, la *LUT* o *Look-Up Table*, y se muestra un ejemplo de su uso.

Una *LUT* es una transformación lineal sobre el dominio espacial de la imagen que establece una relación binaria entre intensidades de color en la imagen. Funciona como una tabla que contiene los 256 posibles valores de intensidad en la imagen y donde cada valor de intensidad de entrada (*input value*) indexa o permite encontrar su respectivo valor de intensidad de salida (*output value*).

Se ha optado por incluir únicamente tipos de *LUT* que incrementan el brillo medio o luminosidad en imágenes predominantemente oscuras, imágenes tales que su momento central de orden uno, es decir la media de los valores de intensidad de cada pixel en la imagen, sea menor que un valor umbral al que llamamos $threshold_{LUT}$. Estas *LUTs* se obtienen de manera dinámica dependiendo de la luminosidad de cada imagen y del $threshold_{LUT}$ que definamos para las pruebas. El rango del valor umbral estudiado en distintas pruebas ha sido entre 30 y 60, subir este valor mitiga el efecto de la *LUT* en imágenes cada vez más iluminadas y bajarlo reduce el número de imágenes que pueden utilizar esta transformación, finalmente se ha escogido un valor de 50, que es el empleado a partir de ahora en todas las pruebas. Debido a que en torno a un 10 % de las imágenes del dataset de prueba contiene imágenes tomadas durante la noche, como se ha comentado anteriormente, su aportación a la mejora en la detección de objetos va a tener por tanto menos alcance que por ejemplo el flip horizontal de la imagen, que

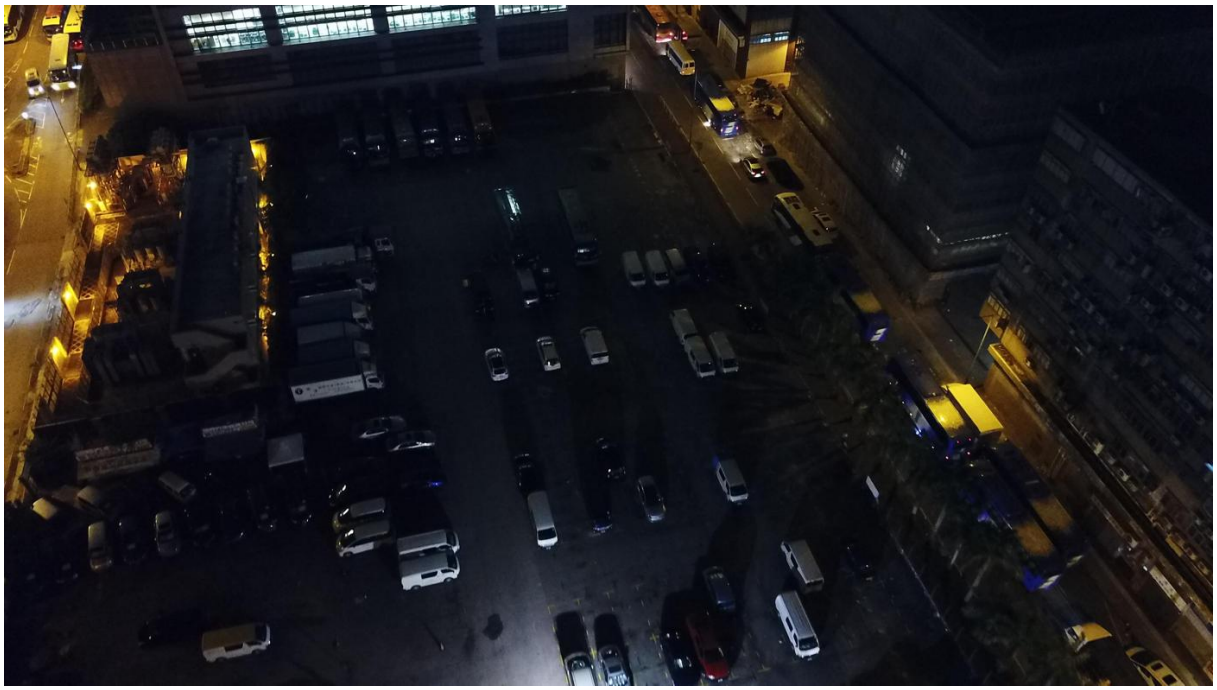

Figura 10: Imagen original oscura extraída del dataset de *Visdrone*.

se aplica indistintamente a todas las imágenes. Sin embargo, es un aumento fácil de aplicar y de bajo coste computacional que puede ayudar a mejorar la detección de objetos en imágenes oscuras como la que aparece en la Figura 10.

En la Figura 11 se muestran las tres mejores transformaciones que mejores resultados han mostrado durante las pruebas. La primera, a la que llamamos *increase* aumenta abruptamente la intensidad y el contraste en las tonalidades bajas de la imagen, a costa de saturar el resto de tonalidades más altas. Es la que resalta con mayor intensidad las partes oscuras de la imagen pero también la que mayor distorsión añade a la imagen de salida, en comparación con los otros métodos. La transformación de tipo *gamma*, más conocida como *Gamma correction* o *Power Law Transform*, es ampliamente utilizada en el campo del procesamiento de imágenes y video para hacer correcciones a la luminosidad de la imagen. Por último la transformación de tipo *clahe*, cuyo nombre viene de *Contrast Limited Adaptive Histogram Equalization* o *CLAHE*, es una operación ya definida dentro de la librería de opencv y que aplica como su nombre indica una ecualización del histograma a toda la imagen pero por regiones o *tiles* para reducir el aumento de contraste generado por las divergencias en las tonalidades de distintas regiones de la imagen. Esta última transformación es la que mejores resultados ha ofrecido en las pruebas

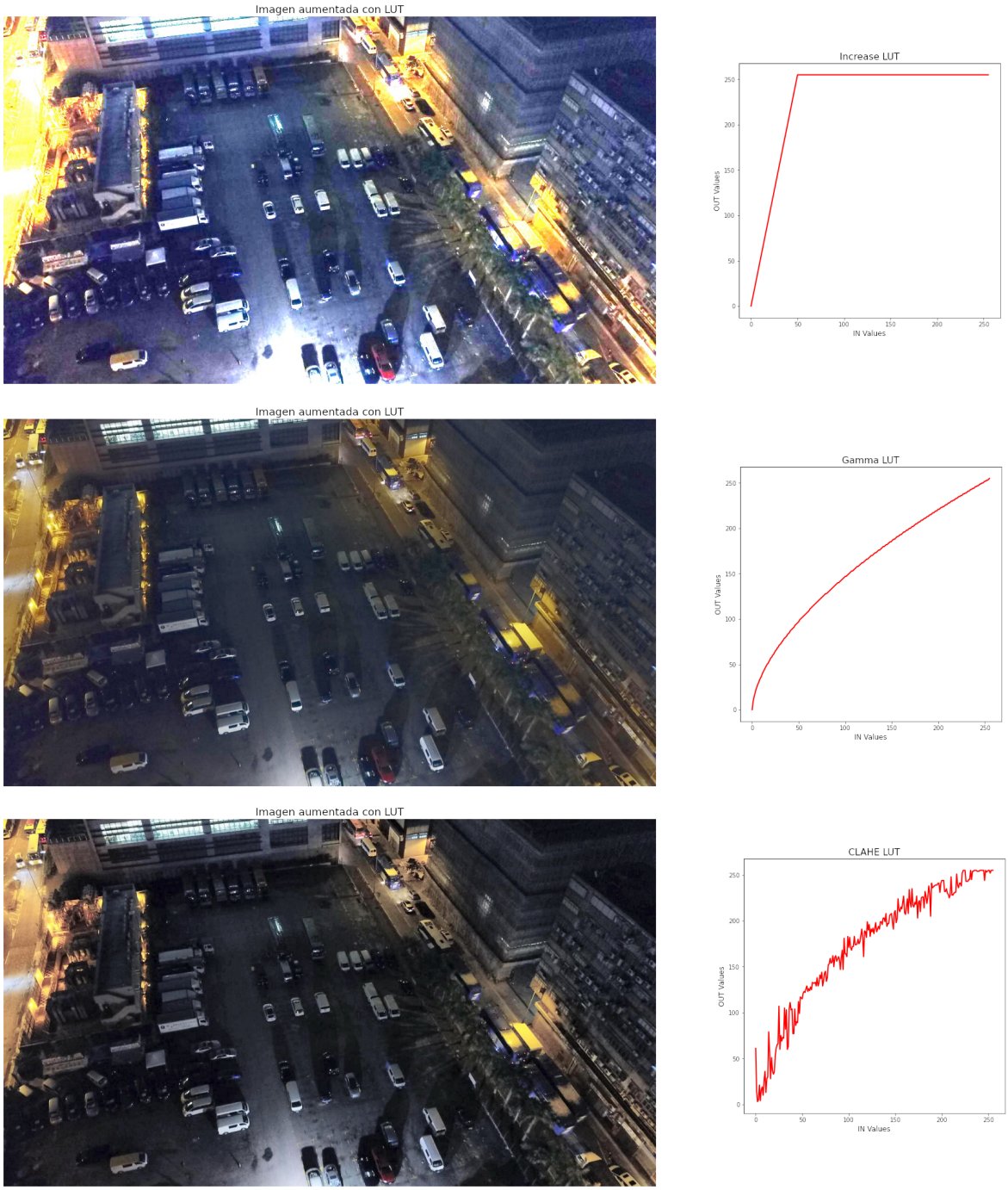


Figura 11: Imágenes transformadas y sus respectivas LUTs representadas en una gráfica.

realizadas y es por tanto el tipo que se ha elegido. De ahora en adelante, cuando hablemos de *LUT* nos estamos refiriendo a esta transformación de tipo *clahe* en concreto.

# 7

# Resultados

En las pruebas establecidas en la fase de resultados, se diferencia entre las dos propuestas principales que se han empleado en este trabajo: Una de ellas, a la que llamamos *PSR* o Propuesta de *SR*, consiste en mejorar el *accuracy* del modelo de detección de objetos aplicando como técnica *TTA* únicamente *SR* a escala x2 con *Image Tiling*. La otra, a la que llamamos *PSRFLUT* o Propuesta de *SR*, *Flip* y *LUT*, es la aplicación conjunta de las tres técnicas de *TTA* vistas en este trabajo. Aparte de esas dos propuestas, se definen otras dos categorías a las que llamamos *BASE* o detección con el modelo de detección original sin ninguna modificación, y *SRO* o detección con sólo *SR*, que sirven como comparativa base del modelo de detección y del modelo de super-resolución respectivamente.

Estas variaciones en la propuesta se han realizado con el objetivo de poder evaluar con mayor detalle el aporte de cada una de las técnicas *TTA* aplicadas en las distintas pruebas. No obstante, el aporte de las técnicas de *Flip* horizontal y *LUT* por separado sobre la imágen no ha sido lo suficiente destacable en cuanto a diferencia entre las distintas métricas de evaluación obtenidas tras las pruebas realizadas, por lo que sólo se incluyen ambas juntas en la tabla de resultados y las gráficas en la propuesta definitiva de *PSRFLUT*, donde ya sí existe una aportación más apreciable.

En este apartado se presentan la tabla de los resultados obtenidos tras la fase de pruebas así como varias gráficas e imágenes que ayuden a interpretar mejor estos datos. Como ya se citó anteriormente, las pruebas se han realizado sobre un subconjunto de 300 imágenes seleccionadas aleatoriamente del dataset de prueba *VisDrone_2019-DET-test-dev*, y han sido obtenidas empleando como *GPU* una *NVIDIA Tesla T4*. En las pruebas realizadas se infiere únicamente sobre la clase 'Coche' o '*Car*', pues corresponde con la clase que aparece con más frecuencia en las imágenes del dataset de prueba. Se ha elegido sólo una clase con el objetivo de evaluar de una manera más consistente los distintos métodos abordados en este trabajo.

La selección de los distintos hiperparámetros empleados no ha sido una tarea trivial, pues, como se enunció anteriormente, la detección de objetos pequeños desde un dataset como *VisDrone* [3] presenta dificultades añadidas a la propia tarea de detectar objetos en una imagen, como son la variación en la altitud del aparato, el ángulo de incidencia de la cámara sobre el terreno o las distintas condiciones lumínicas. El hiperparámetro más importante en cuanto su efecto en los resultados obtenidos es el umbral de detección, que llamamos $threshold_{score}$, el cual determina si una detección es aceptada o no en la lista de anotaciones $\mathcal{L}$ en función de su *score* o puntuación.

Debido a que estamos abordando el problema de la detección de objetos pequeños, tenemos que tener en cuenta que los modelos actuales presentan bastantes dificultades para detectarlos de manera efectiva en comparación a los objetos de mayor tamaño. Esto lógicamente influye en el *score* que reciben. Un objeto pequeño es más difícil de detectar y por ello su puntuación media es normal que sea inferior a la que obtienen los objetos de mayor tamaño. Por dicho motivo, no podemos utilizar el mismo $threshold_{score}$ en un método centrado en detectar objetos pequeños que el que se usaría para detectar principalmente objetos grandes. En la Figura 12 se muestran las puntuaciones medias según el tipo de objeto detectado y el método empleado a un umbral mínimo de detección de 0.2. Como vemos en la Figura, el score medio de los objetos pequeños es aproximadamente la mitad que el de los grandes. La aplicación de nuestra propuesta consigue reducir ligeramente la diferencia entre los tres grupos, especialmente entre objetos medianos y grandes.

Durante la primera fase de pruebas se utilizó 0.1 como el umbral más bajo, pero al visualizar las detecciones en las imágenes aparecían demasiados falsos positivos en la imagen, pues un umbral tan bajo permite al detector de objetos confundir con mayor facilidad cualquier cosa que aparezca en la imagen con un coche, por lo que se decidió subir a 0.2. El umbral más alto queda en 0.5 pues seguir subiendo provoca que muy pocos objetos pequeños detectados lleguen a ser incluidos en la lista de anotaciones de salida. Bajar el umbral permite entonces mejorar la detección de objetos pequeños con los modelos actuales pero también tiene un efecto negativo basado en el aumento de los falsos positivos como veremos más adelante en algún ejemplo visual.

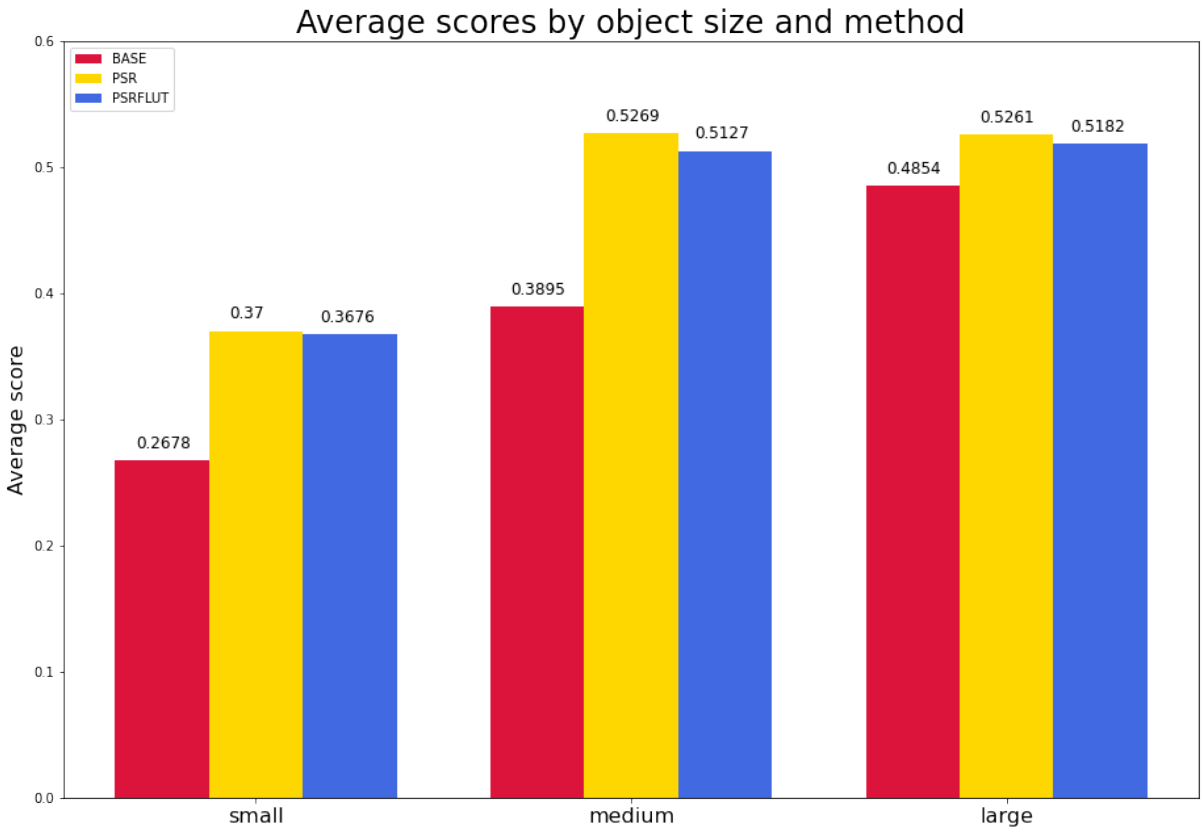


Figura 12: Gráfica de puntuación media según tamaño de objeto y método utilizado.

Dicho esto, se han realizado pruebas a distintos umbrales de detección (0.2, 0.3 y 0.5), uno por cada tabla como se muestra en la Tabla 13, para ver cómo influían en los distintos *mAP* del evaluador de *COCO* para los distintos métodos.

Cada tabla muestra los distintos métodos y propuestas analizados contra las distintas métricas que devuelve el evaluador de *COCO*. Como podemos apreciar, la aplicación de únicamente super-resolución (*SRO*) sobre la imagen ya mejora sustancialmente el *mAP* en los objetos pequeños y también en los medianos, pero no en los grandes, donde de hecho el *mAP* baja un poco. De igual manera, el modelo de detección sin ninguna modificación conocido como (*BASE*), baja de un 7.2 % de *mAP* para objetos pequeños con un umbral de detección de 0.2 a un 0.8 % de *mAP* si subimos el umbral a 0.5, estos resultados experimentales vienen a corroborar lo mencionado arriba. La aplicación de nuestras propuestas *PSR* y *PSRFLUT* consigue mejoras en todas las métricas salvo en la de los objetos grandes, donde baja ligeramente debido en parte al problema con el $threshold_{IOU}$ que trae consigo la división de la imagen de entrada

| Method | IoU=[0.5:0.95] area = all | IoU > 0.5 area = all | IoU > 0.75 area = all | IoU=[0.5:0.95] area = small | IoU=[0.5:0.95] area = medium | IoU=[0.5:0.95] area = large |
|---|---|---|---|---|---|---|
| BASE | 0.204 | 0.322 | 0.221 | 0.072 | 0.340 | **0.628** |
| SRO | 0.320 | 0.497 | 0.350 | 0.208 | 0.441 | 0.488 |
| PSR | *0.333* | *0.511* | *0.364* | *0.210* | *0.456* | 0.626 |
| PSRFLUT | **0.337** | **0.522** | **0.368** | **0.216** | **0.457** | *0.627* |

| Method | IoU=[0.5:0.95] area = all | IoU > 0.5 area = all | IoU > 0.75 area = all | IoU=[0.5:0.95] area = small | IoU=[0.5:0.95] area = medium | IoU=[0.5:0.95] area = large |
|---|---|---|---|---|---|---|
| BASE | 0.156 | 0.225 | 0.179 | 0.033 | 0.282 | 0.611 |
| SRO | 0.292 | 0.436 | 0.325 | *0.175* | 0.418 | 0.470 |
| PSR | *0.302* | *0.449* | *0.339* | *0.175* | *0.428* | **0.622** |
| PSRFLUT | **0.311** | **0.464** | **0.346** | **0.188** | **0.436** | *0.620* |

| Method | IoU=[0.5:0.95] area = all | IoU > 0.5 area = all | IoU > 0.75 area = all | IoU=[0.5:0.95] area = small | IoU=[0.5:0.95] area = medium | IoU=[0.5:0.95] area = large |
|---|---|---|---|---|---|---|
| BASE | 0.080 | 0.102 | 0.099 | 0.008 | 0.147 | 0.469 |
| SRO | 0.210 | *0.295* | 0.244 | *0.086* | 0.348 | 0.400 |
| PSR | *0.216* | 0.294 | *0.252* | *0.086* | *0.353* | *0.541* |
| PSRFLUT | **0.230** | **0.319** | **0.266** | **0.100** | **0.368** | **0.545** |

Tabla 13: Resultados de la evaluación de los distintos métodos a tres umbrales de detección distintos (0.2, 0.3 y 0.5). El mejor resultado en cada métrica se muestra en negrita y el segundo mejor en cursiva.

en tiles para poder aplicar super-resolución de manera más efectiva.

Todas las Tablas de la Figura 13 emplean un $threshold_{IOU}$ de 0.25, éste es otro hiperparámetro que influye a la hora de generar la lista de anotaciones de salida, pues anteriormente se han tenido que combinar todas las anotaciones geneneradas por las distintas variaciones de la misma imagen presentadas al modelo detector. Como se ha explicado anteriormente, el $threshold_{IOU}$ sirve para determinar si dos detecciones distintas se unifican en una sola detección.

En imágenes tomadas a menor altura y con mayor perspectiva es más común que aparezca solapamiento entre los objetos que en imágenes tomadas a mayor altura. Con lo cual si utilizamos un $threshold_{IOU}$ muy alto estamos reduciendo el problema del solapamiento pero a costa de dejar potencialmente más detecciones para un mismo objeto localizado en una región de la imagen compartida por dos o más *tiles* o sub-imágenes. Por otra parte, un $threshold_{IOU}$ muy bajo nos asegura que los objetos aislados que sean detectados van a tener una única *bounding box*, pero a cambio se reduce el número de detecciones en imágenes con objetos solapados donde se unifican sus *bounding boxes* en una sóla. Debido a esto, no existe un parámetro ideal que resuelva a la vez ambos problemas. En nuestro caso, aplicamos *NMS* a través del *Cluste-*

*ring* probando con tres valores umbrales de $threshold_{IOU}$ distintos, 0.1, 0.25 y 0.5, para elegir empíricamente el que mejores resultados genere tanto a nivel de *mAP* general como de visualización de imágenes con gran cantidad de objetos pequeños en particular.

Ambos hiperparámetros, $threshold_{score}$ y $threshold_{IOU}$ nos ayudan a entender entonces las siguientes gráficas, donde muestra el número de detecciones por imagen y método con dos $threshold_{score}$ distintos, 0.3 (Figura 14) y 0.5 (Figura 15). La etiqueta *GT* hace referencia al *ground-truth* o solución para el fotograma, que en este caso, es el número de objetos que realmente aparecen en cada imagen.

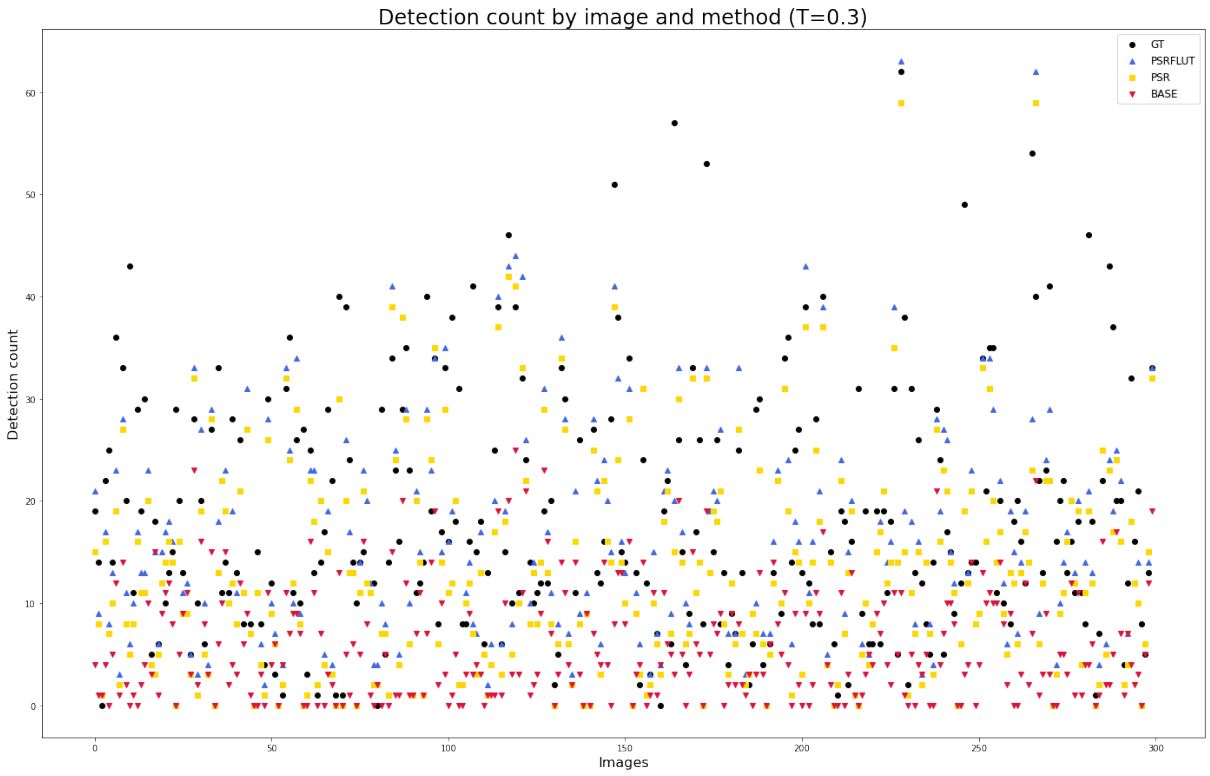


Figura 14: Número de detecciones por imágen y método a un umbral de 0.3.

Como podemos apreciar en las dos figuras anteriores, el número de detecciones desciende de forma generalizada en los distintos métodos conforme subimos el $threshold_{score}$ de 0.3 a 0.5. Esto afecta en especial al modelo original de base sin ninguna otra aportación nuestra denotado como (*BASE*), el cual posee mayores dificultades para generar detecciones con un *score* lo suficientemente alto. La causa por la cual se amontonan los puntos de *BASE* en el

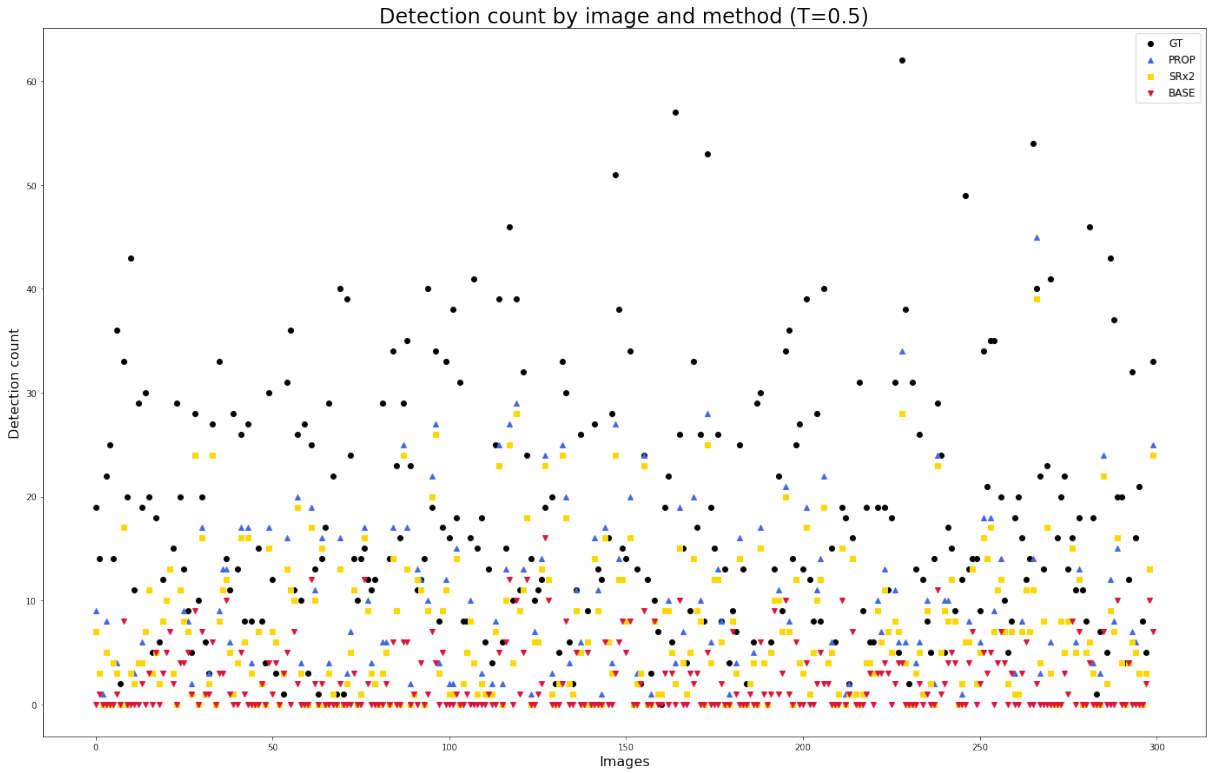


Figura 15: Número de detecciones por imágen y método a un umbral de 0.5.

fondo es provocado por la alta proporción de objetos pequeños en las diferentes imágenes del dataset de prueba. Por otra parte, a 0.3 se observa con más facilidad como algunas imágenes tienen más detecciones en los métodos propuestos que objetos de la clase 'coche' hay en la imagen realmente, esto se debe tanto a las detecciones múltiples de un mismo objeto debido al problema del $threshold_{IOU}$ como a la presencia de falsos positivos en la imagen detectada, que son más fáciles de encontrar a un umbral de 0.3 que a uno de 0.5.

En estas dos últimas gráficas de las figuras 16 y 17, podemos observar como contra más pequeño es el objeto menos detecciones se consiguen generar. La aplicación de nuestra propuesta consigue mejorar bastante el número de detecciones que ofrece el modelo de *BASE* en las áreas más pequeñas, sobre todo en el umbral de detección de 0.3, mientras que los valores convergen conforme nos acercamos a las áreas más grandes. Las líneas de puntos en la gráfica del histograma nos permiten distinguir las áreas correspondientes a objetos pequeños, medianos y grandes según el estándar de *COCO* mencionado anteriormente.

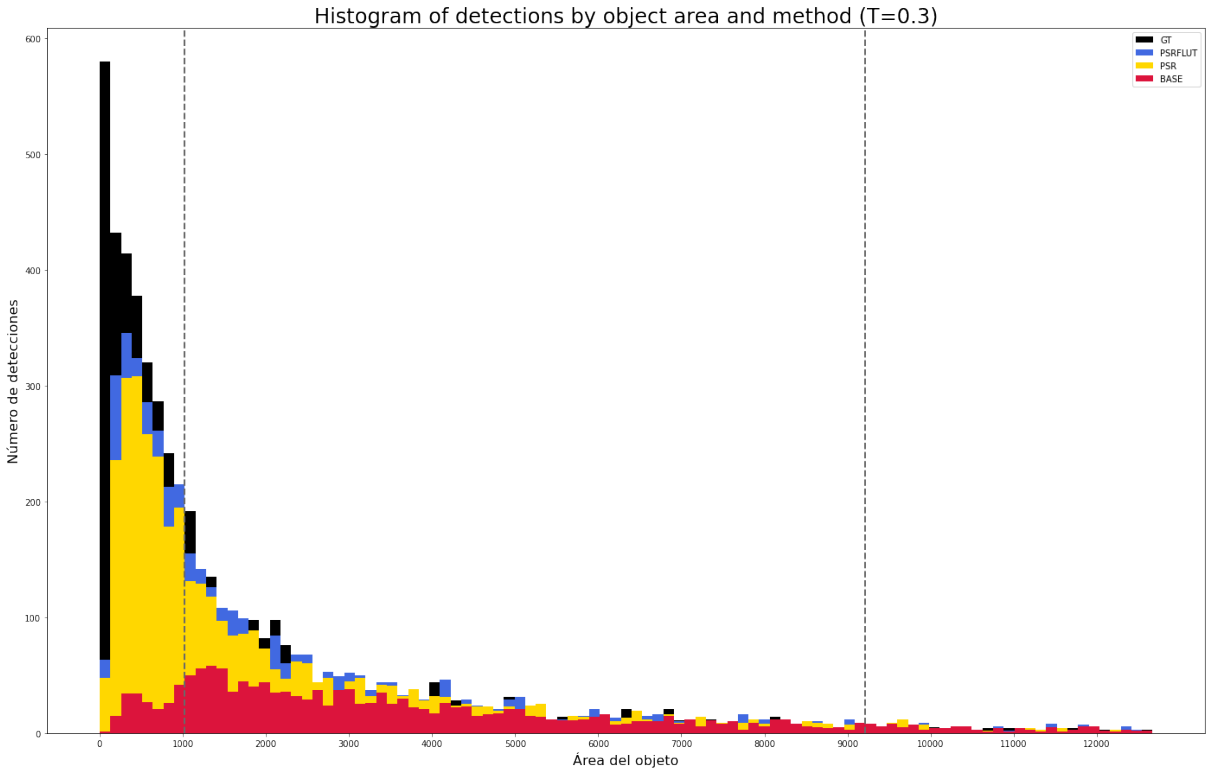


Figura 16: Histograma de detecciones según el área de cada objeto detectado y el método aplicado a un umbral de 0.3.

Pese a que los resultados que muestra la Figura 16 parecen bastante buenos en cuanto a número de detecciones de objetos, es necesario matizarlos, ya que el número de detecciones generadas no significa precisamente que se estén detectando realmente tantos objetos de manera efectiva en la imagen. A continuación, se presenta la Tabla comparativa 18, la cual muestra el número total y la ratio en la detección de objetos según el método aplicado y se compara a su vez con el *mAP* de *COCO* mostrado previamente en la Figura 13, esto nos puede dar una idea más cercana al rendimiento real de cada propuesta en términos relativos.

En esta Tabla 18, se puede ver que el aumento en las detecciones es mayor al aumento del *mAP* de *COCO*. Algo que nos muestra que no todo el aumento en la detección de objetos se traduce en un aumento tangible de objetos reales detectados en la imagen. En el caso ideal, si todas las detecciones fueran correctas, la diferencia, $\delta$, entre la ratio de detección frente al *ground-truth* y el *mAP* correspondiente tendería a cero, pero estadísticamente es más probable que aparezcan más falsos positivos cuanto más detecciones se hayan realizado.

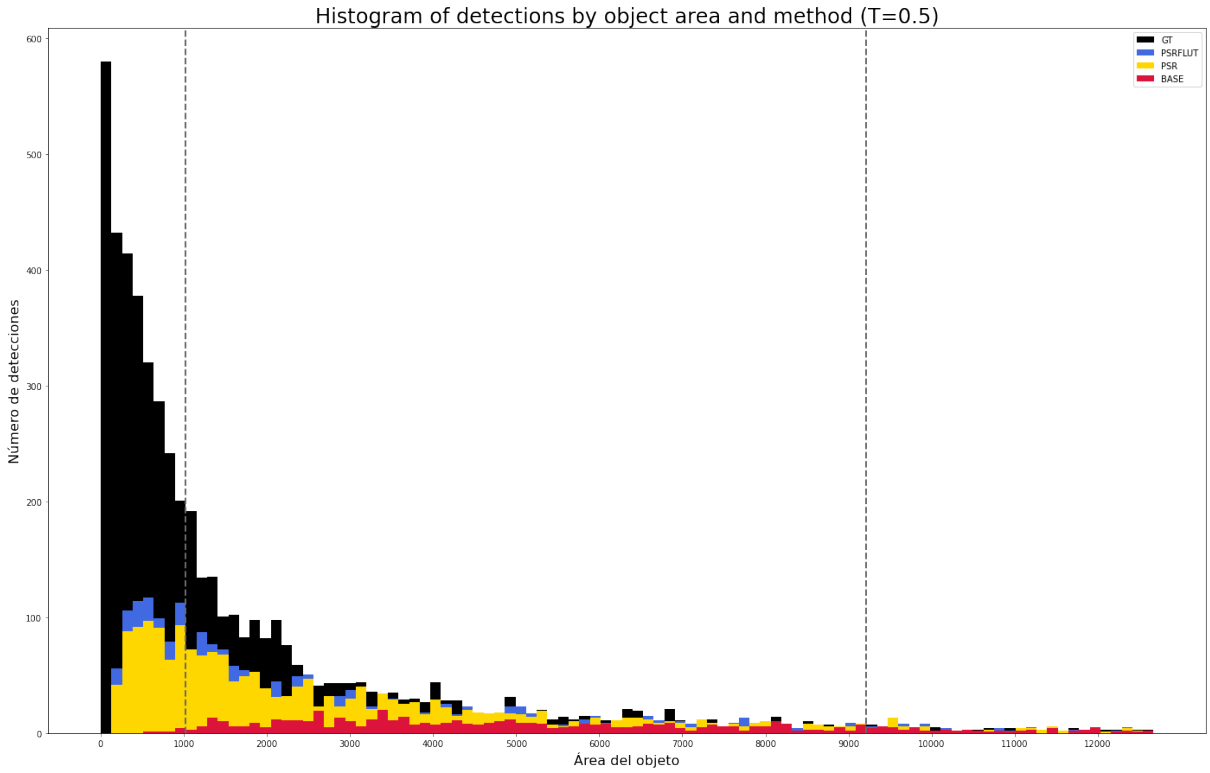


Figura 17: Histograma de detecciones según el área de cada objeto detectado y el método aplicado a un umbral de 0.5.

Dicho esto, podemos concluir en que la aplicación de nuestra propuesta consigue claramente mejorar de manera sustancial la detección efectiva de objetos pequeños en imágenes empleando un modelo de detección que no ha sido entrenado para esta tarea en concreto.

Por último, se muestran algunas imágenes donde se compara visualmente la detección con el modelo original (*BASE*) junto a nuestra propuesta definitiva (*PSRFLUT*). Véase por ello las Figuras 19, 20, 21 y 22.

| | method | $threshold_{score}$ | detections | ground-truth | ratio | mAP | δ(ratio, mAP) |
|---|---|---|---|---|---|---|---|
| small objects | BASE | 0.3 | 200 | 2854 | 0.07 | 0.033 | 0.037 |
| | | 0.5 | 7 | 2854 | 0.0024 | 0.008 | -0.0056 |
| | PSR | 0.3 | 1769 | 2854 | 0.6198 | 0.174 | 0.4458 |
| | | 0.5 | 566 | 2854 | 0.1983 | 0.086 | 0.1123 |
| | PSRFLUT | 0.3 | 2017 | 2854 | 0.7067 | 0.188 | 0.5187 |
| | | 0.5 | 684 | 2854 | 0.2396 | 0.100 | 0.1396 |

| | method | $threshold_{score}$ | detections | ground-truth | ratio | mAP | δ(ratio, mAP) |
|---|---|---|---|---|---|---|---|
| medium objects | BASE | 0.3 | 1317 | 2162 | 0.6091 | 0.282 | 0.3271 |
| | | 0.5 | 491 | 2162 | 0.2271 | 0.147 | 0.0801 |
| | PSR | 0.3 | 2031 | 2162 | 0.9394 | 0.428 | 0.5114 |
| | | 0.5 | 1367 | 2162 | 0.6322 | 0.353 | 0.2792 |
| | PSRFLUT | 0.3 | 2254 | 2162 | 1.0425 | 0.436 | 0.6065 |
| | | 0.5 | 1468 | 2162 | 0.679 | 0.368 | 0.311 |

| | method | $threshold_{score}$ | detections | ground-truth | ratio | mAP | δ(ratio, mAP) |
|---|---|---|---|---|---|---|---|
| large objects | BASE | 0.3 | 251 | 187 | 1.3422 | 0.611 | 0.7312 |
| | | 0.5 | 149 | 187 | 0.7967 | 0.469 | 0.3277 |
| | PSR | 0.3 | 182 | 187 | 0.9732 | 0.622 | 0.3512 |
| | | 0.5 | 169 | 187 | 0.9037 | 0.541 | 0.3627 |
| | PSRFLUT | 0.3 | 257 | 187 | 1.3743 | 0.620 | 0.7543 |
| | | 0.5 | 176 | 187 | 0.9411 | 0.545 | 0.3961 |

Tabla 18: Comparativa entre el *ratio* de objetos detectados y su respectivo *mAP* obtenido por método y umbral de detección.

Figura 19: Comparativa obtenida en una imagen de baja luminosidad (representada en la esquina superior izquierda), entre el modelo original de base (esquina superior derecha) y nuestra propuestas *PSR* (esquina inferior izquierda) y *PSRFLUT* (esquina inferior derecha).

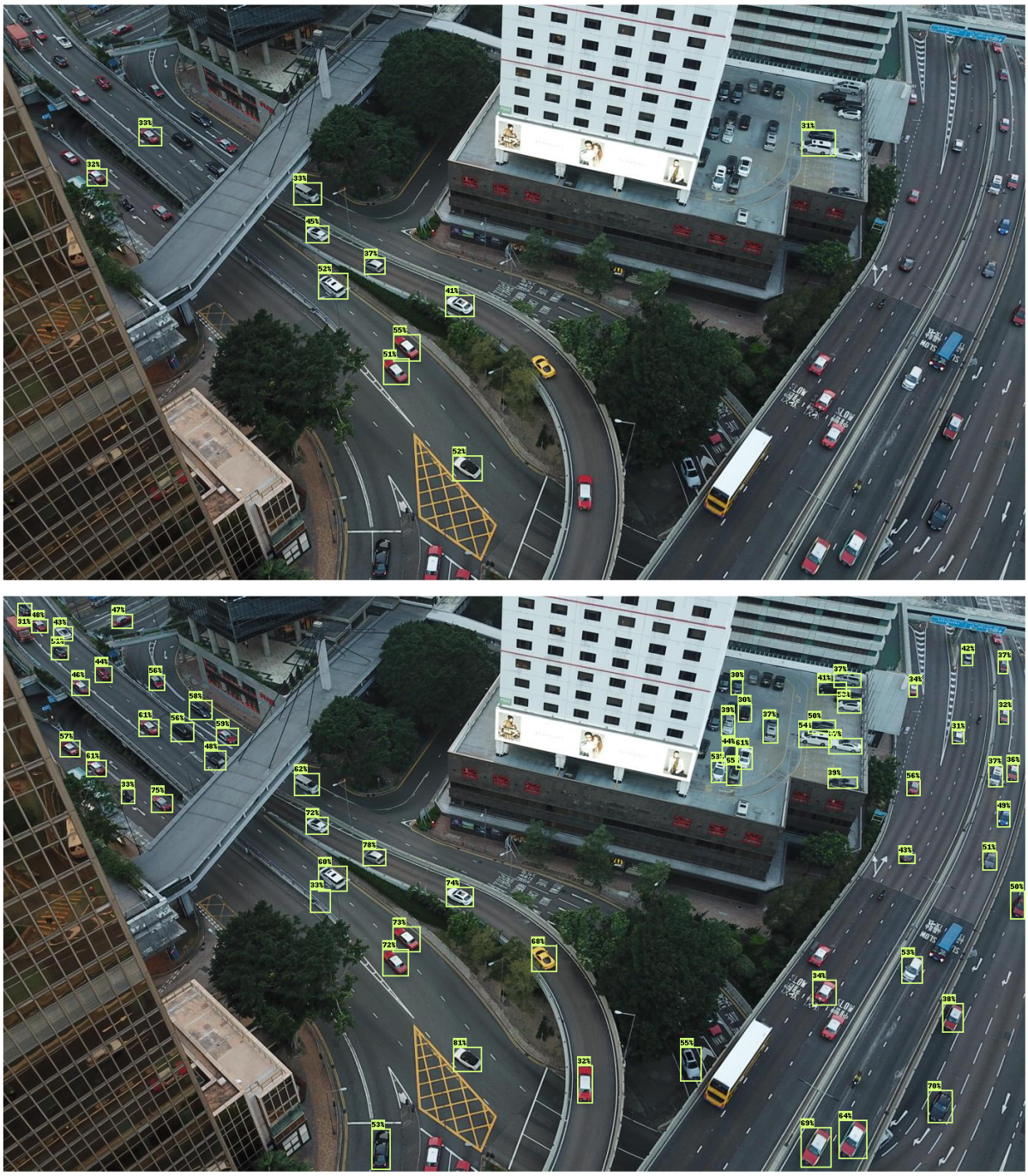

Figura 20: Comparativa entre detecciones del modelo base (parte superior) y nuestra propuesta (parte inferior).


Figura 21: Comparativa entre detecciones del modelo base (parte superior) y nuestra propuesta (parte inferior).



Figura 22: Comparativa entre detecciones del modelo base (parte superior) y nuestra propuesta (parte inferior).

# 8

# Conclusiones y Líneas Futuras

A pesar de que la detección de objetos en imágenes es un tema que cada vez se está investigando más y adquiriendo un mayor protagonismo por sus múltiples aplicaciones, todavía sigue existiendo una diferencia importante en cuanto a la tasa de detección entre objetos grandes y pequeños pese a los avances conseguidos en estos últimos años.

Aparte de un nuevo paradigma de modelo capaz de afrontar este problema en concreto, para seguir reduciendo la diferencia entre la detección de objetos de pequeño tamaño frente a los de tamaño grande será necesario aumentar los datasets ya disponibles o crear otros nuevos que permitan la creación de nuevos modelos entrenados específicamente para esta tarea, con otro tipo de arquitectura distinta a la empleada actualmente, más pensada para detectar objetos de mayor tamaño. Mientras tanto, el aplicar distintas técnicas de aumento durante la fase de inferencia, conocidas como *TTA*, se presenta como una alternativa más viable y asequible de abordar en estos momentos para mejorar la tasa de detección de objetos pequeños con los modelos que tenemos disponibles actualmente, de todas las técnicas disponibles, super-resolución es sin duda la más efectiva de cara a mejorar el rendimiento de los modelos actuales. Los continuos avances y estudios realizados en el campo de la super-resolución son una de las claves que pueden ayudar a seguir mejorando en este sentido. *BSRGAN* [43], uno de los últimos modelos de super-resolución que han sido publicados, parece que llega incluso a mejorar el modelo utilizado en este trabajo, aunque tiene el inconveniente que actualmente no permite utilizar de forma externa y sin modificaciones en el código más de un factor de escalado.

Nuestra propuesta ha conseguido mejorar la tasa de detección de objetos pequeños en imágenes aéreas del dataset de VisDrone [3], elevando en más de 10 puntos el mAP obtenido por modelos pre-entrenados con MS COCO [1]. Pero, como hemos visto antes con los resultados obtenidos, el aumento en la detección de los objetos más pequeños sigue siendo todavía una tarea con bastante margen de mejora y quedan algunos aspectos por mejorar de esta propuesta, como la dicotomía que existe entre dar más prioridad a reducir el solapamiento de objetos

distintos en una misma ventana o dar más prioridad a unificar detecciones múltiples de un mismo objeto como consecuencia de la aplicación de técnicas *TTA* a través de *Image Tiling*.

De cara a líneas futuras de trabajo, un aspecto a mejorar de nuestra propuesta sería abordar por separado este problema del $threshold_{IOU}$ antes mencionado, una opción para resolverlo sería quizás utilizar un valor umbral dependiente del área del objeto, ya que los objetos grandes son más propensos a quedar con más de una bounding box pues al ser más grandes y ocupar más espacio en la imagen tienden a ser divididos entre varios tiles con más frecuencia. Otra opción sería distinguir entre solapamiento de bounding boxes en un mismo tile, consecuencia de solapamiento entre objetos en la imagen, y solapamiento de bounding boxes entre varias sub-imágenes, consecuencia de las técnicas de aumento empleadas en la propuesta.

Otra posible mejora a nuestra propuesta sería incluir un nuevo aumento respecto a la imagen original con los objetos rotados y alineados horizontalmente si previamente se obtiene el heading u orientación de cada objeto a través de sus respectivos autovectores (*eigenvectors*). Esto permitiría presentar al modelo de detección el mismo objeto pero de otra manera también natural. Se ha observado durante las pruebas visuales de las detecciones sobre imágenes que el modelo empleado tenía mayor facilidad a la hora de detectar vehículos que estaban totalmente de perfil, mientras que por otra parte se dejaba sin detectar otros que estaban muy inclinados en diagonal o en perpendicular a la cámara.

Adicionalmente, como trabajo futuro establecemos la aplicación de super-resolución a una escala x4 de manera generalizada a toda la imagen. Se abordó su aplicación en regiones de la imagen donde se encuentran objetos ya previamente detectados pero cuyo score no llega a un valor umbral máximo, el $threshold_{high}$. Una posible mejora a este intento de rescatar aquellas detecciones cercanas al umbral máximo sería aumentar la sub-imagen donde se encuentran aplicando de nuevo Image Tiling y super-resolución x4 respecto a la imagen original. Esto reduciría el enorme coste computacional que suponía aplicar esta primera idea y a su vez resolvería el problema del fondo artificial aumentado en el entorno de los objetos detectados. El uso de dos valores umbrales en vez de uno se basa en un concepto ya visto dentro de la visión por computador, el operador de Canny [47], que se emplea para la detección de bordes en imágenes, hace uso de dos umbrales, uno alto y otro bajo, para tratar de cerrar los bordes

en la imagen cuyo valor no llega al umbral alto pero sí que está por encima del bajo. La idea en este caso es la misma pero aplicada a rescatar los objetos que están cerca de ser detectados, por encima del $threshold_{low}$, pero que no llegan al umbral alto, el $threshold_{high}$. Tras observar y analizar los resultados obtenidos en las pruebas se ha establecido un punto de inflexión en el score entre 0.2 y 0.3, dependiendo de la imagen, donde se dejan escapar muchas detecciones si el umbral de detección es superior.

# Bibliografía


[1] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: Common Objects in Context," en *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele y T. Tuytelaars, eds., Cham: Springer International Publishing, 2014, págs. 740-755, ISBN: 978-3-319-10602-1.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li y L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," en *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, págs. 248-255. DOI: 10.1109/CVPR.2009.5206848.

[3] Y. Cao, Z. He, L. Wang et al., "VisDrone-DET2021: The Vision Meets Drone Object detection Challenge Results," en *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, págs. 2847-2854. DOI: 10.1109/ICCVW54120.2021.00319.

[4] H.-Y. Lin, K.-C. Tu y C.-Y. Li, "VAID: An Aerial Image Dataset for Vehicle Detection and Classification," *IEEE Access*, vol. 8, págs. 212 209-212 219, 2020. DOI: 10.1109/ACCESS.2020.3040290.

[5] G. Cheng, X. Yuan, X. Yao, K. Yan, Q. Zeng y J. Han, *Towards Large-Scale Small Object Detection: Survey and Benchmarks*, 2022. DOI: 10.48550/ARXIV.2207.14096. dirección: https://arxiv.org/abs/2207.14096.

[6] G.-S. Xia, X. Bai, J. Ding et al., "DOTA: A Large-Scale Dataset for Object Detection in Aerial Images," en *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, págs. 3974-3983. DOI: 10.1109/CVPR.2018.00418.

[7] Y. Gong, *iSAID-Reduce100*, 2021. DOI: 10.21227/6sab-0964. dirección: https://dx.doi.org/10.21227/6sab-0964.

[8] G.-S. Xia, J. Hu, F. Hu et al., "AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, n.º 7, págs. 3965-3981, 2017. DOI: 10.1109/TGRS.2017.2685945.

[9] A. Shamsoshoara, F. Afghah, A. Razi, L. Zheng, P. Fulé y E. Blasch, *The FLAME dataset: Aerial Imagery Pile burn detection using drones (UAVs)*, 2020. DOI: 10.21227/qad6-r683. dirección: https://dx.doi.org/10.21227/qad6-r683.

[10] V. Stojnić, V. Risojević, M. Muštra et al., "A Method for Detection of Small Moving Objects in UAV Videos," *Remote Sensing*, vol. 13, n.º 4, 2021, ISSN: 2072-4292. DOI: 10.3390/rs13040653. dirección: https://www.mdpi.com/2072-4292/13/4/653.

[11] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri y R. M. Summers, "ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," en *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, págs. 3462-3471. DOI: 10.1109/CVPR.2017.369.

[12] N.-D. Nguyen, T. Do, T. D. Ngo, D.-D. Le y C. F. Valenti, "An Evaluation of Deep Learning Methods for Small Object Detection," *JECE*, vol. 2020, ene. de 2020, ISSN: 2090-0147. DOI: 10.1155/2020/3189691. dirección: https://doi.org/10.1155/2020/3189691.

[13] J. Zhou, C.-M. Vong, Q. Liu y Z. Wang, "Scale adaptive image cropping for UAV object detection," *Neurocomputing*, vol. 366, págs. 305-313, nov. de 2019. DOI: 10.1016/j.neucom.2019.07.073. dirección: https://doi.org/10.1016/j.neucom.2019.07.073.

[14] L. Liebel y M. Körner, "SINGLE-IMAGE SUPER RESOLUTION FOR MULTISPECTRAL REMOTE SENSING DATA USING CONVOLUTIONAL NEURAL NETWORKS," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B3, págs. 883-890, 2016. DOI: 10.5194/isprs-archives-XLI-B3-883-2016. dirección: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLI-B3/883/2016/.

[15] M. Haris, G. Shakhnarovich y N. Ukita, "Task-driven super resolution: Object detection in low-resolution images," en *International Conference on Neural Information Processing*, Springer, 2021, págs. 387-395.

[16] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu y J. Jiang, "Edge-Enhanced GAN for Remote Sensing Image Superresolution," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, n.º 8, págs. 5799-5812, 2019. DOI: 10.1109/TGRS.2019.2902431.

[17] I. García-Aguilar, R. M. Luque-Baena y E. López-Rubio, "Improved detection of small objects in road network sequences using scpCNN/scp and super resolution," *Expert Systems*, vol. 39, n.º 2, dic. de 2021. DOI: 10.1111/exsy.12930. dirección: https://doi.org/10.1111/exsy.12930.

[18] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec y K. Cho, *Augmentation for small object detection*, 2019. DOI: 10.48550/ARXIV.1902.07296. dirección: https://arxiv.org/abs/1902.07296.

[19] H. Tayara, K. G. Soo y K. to Chong, "Vehicle Detection and Counting in High-Resolution Aerial Images Using Convolutional Regression Neural Network," *IEEE Access*, vol. 6, págs. 2220-2230, 2018.

[20] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi y P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Scientific Reports*, vol. 10, n.º 1, mar. de 2020. DOI: 10.1038/s41598-020-61808-3. dirección: https://doi.org/10.1038/s41598-020-61808-3.

[21] R. Li, R. Wang, J. Zhang et al., "An Effective Data Augmentation Strategy for CNN-Based Pest Localization and Recognition in the Field," *IEEE Access*, vol. 7, págs. 160 274-160 283, 2019. DOI: 10.1109/ACCESS.2019.2949852.

[22] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 779-788. DOI: 10.1109/CVPR.2016.91.

[23] C.-Y. Wang, A. Bochkovskiy y H.-Y. M. Liao, *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 2022. DOI: 10.48550/ARXIV.2207.02696. dirección: https://arxiv.org/abs/2207.02696.

[24] X. Wang, K. Yu, S. Wu et al., "Esrgan: Enhanced super-resolution generative adversarial networks," en *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018, págs. 0–0.

[25] J. Rabbi, N. Ray, M. Schubert, S. Chowdhury y D. Chao, "Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network," *Remote Sensing*, vol. 12, n.º 9, 2020, ISSN: 2072-4292. DOI: 10.3390/rs12091432. dirección: https://www.mdpi.com/2072-4292/12/9/1432.

[26] O. C. Koyun, R. K. Keser, İ. B. Akkaya y B. U. Töreyin, "Focus-and-Detect: A small object detection framework for aerial images," *Signal Processing: Image Communication*, vol. 104, pág. 116 675, 2022, ISSN: 0923-5965. DOI: https://doi.org/10.1016/

j.image.2022.116675. dirección: https://www.sciencedirect.com/science/article/pii/S0923596522000273.

[27] M. Tan, R. Pang y Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," en *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, págs. 10 778-10 787. DOI: 10.1109/CVPR42600.2020.01079.

[28] M. Tan y Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," en *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri y R. Salakhutdinov, eds., ép. Proceedings of Machine Learning Research, vol. 97, PMLR, sep. de 2019, págs. 6105-6114. dirección: https://proceedings.mlr.press/v97/tan19a.html.

[29] X. Zhou, D. Wang y P. Krähenbühl, *Objects as Points*, 2019. DOI: 10.48550/ARXIV.1904.07850. dirección: https://arxiv.org/abs/1904.07850.

[30] R. Girshick, J. Donahue, T. Darrell y J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," en *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, págs. 580-587. DOI: 10.1109/CVPR.2014.81.

[31] R. Girshick, "Fast R-CNN," en *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, págs. 1440-1448. DOI: 10.1109/ICCV.2015.169.

[32] S. Ren, K. He, R. Girshick y J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," en *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama y R. Garnett, eds., vol. 28, Curran Associates, Inc., 2015. dirección: https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[33] W. Liu, D. Anguelov, D. Erhan et al., "SSD: Single Shot MultiBox Detector," en *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe y M. Welling, eds., Cham: Springer International Publishing, 2016, págs. 21-37.

[34] B. Lim, S. Son, H. Kim, S. Nah y K. M. Lee, *Enhanced Deep Residual Networks for Single Image Super-Resolution*, 2017. DOI: 10.48550/ARXIV.1707.02921. dirección: https://arxiv.org/abs/1707.02921.

[35] K. He, X. Zhang, S. Ren y J. Sun, "Deep Residual Learning for Image Recognition," en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 770-778. DOI: 10.1109/CVPR.2016.90.

[36] W. Shi, J. Caballero, F. Huszár et al., "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," en *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 1874-1883. DOI: 10.1109/CVPR.2016.207.

[37] C. Dong, C. C. Loy y X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network," en *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe y M. Welling, eds., Cham: Springer International Publishing, 2016, págs. 391-407, ISBN: 978-3-319-46475-6.

[38] W.-S. Lai, J.-B. Huang, N. Ahuja y M.-H. Yang, "Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, n.º 11, págs. 2599-2613, 2019. DOI: 10.1109/TPAMI.2018.2865304.

[39] E. L. Denton, S. Chintala, a. szlam arthur y R. Fergus, "Deep Generative Image Models using a ☒Laplacian Pyramid of Adversarial Networks," en *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama y R. Garnett, eds., vol. 28, Curran Associates, Inc., 2015. dirección: https://proceedings.neurips.cc/paper/2015/file/aa169b49b583a2b5af89203c2b78c67c-Paper.pdf.

[40] C. Chaofeng, X. Shi, Y. Qin, X. Li, X. H. T. Yang y S. Guo., *Real-world blind super-resolution via feature matching with implicit high-resolution priors -ORCA — orca.cardiff.ac.uk*, https://orca.cardiff.ac.uk/id/eprint/151095, 2022.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza et al., "Generative Adversarial Networks," *Commun. ACM*, vol. 63, n.º 11, págs. 139-144, oct. de 2020, ISSN: 0001-0782. DOI: 10.1145/3422622. dirección: https://doi.org/10.1145/3422622.

[42] C. Ledig, L. Theis, F. Huszár et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," en *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, págs. 105-114. DOI: 10.1109/CVPR.2017.19.

[43] K. Zhang, J. Liang, L. Van Gool y R. Timofte, "Designing a Practical Degradation Model for Deep Blind Image Super-Resolution," en *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, oct. de 2021, págs. 4791-4800.

[44] Z. Wang, A. Bovik, H. Sheikh y E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, n.º 4, págs. 600-612, 2004. DOI: 10.1109/TIP.2003.819861.

[45] L. Zhang, L. Zhang, X. Mou y D. Zhang, "FSIM: A Feature Similarity Index for Image Quality Assessment," *IEEE Transactions on Image Processing*, vol. 20, n.º 8, págs. 2378-2386, 2011. DOI: 10.1109/TIP.2011.2109730.

[46] A. F. Hassan, Z. M. Hussain y D. Cai-lin, "An Information-Theoretic Measure for Face Recognition: Comparison with Structural Similarity," *International Journal of Advanced Research in Artificial Intelligence*, vol. 3, n.º 11, 2014. DOI: 10.14569/IJARAI.2014.031102. dirección: http://dx.doi.org/10.14569/IJARAI.2014.031102.

[47] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, n.º 6, págs. 679-698, 1986. DOI: 10.1109/TPAMI.1986.4767851.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA