Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Detección de objetos abandonados en secuencias de imágenes

Autor: Pablo Calvo del Castillo

Tutora: Cristina Losada Gutiérrez

2022

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

Detección de objetos abandonados en secuencias de imágenes

Autor: Pablo Calvo del Castillo

Tutora: Cristina Losada Gutiérrez

Tribunal:

Presidente: Ana Isabel de Andrés Rubio

Vocal 1º: Raúl Mateos Gil

Vocal 2º: Cristina Losada Gutiérrez

Fecha de depósito: 4 de julio de 2022

A todas las personas que han colaborado en el trabajo

"El modo de dar una vez en el clavo es dar cien veces en la herradura."
Miguel de Unamuno

Agradecimientos

Las raíces bajo la tierra no piden recompensa por hacer
que las ramas den frutos.

Rabindranath Tagore

Este trabajo representa el final de una etapa, y después de tantas horas, solo queda agradecer a todas las personas que han participado de forma directa o indirecta.

En primer lugar, dar las gracias a los profesores, en especial a mi tutora Cristina Losada, por estar siempre disponible para solucionar los problemas que surgían. Agradecer la oportunidad ofrecida de conocer el mundo de la investigación y los conocimientos obtenidos.

A mis amigos de la carrera, por acompañarme en esta importante etapa de la vida. Daros las gracias por no ser solo compañeros de clase y poder contar con vosotros en cualquier momento. Especial mención a Juanjo, por todos nuestros debates para al final resolver casi cualquier problema que se pudiera plantear, gracias por tu ayuda. Destacar también a Jesús, Raquel y Guille, por estar en cualquier momento.

Por último, a mi familia, mis padres y mi hermano, por el apoyo brindado todo este tiempo. Sin vosotros no hubiera sido posible. Agradecer a María, por aguantar todas las charlas de los problemas que surgían.

Resumen

El objetivo de este Trabajo Fin de Grado es el desarrollo de un sistema de detección de objetos abandonados en secuencias de vídeo. Se han estudiado los métodos existentes, eligiendo dos para su implementación y evaluación. El primero en Matlab, a partir del análisis de formas para la detección de objetos, y el segundo basado en la red YOLOv3 en Python, mejorando los resultados. Tras la detección, se incluye una etapa de seguimiento y se aplica un conjunto de reglas para determinar un abandono. Se han evaluado ambos métodos en la colección de PETS2006, permitiendo comparar y validar las propuestas.

Palabras clave: Detección de abandonos, Matlab R2020b, YOLOv3, Filtro de Kalman, seguridad videovigilada.

Abstract

The aim of this Final Degree Thesis is the development of a system capable of detecting abandoned objects from video sequences. A theoretical analysis of the existing methods has been carried out, choosing two proposals for implementation. The first one in Matlab, base on the analysis of shapes for object detection, and the second one using YOLOv3 network in Python, what allows improving the results. After detection, a tracking stage is included and a set of rules is applied to determine an abandoned object. Both methods have been evaluated in the PETS2006 dataset, what has allowed comparing and validating both proposals.

Keywords: Abandoned objects, Matlab R2020b, YOLOv3, Kalman Filter, Video Surveillance Security.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

BGS	*Background Subtraction*.
BN	Batch Normalization.
CNN	Redes Neuronales Convolucionales.
COCO	Common Objects In Context.
FN	Falso Negativo.
FP	Falso Positivo.
GEINTRA	Grupo de Ingeniería Electrónica aplicada a Espacios Inteligentes y Transporte.
MLP	Perceptrón multicapa.
OA	Objetos Abandonados.
RBF	Redes de base radial.
ReLU	Rectified Linear Unit.
RNN	Redes Neuronales Recurrentes.
SSD	*Single Shot Detector*.
TFG	Trabajo fin de grado.
TN	Verdadero Negativo.
TP	Verdadero Positivo.
YOLO	You Only Look Once.

Capítulo 1

Introducción

No esperes a que las condiciones sean perfectas para empezar, el empezar hace las condiciones perfectas.

Allan Cohen

1.1 Presentación

En la última década, las mejoras en las tecnologías de la información y las comunicaciones, han hecho que la tecnología se haya incorporado en diferentes ámbitos de la vida diaria. En este contexto, la visión artificial es una de las tecnologías actualmente en auge dentro de la investigación, debido a su utilidad en múltiples entornos y aplicaciones, ya que facilita y agiliza la realización de diferentes procesos. Entre todos los ámbitos en los que se aplica, destaca la detección y seguimiento de personas [9–11], la monitorización de la salud [12,13] o los sistemas centrados en su seguridad; los cuales son capaces de detectar situaciones anómalas o de riesgo automáticamente [14–17].

En el contexto del reconocimiento de eventos anómalos a partir de imágenes o vídeos, una de las tareas de interés es la detección de objetos abandonados (especialmente en entornos públicos [18–20]), en la que se centra este Trabajo fin de grado (TFG). Así, el objetivo general del trabajo consta del estudio, diseño, implementación y evaluación de un sistema para la detección de objetos abandonados en secuencias de imágenes.

Para la resolución de este problema, en el presente TFG, se han propuesto, implementado y evaluado dos alternativas para la detección de objetos abandonados; la primera de ellas se basa en métodos clásicos, mientras que la segunda emplea redes neuronales. A continuación, se describen brevemente las soluciones propuestas, que se presentan con mayor detalle en el capítulo 3. Además, en el capítulo 4, se comparan los resultados obtenidos con ambas soluciones.

Cabe destacar que este TFG se ha desarrollado en el marco del proyecto de investigación ARGOS+ (PIUAH21/IA-016), financiado por la Universidad de Alcalá, tomando como punto de partida los trabajos de investigación realizados con anterioridad en el grupo de investigación Grupo de Ingeniería Electrónica aplicada a Espacios Inteligentes y Transporte (GEINTRA) [21], partiendo del material existente para mejorarlo y optimizarlo para posteriormente analizar y comparar diferentes formas de abordar el problema bajo análisis.

1.2 Soluciones propuestas

Como se ha comentado previamente, en este TFG, se han propuesto, implementado y evaluado dos propuestas diferentes para la detección de objetos abandonados en secuencia de imágenes. En este apartado se exponen las ideas principales que se han utilizado en cada uno de los métodos, las cuáles en conjunto, hacen posible lograr el objetivo que se busca. Se acompaña de un esquema general del funcionamiento (figura 1.1) donde aparecen los principales pasos que se deben realizar.

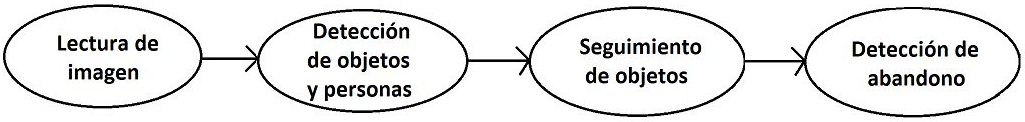


Figura 1.1: Esquema general de funcionamiento de los sistemas para detección de objetos abandonados abordados en este TFG.

A continuación se detalla cómo se abordan cada una de las etapas incluidas en el esquema de la figura 1.1 en las dos alternativas propuestas en este TFG.

La primera propuesta emplea técnicas clásicas de eliminación de fondo y segmentación de objetos, y se desarrolló a partir de la propuesta disponible en un repositorio de GitHub [22]. Así, en el primero de los métodos, se emplea Matlab como plataforma de programación. Tras la segmentación de objetos, realizando un análisis y seguimiento de los mismos, se decide en qué momento ha sido abandonado un objeto en función de la evolución de su posición con el tiempo.

En el primer método, se obtienen los objetos que se desean analizar realizando un procesamiento de la imagen. Para ello, una vez introducido el vídeo o la secuencia de imágenes de entrada, se obtiene el fondo de la escena. Este paso resulta de gran importancia, ya que la precisión de los resultados depende directamente de la calidad del fondo obtenido. En el caso de la solución empleada como punto de partida [22], se realiza la extracción de fondo estático, en el cuál se elige como fondo la primera imagen del vídeo, lo que produce errores importantes en caso de que existan objetos y/o personas en esa primera imagen. De esta forma, pocas veces se consigue obtener el fondo que se desea para las imágenes introducidas. Para mejorar los resultados, en el sistema desarrollado en este TFG, se ha incluido el análisis de las primeras imágenes, lo que permite obtener el mejor fondo de los primeros segundos del vídeo, evaluando así la cantidad de ruido (objetos que no forman parte del fondo) que presenta cada imagen.

Independientemente del fondo almacenado se realiza una resta de fondo, seguida de un filtrado y operaciones morfológicas sobre la imagen para detectar los objetos y personas que se encuentran en el primer plano. Con un análisis de formas se consigue identificar la situación y características de todas las personas y objetos de la imagen, almacenando su información temporal a lo largo de diferentes imágenes de la secuencia.

Posteriormente, a partir de la información extraída de la imagen, se establecen las condiciones de abandono que provocan la activación de un aviso en caso de que algún objeto las cumpla. En el caso del método analizado en este TFG, esas condiciones están relacionadas con el número de imágenes consecutivas en las que el objeto permanece en el mismo lugar.

Como segunda alternativa se utilizan redes neuronales para lograr la detección de los objetos, dado que en los últimos años han demostrado su eficacia en tareas de detección y segmentación de objetos y personas.

Se crean sistemas para multitud de situaciones, utilizados desde la lectura de códigos de barras [23], sistemas para facilitar las labores agrícolas y ganaderas [24] con sistemas de localización y conteo de animales, o incluso, sistemas empleados en el espacio [25] para la detección de residuos de otras misiones.

Se elige la red de YOLO [26] para la etapa de detección de objetos y personas, por lo que se decide programar en lenguaje Python, debido a las facilidades que presenta la red escogida. Posteriormente, se incluye una etapa de seguimiento, a partir de cuya salida se realiza la detección de objetos abandonados, siendo el esquema general similar al de la primera alternativa. En la figura 1.2 se aprecian las principales acciones realizadas.

Figura 1.2: Esquema general de funcionamiento de la propuesta para detección de objetos abandonados basada en redes neuronales.

Una vez introducidas las imágenes de entrada en el programa, se carga la red de YOLO, e introduciendo individualmente cada una de las imágenes, se extraen los objetos que la componen. Una de las ventajas de este tipo de redes neuronales es que no solamente realizan la detección, sino que también permiten clasificar los elementos detectados, discriminando entre personas y objetos de interés (mochilas, maletas, etc.).

A continuación se realiza el seguimiento y asociación de todas las personas detectadas utilizando un banco de filtros de Kalman, que es capaz de predecir la posición de la persona en caso de no ser detectada en la imagen. Debido a que puede haber varias personas y/o objetos de interés en la imagen, es necesario realizar un proceso de asociación entre las predicciones de los filtros de Kalman y las detecciones realizadas por la red neuronal. Con el seguimiento temporal de objetos y personas, se relaciona cada objeto con una persona, suponiendo que permanecerán juntos mientras sean detectados.

El último paso consiste en evaluar todas las detecciones almacenadas para poder concluir si alguno de los objetos ha sido abandonado. Este análisis se centra en la distancia a la que se sitúa la persona asociada al objeto, además de la posición del objeto en el momento en el que la persona se aleja. Una vez se define el abandono, se debe comprobar su situación para asegurar que sigue estando abandonado.

1.3 Organización del documento

Este documento recoge la memoria del TFG. Tras esta introducción, el documento se organiza como se detalla a continuación:

En el capítulo 2 se estudian los conceptos teóricos necesarios para llevar a cabo el trabajo. En primer lugar, resolviendo la problemática mediante métodos clásicos, se analiza la segmentación objeto-fondo, así como las utilidades que presenta. Se continúa con el estudio de las Redes Neuronales Convolucionales (CNN), prestando especial atención a la red empleada (YOLO) para poder llevar a cabo el segundo método. Además, se analiza el funcionamiento del filtro de Kalman y sus principios básicos. Se finaliza este apartado con la validación de objetos abandonados, para ver diferentes maneras y condiciones para poder detectar cuando se realiza el abandono de un objeto.

A continuación, en el capítulo 3, se detalla el desarrollo de cada uno de los algoritmos planteados, y siguiendo la estructura principal explicada anteriormente, se incluyen las explicaciones detalladas de cada una de las etapas necesarias, detallando los procedimientos llevados a cabo para conseguir el objetivo final del TFG. También se explican los problemas que han ido surgiendo en la realización y programación; así como las soluciones implementadas para lograr un mejor resultado.

Posteriormente, el capítulo 4 recoge los principales resultados experimentales obtenidos tras realizar pruebas sobre diferentes secuencias de imágenes que sirven para cuantificar con estadísticas el funcionamiento logrado. Se realiza una comparación de ambos métodos a partir de secuencias de imágenes de un *dataset* etiquetado, incluyendo una explicación para los resultados extraídos.

En el capítulo de conclusiones se añaden las sensaciones extraídas al finalizar el trabajo, observando la evolución de las propuestas y las debilidades que presentan, así como las posibles mejorías que se podrían seguir añadiendo en proyectos futuros.

Para finalizar se realiza un presupuesto real que supondría la implantación de un sistema en una ubicación real; añadiendo los costes de: materiales, los asociados al diseño, montaje y mano de obra.

Capítulo 2

Estudio teórico

La cultura se adquiere leyendo libros; pero el conocimiento del mundo, que es mucho más necesario, sólo se alcanza leyendo a los hombres y estudiando las diversas ediciones que de ellos existen.

Lord Chesterfield

2.1 Introducción

En este capítulo se realiza un análisis de los conceptos teóricos necesarios, para lograr el correcto desarrollo del presente trabajo, que permitan abordar el problema de la detección de objetos abandonados de la mejor forma posible.

El primer problema que se encuentra consiste en la detección de objetos presentes en la imagen, ya que una vez detectados, se puede utilizar su información para extraer conclusiones. Para poder comenzar a abordar el problema se recogen diferentes alternativas capaces de detectar los objetos de las imágenes.

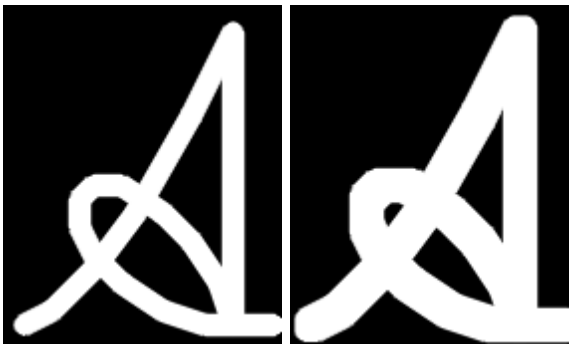
En primer lugar, se describen algunos de los métodos de detección basadas en técnicas clásicas de visión artificial, para a continuación presentar alternativas más modernas basadas en redes neuronales.

2.2 Sistemas clásicos de detección

Como se ha comentado, en primer lugar se estudian los **sistemas clásicos de detección o sistemas de segmentación**. Tal y como se explica en la publicación "Técnicas y algoritmos básicos de visión artificial" [27], la segmentación consiste en dividir una imagen digital en diferentes zonas individuales, es decir, diferenciar el fondo y los diferentes objetos de la imagen. Este proceso varía en su complejidad en función de la imagen introducida, pero una vez realizada la segmentación, se deben conocer todas las características propias de cada objeto.

Para poder realizar la extracción de objetos es habitual ejecutar una serie de transformaciones morfológicas (como son la dilatación, apertura, cierre ...); estas operaciones realizan una modificación en la estructura de los objetos, lo que permite separar unos objetos de otros, obtener los contornos que los delimitan, reconstruir elementos distorsionados, u obtener formas simples a partir de otras más complejas. A continuación se exponen las principales modificaciones más empleadas en el tratamiento de imágenes:

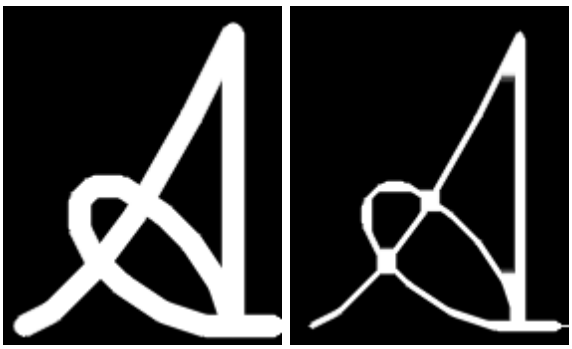
- **Dilatación binaria.** Se emplea para reducir los agujeros interiores y engrosar los bordes del objeto, como se puede ver en el ejemplo de la figura 2.1. También sirve para unir líneas discontinuas que puedan aparecer por algún proceso de filtrado.



(a) Imagen binaria inicial. (b) Dilatación binaria.

Figura 2.1: Ejemplo de la operación de dilatación binaria para engrosar las formas [1].

- **Erosión binaria.** De manera contraria a la dilatación, se utiliza para reducir los contornos, consiguiendo separar objetos que se encontraban unidos ligeramente, como se observa en el ejemplo de la figura 2.2.



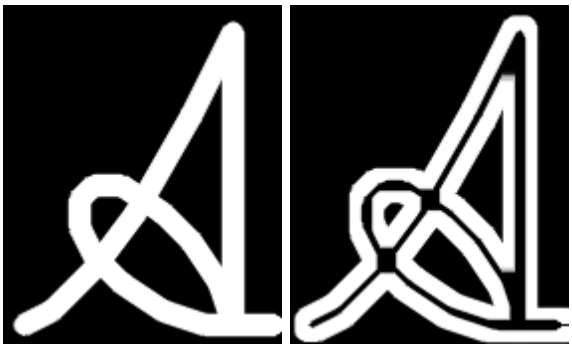
(a) Imagen binaria inicial. (b) Erosión binaria

Figura 2.2: Ejemplo de la erosión binaria para apreciar el efecto que se produce [1].

- **Apertura.** Consiste en aplicar una erosión seguida de una dilatación (figura 2.1). En el primer proceso se logra delimitar los contornos de cada forma, eliminando el ruido existente en el imagen binaria. El segundo paso consigue la dilatación de cada forma para ampliar sus bordes.

Figura 2.3: Ejemplo de la operación de apertura a partir de imagen binaria para la eliminación de pequeños agujeros [1].

- **Cierre.** En este caso se aplica en primer lugar una dilatación seguida por la erosión. De nuevo se consigue eliminar el ruido (figura 2.4).



(a) Imagen binaria inicial. (b) Cierre.

Figura 2.4: Ejemplo de la operación de cierre para la obtención de los contornos de la forma [1].

Además, existen numerosas operaciones menos comunes que pueden ser útiles dependiendo de la aplicación. Lo más común es realizar dichas modificaciones una vez se convierte la imagen principal a binaria, ya que es más sencillo y mejora los resultados obtenidos.

A la hora de realizar la segmentación, existen diferentes técnicas para el proceso, y en la actualidad se siguen investigando nuevos métodos para abordar problemas más complejos que no se puedan resolver con las alternativas actuales. Algunas de las técnicas más comunes se presentan a continuación, analizando su funcionamiento y las ventajas que presentan.

2.2.1 Segmentación basada en la umbralización

Esta técnica de detección se basa en agrupar los píxeles según los niveles de intensidad luminosa. Para ello se debe analizar una imagen en escala de grises y se recoge el número de píxeles que tienen cada tonalidad de gris.

En la figura 2.5, se observa una imagen extraída de una página de fotografías [28], y su correspondiente histograma. Se observa una concentración en los tonos claros, ya que la imagen original no presenta mucho contraste; si la imagen tuviera diferentes objetos representados con distintas tonalidades de grises, se reflejaría en el histograma como un montículo por cada objeto, como puede apreciarse en la figura 2.6, donde aparecen varias formas con distintas tonalidades de grises.

Figura 2.5: Ejemplo de imagen en escala de grises con su correspondiente histograma.

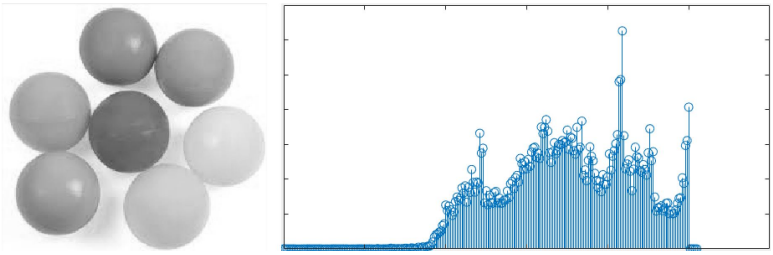


Figura 2.6: Ejemplo de imagen con varios objetos en escala de grises con su correspondiente histograma.

Una vez conocido el histograma, el problema de la técnica de umbralización consiste en encontrar el valor del umbral que sea capaz de distinguir entre los píxeles que forman parte del fondo y los que corresponden a objetos. El inconveniente que presenta es la gran dependencia que tiene con las condiciones de iluminación. Si la iluminación es homogénea y constante puede determinarse un umbral adecuado mediante la experimentación. Algunas de las formas para obtener el umbral que consiga separar el fondo de los objetos son las siguientes:

- Método P-cuantil. Es un método bastante limitado, y consiste en utilizar el tamaño o área del objeto que se quiere extraer para asignar el mismo porcentaje de píxeles al objeto. Es utilizado para el reconocimiento de caracteres.
- Búsqueda de mínimos. Como se ha visto anteriormente, cada uno de los objetos presentes en la imagen se representan como un montículo en el histograma, por lo que identificando los mínimos es posible separar cada uno de los objetos fijando el umbral en el propio mínimo (figura 2.7).

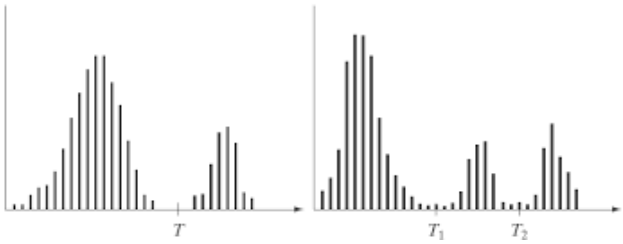


Figura 2.7: Ejemplos de umbralización basado en la búsqueda de mínimos con 2 y 3 objetos respectivamente.

- Técnicas de reconocimiento de formas. Se aproximan cada uno de los lóbulos producidos por los objetos en el histograma a una campana de Gauss, ajustando a cada objeto una función de Gauss. Resulta más práctico al dividir la imagen en diferentes zonas, calculando el histograma de cada una de ellas (figura 2.8).

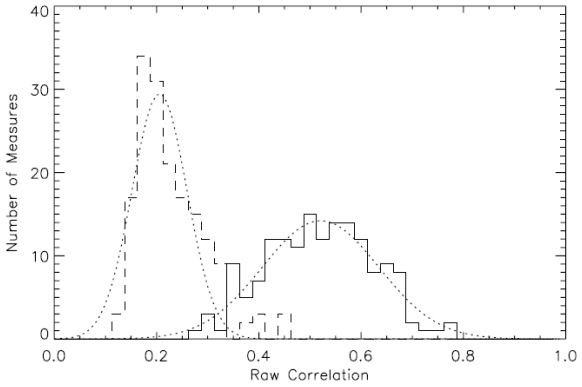


Figura 2.8: Ejemplo de aproximación de un histograma a la función de Gauss. Cada aproximación posible corresponde a una forma diferente.

2.2.2 Segmentación basada en bordes

En este caso se utilizan detectores para obtener los bordes de los diferentes objetos, para a continuación crear cadenas de bordes que tendrán una correspondencia con estos [2]. A continuación se indican los métodos más simples para la obtención de objetos a partir de los bordes que se explican en [27]:

- **Extracción de las fronteras**. Las fronteras son bordes unidos que caracterizan la forma del objeto. Un algoritmo bastante utilizado es **el método de la tortuga**, que consiste en evaluar un píxel. Si está en un punto negro, se deben girar 90^o hacia la izquierda y avanzar, realizando el giro hacia la izquierda en caso de que sea blanco. Almacenando todos los píxeles negros, se delimitará una forma cuando se vuelva al primer píxel negro encontrado.
- **Trasformada de Hough**. Consiste en crear un vector por cada uno de los contornos detectados. Presenta algún inconveniente como la lentitud del cálculo.
- **Transformada del rellenado creciente**. Consiste en asociar a cada píxel un número con la distancia al borde más cercano, lo cual permite distinguir fácilmente cada uno de los objetos.

A continuación, se aplica la segmentación en cuestión sobre la figura 2.9a; una vez realizado el proceso se obtiene la figura 2.9b, en esta última se aprecia el contorno y la forma del objeto a partir de los bordes obtenidos.

(a) Imagen de entrada. (b) Resultado de la segmentación.

Figura 2.9: Ejemplo de imagen de entrada e imagen segmentada , donde se delimita la casa a partir de sus bordes [2].

2.2.3 Segmentación basada en la eliminación de fondo

De todas las técnicas que se han visto anteriormente, éste es el método que se escoge para la realización del trabajo mediante métodos clásicos. Esta segmentación es de tipo *Background Subtraction* (BGS); al igual que cualquier técnica de segmentación, se utiliza para poder separar entre el fondo y el frente de la imagen.

Para que el método funcione correctamente se debe generar un modelo del fondo que se utiliza posteriormente para obtener los objetos. El proceso de segmentación se divide en 2 pasos, el primero de ellos consiste en la obtención del modelo de fondo, continuando con la extracción de los objetos una vez se tiene el fondo. En la figura 2.10 se visualiza un ejemplo del alcance que se consigue al aplicar esta segmentación.

Figura 2.10: Ejemplo de segmentación de fondo de una imagen para la obtención de personas en primer plano [3].

1. **Selección y almacenamiento del fondo**

   Como se ha comentado, en primer lugar se debe determinar el fondo de las imágenes que se quiere analizar. Existen diversas técnicas para su obtención, las cuales se pueden clasificar en fondo estático (cuando se identifica una única vez para toda la secuencia de imágenes) o fondo dinámico (cuando el fondo puede ir evolucionando con el tiempo). Los principales métodos de extracción de fondo estático son las siguientes:

   - Una de las técnicas más empleadas para la obtención de fondo se basa en asociar la primera imagen de una colección como fondo, asumiendo que todo lo que aparece en la imagen formará parte del citado fondo.
   - En alguna de las colecciones se aporta una imagen aislada que contiene el fondo de todas las imágenes, lo que facilita el trabajo y mejora los resultados. A la hora de la implementación de dichos sistemas en entornos reales se captura un instante en el cual solo aparezca en la imagen el fondo de la escena (sin ningún objeto ni persona), facilitando el trabajo de su obtención. Este proceso no resulta válido cuando varían las imágenes en su perspectiva u orientación, tampoco son de utilidad cuando el fondo de la imagen pueda sufrir variaciones.

   La obtención de este fondo es crucial para la segmentación, ya que de ello depende la correcta extracción de los objetos. Por este motivo se crean nuevos métodos de obtención de fondo más complejos que consiguen una mejoría en los resultados, dando lugar a la extracción dinámica de fondo, para la cuál se necesita más de una imagen, como se explica a continuación:

- Una solución consiste en dividir la imagen en diversas secciones y evaluar en varias imágenes si cada una de las secciones puede ser considerada como fondo. Se logra mejorar los resultados, ya que, si por ejemplo se divide una imagen en 30 secciones, en la primera imagen puede aparecer una persona en las secciones 12 y 22, por lo que utilizar la imagen completa como fondo no sería correcto. En este caso se almacenan todas las secciones excepto donde aparezcan objetos y cuando la persona se desplace a una nueva ubicación en las siguientes imágenes, se almacenan las regiones restantes donde solo aparezca fondo.
- Otra solución similar consiste en evaluar el ruido (objetos no pertenecientes al fondo) que existe en cada imagen, almacenando como fondo la imagen que menos ruido presente entre todas las evaluadas. Se obtienen mejores resultados que eligiendo únicamente la primera imagen, ya que permite comparar y elegir la mejor opción.

2. **Separación de fondo y objetos**

   Una vez se logra extraer un fondo válido de la secuencia de imágenes, se procede a obtener los objetos en primer plano de cada imagen por separado. Para ello se realizan operaciones con todos los píxeles de la imagen, realizando la diferencia entre el píxel del fondo y el de la imagen. Aparecen 2 situaciones diferentes en la sustracción de fondo. En primer lugar el píxel evaluado forma parte del fondo, por lo que la diferencia entre ambos píxeles resulta 0 o un valor muy pequeño, a la hora de representar el resultado se obtiene el color negro. Por otro lado el píxel seleccionado puede pertenecer a un objeto en primer plano que se desea extraer, por lo que la diferencia sería no nula, resultando un color diferente al negro. De esta forma es posible obtener una imagen en la que aparezcan los objetos en primer plano en un color y el resto de la imagen (correspondiente con el fondo) en negro.

   Es bastante común representar el resultado de la operación en formato binario, de esta forma se consigue separar entre el fondo (representado con un 0 ,en negro) y los objetos (respresentados con un 1, en blanco). El resultado es visualmente más cómodo además de facilitar las modificaciones que se deben realizar posteriormente (por ejemplo el cierre).

2.3 Detección de objetos y personas mediante redes neuronales

Continuando con el estudio de los métodos empleados para la detección de objetos abandonados, se aborda su resolución utilizando redes neuronales para realizar la detección de objetos en la imagen. A priori, esta opción soluciona algunos problemas que surgen a partir de la realización mediante métodos clásicos, por ejemplo, no se tiene que realizar la extracción de fondo, por lo que los resultados no dependerán de ello. En este caso, los resultados dependerán del entrenamiento y configuración de la red neuronal.

Las redes neuronales funcionan de forma similar a un cerebro humano, en la cual un conjunto de neuronas se encuentran conectadas entre sí y trabajan simultáneamente sin una tarea propia para cada neurona. Las conexiones existentes entre ellas se ajustan a medida que se entrena la red. Este entrenamiento hace que la red aprenda progresivamente la información utilizada en el entrenamiento. Para dicho entrenamiento, se producen numerosas iteraciones que finalizan cuando se consigue clasificar la totalidad de la información, o bien cuando no aumenta el número de aciertos.

De manera general, todas las neuronas que forman la red están organizadas en diferentes capas; estas capas están organizadas en una capa de entrada donde se presentan los datos de entrada, una o varias capas ocultas y la capa de salida, para representar el campo de destino. Los valores se propagan desde una neurona hasta otra de la siguiente capa, y cuando llega a la última capa se genera el resultado.

Existen diferentes tipos de redes neuronales, utilizadas en función de la aplicación que se aplica [29]. Los tipos más empleados son los siguientes:

- **Redes monocapa o perceptrón simple.** Es la red neuronal más simple. La capa de entrada proyecta las entradas hacia una capa de neuronas donde se realizan las operaciones de cálculo.
- **Perceptrón multicapa (MLP).** Funciona de manera similar a las redes monocapa pero entre la capa de entrada y la de salida presentan capas intermedias denominadas capas ocultas. Puede estar total o parcialmente conectada en función del número de conexiones.
- **Redes Neuronales Convolucionales (CNN).** En este tipo de red no se realizan todas las conexiones, solo se conectan con un subgrupo. Se consigue reducir el número de neuronas necesarias y se aumenta la capacidad de computo.
- **Redes Neuronales Recurrentes (RNN).** No está formada por capas, las neuronas se conectan arbitrariamente. La red podría tener memoria en caso de que se realicen conexiones cíclicas.
- **Redes de base radial (RBF).** Las salidas de estas redes se calculan en función de la distancia a un punto denominado centro. La salida se obtiene mediante la combinación lineal de las funciones utilizadas por cada neurona individualmente.

A continuación se centra el estudio en las CNN, debido al empleo de estas redes para visión artificial y más concretamente para la detección de objetos. Se emplean este tipo de redes gracias a la velocidad de procesamiento que permite agilizar el análisis de imágenes.

2.3.1 Redes neuronales convolucionales

Este tipo de red neuronal artificial resulta especialmente útil en el procesamiento de imágenes. Su aprendizaje debe estar supervisado y tras este entrenamiento, serán capaces de detectar objetos dentro de una imagen.

Para que la red sea capaz de detectar un objeto se necesitan una gran cantidad de imágenes etiquetadas para que pueda aprender las características básicas de dicho objeto. A mayor número de imágenes aumenta la precisión en la detección, característica que también se incrementa con la variedad de objetos que aparecen en las imágenes.

Analizando la estructura de estas redes se encuentran formadas por numerosas capas, entre las que destacan las capas convolucionales, las cuales les dan el nombre. La información de las imágenes se transmitirá de una capa a otra a partir de las neuronas, llegando a la salida la información extraída.

Capa de entrada

La primera capa por la que se encuentra formada es la capa de entrada, la cuál habitualmente separa la imagen en sus 3 componentes de color RGB. En cada una de las 3 nuevas imágenes se representa cada píxel por un número entre el 0 y el 1, que indica la cantidad de color que contiene el píxel. En la figura 2.11, extraída de [4], se observa un ejemplo simple del funcionamiento de la capa de entrada, se asocia para cada componente de color un valor por cada píxel en función de la cantidad de dicho color que contiene el píxel original.

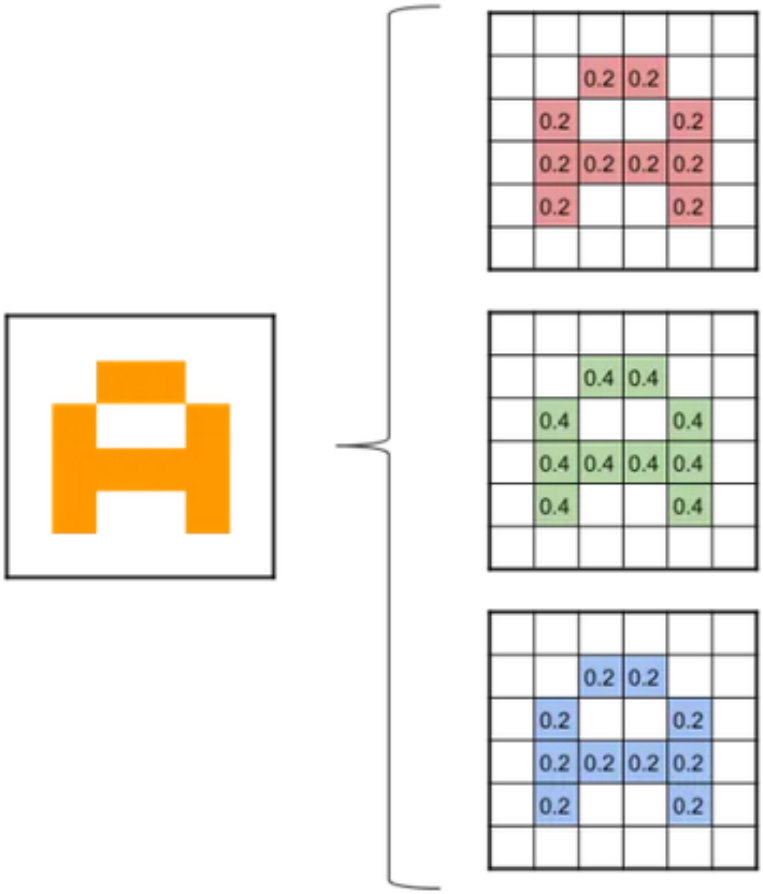


Figura 2.11: Ejemplo del almacenamiento de imagen de entrada en una red neuronal [4].

Capas convolucionales

Las capas que caracterizan este tipo de redes son las capas convolucionales, que funcionan a partir de unos filtros de tamaño reducido formados por 3 dimensiones[1], que se desplazan por la imagen para obtener la salida de la capa. Dichos filtros analizan una zona de la imagen obteniendo en cada posición un valor de salida, para ello se utiliza una matriz denominada *Kernel*, con la cuál se realiza el producto escalar con los valores de los píxeles. A continuación se desplazan los filtros para obtener un nuevo valor para la siguiente zona. Las dimensiones de la imagen formada por las salidas, se logra reducir en tamaño gracias a la agrupación que realizan los filtros de cada capa. También es común incluir más filtros de las dimensiones de entrada, ya que de esta forma se consigue extraer más características, resultando a la salida una imagen de menor tamaño pero con mayor profundidad.

Una vez conocidas las matrices resultantes de las operaciones se aplica la función de activación, la cuál ajusta los valores para que se correspondan con el formato de salida de la capa. Una de las funciones de activación más utilizadas se denomina Rectified Linear Unit (ReLU), que consiste en hacer 0 todas las posiciones que contengan valores negativos y mantener iguales los valores positivos.

En la figura 2.12 se observa un ejemplo de las modificaciones que produciría la capa convolucional desde la entrada de una matriz, que representa una de las imágenes de salida de la capa de entrada, la modificación a partir de la matriz *Kernel* y el resultado final aplicando la función de activación de ReLU.

[1]Las imágenes RGB, presentan 3 dimensiones para cada uno de los colores, por lo que los filtros de las capas deberán tener el mismo tamaño.

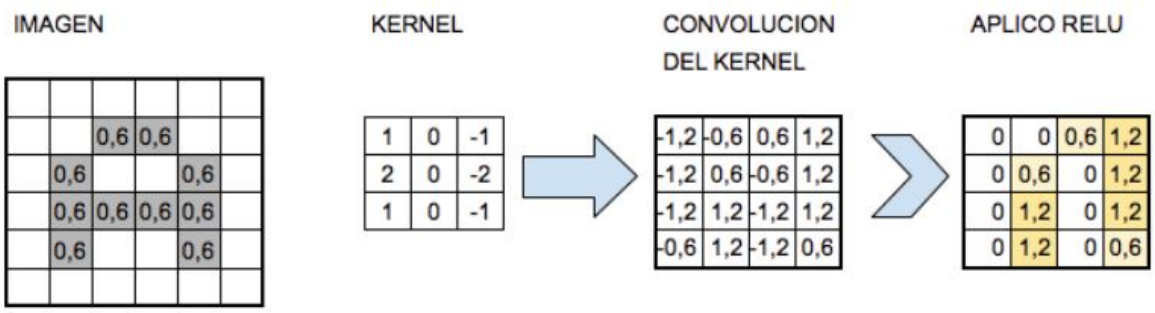


Figura 2.12: Ejemplo de convolución con un *Kernel* y función de activación [4].

La primera capa convolucional sólo es capaz de detectar aspectos básicos de la imagen como pueden ser las líneas o bordes de un objeto, pero a medida que se incluyen más capas, estas son capaces de detectar formas más complejas, haciendo posible la detección de objetos en todo su conjunto.

Capas de reducción o agrupamiento

Estas capas son denominadas de reducción o agrupamiento de sus orígenes, *Pooling* o *Subsampling*, y se encuentran presentes en casi todas las redes convolucionales, ya que sirven para reducir el tamaño de los datos. Resultan ser capas más simples que las convolucionales, ya que en estas se realiza la agrupación de datos de las salidas anteriores. Se realizan agrupaciones de sus elementos de entrada, y se genera a su salida un único valor por cada agrupación. Existen 2 estándares comunes de agrupaciones:

1. *Max* o máximo. El resultado total de la agrupación coincide con el valor más alto de todos los anteriores. La técnica de *Max-Pooling* es usada frecuentemente.

2. *Mean* o promedio. En la técnica de *Mean-Pooling*, la salida será el valor medio de todos los valores de la agrupación.

En la figura 2.13 se observa un ejemplo empleando la técnica de *Max-Pooling* a la salida del ejemplo de la capa convolucional representado en la figura 2.12.

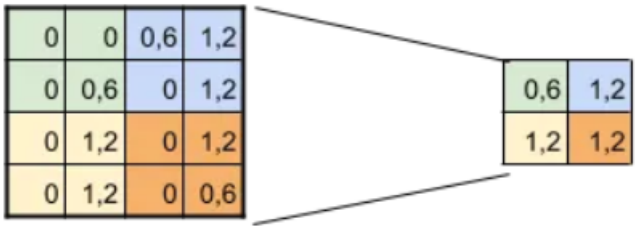


Figura 2.13: Ejemplo de Muestreo mediante Max-pooling de 2x2 reduciendo el tamaño a la mitad [4].

Capas *fully-connected*

Esta capa funciona conectando todas las neuronas de salida de una capa con todas las de la capa siguiente; es posible utilizarla después de cualquier capa convolucional sin necesidad de ser la última, pero esta práctica no se suele realizar en redes convolucionales ya que conlleva numerosas conexiones y neuronas, y al utilizar la red con imágenes aumentarían demasiado los pesos que debería soportar. Por esta razon, en las CNN se suelen situar las capas *fully-connected* una vez se han realizado todas las convoluciones, cuando las dimensiones se han reducido.

Se emplea para clasificar la información recogida por el resto de la red, agrupando los resultados en las clases introducidas en el entrenamiento, además de indicar las probabilidades de pertenecer a cada clase.

Una práctica bastante común es aplicar al final la función denominada *Softmax*, capaz de conectar la capa de salida final con las clases que estamos clasificando. Las salidas presentan un formato denominado *one-hot-encoding*, formado por una matriz que tiene tantas posiciones como clases, y cada posición se asocia a una de las clases.

En la figura 2.14 se observa la estructura general de una red neuronal convolucional, en la cual se representa al inicio la capa de entrada para introducir la imagen que se desea analizar, a continuación se alternan las capas convolucionales con capas de *Pooling*. Las primeras serán capaces de detectar las características más básicas, como pueden ser los bordes, y a medida que se aplican mayor número de capas se logran detectar partes de objetos y finalmente objetos completos. Una vez se suceden todas las capas convolucionales se utilizan las *fully-connected* para agrupar los resultados en las clases predefinidas en la red, de esta manera, a la salida de la red se obtienen los objetos detectados en la imagen de entrada.

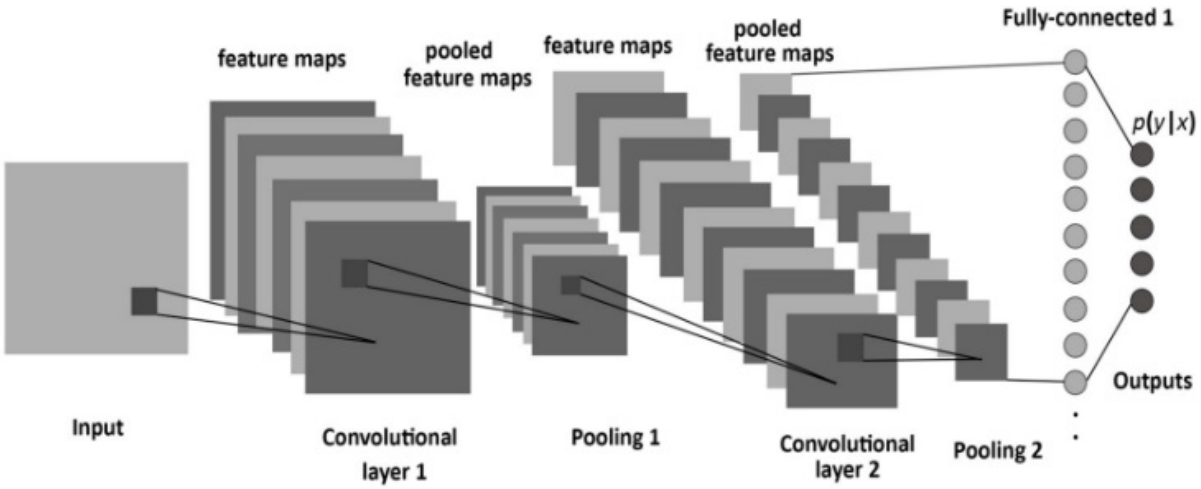


Figura 2.14: Esquema general de una red neuronal convolucional [5].

En la actualidad, existen numerosas redes convolucionales ajustadas para lograr la detección de objetos en una imagen; algunas de las redes más utilizadas son las siguientes:

- *Single Shot Detector* (SSD) [30]. Destaca por ser capaz de detectar desde los más pequeños objetos hasta los más grandes [30].
- RetinaNet [31]. Es una de las redes que obtiene mejores resultados gracias al empleo de una nueva función denominada *Focal Loss*.
- Google Spinnet. Se sale de la estructura piramidal típica de las CNN, alternando diferentes tamaños en las convoluciones.
- Facebook DETR [32]. Resulta una red muy rápida y eficiente enfocando el problema de la detección de forma directa.
- YOLO [7]. Se logran obtener velocidades nunca alcanzadas realizando una única pasada por cada convolución. Se decide la implementación de YOLO para facilitar el análisis de vídeos en tiempo real.

A continuación, se detallan las principales características de la red neuronal elegida en este TFG: YOLO.

2.3.2 YOLO

YOLO es un sistema de código abierto capaz de realizar detecciones de objetos en tiempo real. Para lograr su objetivo emplea la CNN de *Darknet-53 Network* o simplemente *darknet*. Esta red está compuesta por 53 capas de tipo convolucional, que pueden ser de dimensiones 1x1 o 3x3. Las salidas de todas las capas convolucionales van conectadas a una capa de Batch Normalization (BN) y seguidamente a una función de activación de ReLU. La capa de BN se añade para la optimización a la hora del entrenamiento y consiste en agrupar en lotes las imágenes introducidas en éste, acelerando así el proceso. La capa de ReLU multiplica todos los valores negativos por un coeficiente para rectificarlos, dejando intactos los valores positivos.

En la figura 2.15 se observa la estructura que utiliza la red neuronal de YOLO, pudiendo observar los filtros y las dimensiones empleados en las convoluciones, así como las últimas capas para la clasificación en las diferentes clases.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 2.15: Estructura de las capas utilizadas en YOLO [6].

El funcionamiento de esta red neuronal se basa en dividir la imagen completa en diferentes regiones, separando toda la sección de la imagen en una rejilla de tamaño $S \times S$. Cada sección será la encargada de detectar los objetos que contenga, así como asociar un valor de confianza que represente la precisión de la predicción.

A la salida de la red, se obtienen las coordenadas del punto central del objeto detectado, además de las dimensiones del cuadro que contiene dicho objeto; también se puede obtener la clase a la que pertenece

el objeto y la confianza de la predicción. Con este algoritmo se obtiene un bajo error con nuevos objetos, a pesar de sus diferencias con los empleados para el entrenamiento [33, 34].

Desde la aparición de YOLO, el sistema no ha parado de evolucionar, lo que ha dado lugar a diferentes versiones (YOLOv2, YOLOv3, YOLOv4). Una de las versiones más empleadas de YOLO ha sido Yolov3, la cuál es capaz de optimizar la velocidad de análisis de imágenes hasta 4 veces comparando con otros métodos, como se puede observar en la figura 2.16. En esta gráfica se representa el tiempo necesario para la detección de cada red para diferentes valores en la precisión media de la predicción.

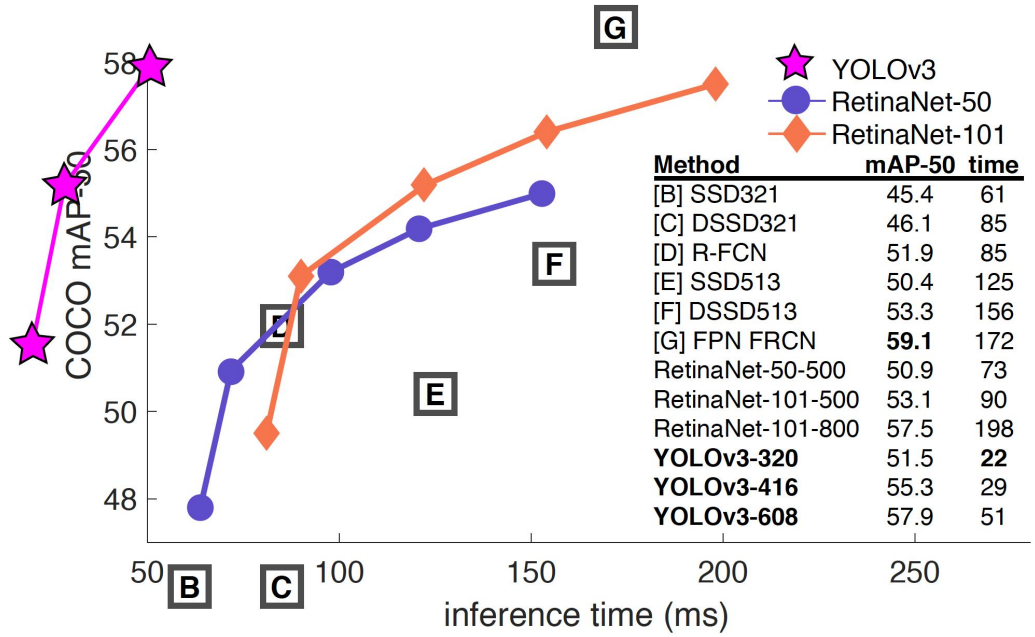


Figure 3. Again adapted from the [9], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy → [16]. Oh, I forgot, we also fix a data loading bug in YOLOv2, that helped by like 2 mAP. Just sneaking this in here to not throw off layout.

Figura 2.16: Velocidad de procesamiento de Yolov3 comparado con otras redes [7].

Para que YOLO pueda funcionar es necesario entrenarlo con un *dataset* que contenga un número elevado de imágenes. Inicialmente, los pesos se obtienen haciendo uso de los *datasets* de Common Objects In Context (COCO) [35, 36]. Con esta base de datos se consigue poner en contexto el conjunto de datos de entrada detectado mediante la red de YOLO. Este dataset, que se emplea habitualmente en visión artificial, contiene un total de 80 clases recogidas en 117 mil imágenes, entre las que se incluyen personas y numerosos objetos fundamentales, para distintas aplicaciones de detección de objetos en imágenes.

Una vez conocidos los métodos para afrontar la detección de objetos en una imagen se dedican las siguientes secciones para resolver los aspectos necesarios para la detección de abandonos. Con la información de los objetos recopilada debe realizarse un tratamiento de los datos para observar la posición y características de los objetos, para finalmente concluir que se ha abandonado cuando se cumplen las condiciones necesarias.

2.4 Validación de objetos abandonados para métodos clásicos

Una vez se detectan los objetos que forman parte de la imagen, el siguiente paso consiste en observar las características propias para ver si se trata de un objeto abandonado. Existen diversos métodos que comparan diferentes características de la región de interés del objeto, como pueden ser el color, movimiento o tamaño. Alguna de las aproximaciones más empleadas en la actualidad son las siguientes:

2.4.1 Aproximación basada en los bordes

En este método se tiene en cuenta la energía que rodea el objeto estático comparando los resultados obtenidos con la imagen obtenida del fondo. Esta técnica no es sensible a los cambios de color, por lo que funciona bien en situaciones camufladas o de baja luminosidad.

2.4.2 Aproximación basada en los colores

En este caso se estudian los colores en el interior de la región de interés delimitada por el objeto, así como los colores existentes en sus alrededores. Al contrario que en el método anterior (apartado 2.4.1), no se comporta bien en situaciones donde el objeto se encuentra camuflado.

2.4.3 Aproximación híbrida

Aprovechando ambas ventajas, este método consiste en la utilización de las 2 aproximaciones anteriores, estudiando tanto los bordes como los colores de la silueta del objeto. De esta manera se consigue una solución de compromiso que nos lleva en la mayoría de situaciones a determinar correctamente la naturaleza del objeto.

2.4.4 Aproximación basada en el movimiento

Otra técnica consiste en observar la variación con el tiempo de la posición que ocupa el objeto, pudiendo determinar si se encuentra en movimiento o parado. Por ello es necesario introducir una etapa de seguimiento de los objetos, para lo que existen diferentes alternativas; una de ellas es el Filtro de Kalman cuyo fundamento teórico se presenta en el apartado 2.5

2.5 Seguimiento de objetos mediante Filtro de Kalman

El filtro de Kalman es un algoritmo de predicción capaz de identificar un estado o variable oculta (desconocida) de un sistema, es empleado, por ejemplo, en el receptor GPS, donde se necesita estimar un valor de la posición a partir de la posición anterior y la velocidad [37]

Se denominan estados a los parámetros que se desea conocer; la entrada del filtro es el estado actual, mientras que la salida del algoritmo son los siguientes estados.

En la figura 2.17 se refleja un ejemplo de utilización del Filtro de Kalman. En ella, las entradas son las mediciones de la temperatura (en color azul); la línea roja representa las predicciones del Filtro. Como se puede observar, éstas contienen un gran error con las primeras medidas, pero según se obtiene más información, las predicciones se acercan al valor verdadero siempre y cuando se configure correctamente.

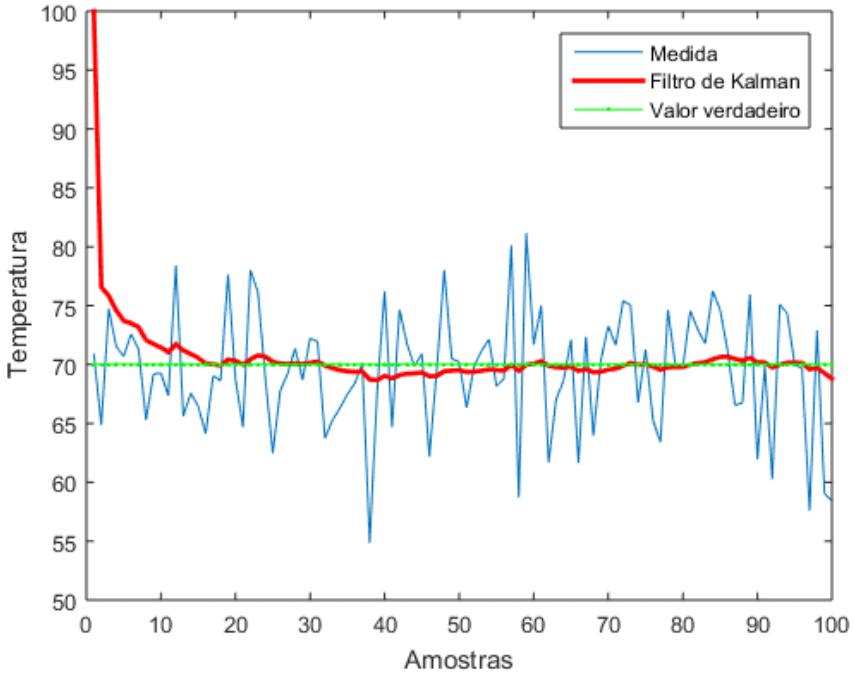


Figura 2.17: Funcionamiento del Filtro de Kalman para la predicción de la temperatura. [8]

El estudio se centra en tiempo discreto para relacionarlo posteriormente con la aplicación para la que se requiere, donde cada una de las muestras pasará a ser la información de cada fotograma. En cada una de las muestras se realiza primeramente una predicción a partir de la información de muestras anteriores, y una vez conocidas las mediciones se procede a la fase de corrección, ajustando los valores predichos con la información extraída de las mediciones.

Fase de Predicción

Como se ha comentado, en primer lugar sucede la predicción. En esta fase se comienza con una estimación de los estados en la siguiente muestra a partir de los datos medidos en la muestra actual. Para ello se emplea la matriz de transición de estados representada a partir de la letra griega Φ. Para finalizar con la predicción se calcula la matriz de covarianza (P), cuantificando el error asociado a la estimación antes de la realización de nuevas medidas. Todo esto se ve reflejado en las siguientes ecuaciones:

$$x_{k|k-1} = \Phi_k x_{k-1|k-1} \tag{2.1}$$

$$P_{k|k-1} = \Phi_k P_{k-1|k-1} \Phi_k^T + Q_k \tag{2.2}$$

Fase de corrección

La segunda fase corresponde a la de corrección, donde se actualizan todos los parámetros una vez realizadas las nuevas medidas. Para comenzar se actualizan los valores de salida del sistema, los cuales dependen tanto de las medidas obtenidas como de los valores predichos anteriormente; también se calcula la ganancia de Kalman, la cuál representa el peso de las medidas respecto a las predicciones. Se actualizan los estados añadiendo los términos relativos a las medidas que no se podían tener en cuenta en la fase de predicción; se vuelve a actualizar la matriz de covarianza una vez se han realizado las medidas. De esta manera, todas las matrices van evolucionando a medida que se toman más medidas, reduciendo el error de las predicciones a medida que las muestras aumentan. Las ecuaciones que se emplean para realizar el proceso de corrección son las siguientes:

$$y_k = z_k - H_k x_{k|k-1} \tag{2.3}$$

$$K_k = P_{k|k-1} H_k^T (H_K P_{k\bar{c}1} H_k^T + R_k)^{-1} \tag{2.4}$$

$$x_{k|k} = x_{k|k-1} + K_k y_k \tag{2.5}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.6}$$

Ajuste de parámetros Q y R

Las matrices Q y R representan las varianzas asociadas a los ruidos de los estados y de las salidas respectivamente. En la práctica dichas matrices se modifican, lo que produce una variación de la ganancia de Kalman, consiguiendo ajustar los resultados en función de las características del sistema implementado. En caso de que las medidas realizadas sean lo suficientemente buenas como para confiar en ellas, se debe reducir el valor de la ganancia K, esto se consigue reduciendo el valor de Q o aumentando el de R. Si por el contrario se quiere dar mayor importancia a las estimaciones, se debe aumentar la ganancia de Kalman, aumentando el valor de Q o reduciendo el de R.

2.6 Conclusiones

En el estudio teórico se exponen las bases del trabajo, sirviendo como punto de partida para conocer el funcionamiento interno de los métodos que posteriormente se emplean para resolver la detección de objetos abandonados, así como para saber donde atacar cuando surja algún problema.

Con los conceptos asimilados y conocido el alcance de cada método, el siguiente paso es la realización de la estructura de los programas, desarrollando las ideas principales que, en su conjunto, darán lugar a un resultado válido.

Capítulo 3

Desarrollo

La satisfacción radica en el esfuerzo, no en el logro. El esfuerzo total es la victoria total.

Gandhi

3.1 Introducción

En este capítulo se describe el trabajo desarrollado para la implementación y evaluación de las dos propuestas para detección de objetos abandonados. Para ello se explican cada una de las etapas de las que se componen.

Como se comentó en el capítulo 1, el primer paso es realizar la lectura del vídeo o secuencia de imágenes de entrada al sistema, el cuál se irá reproduciendo mientras se analizan y se obtiene información de cada uno de los fotogramas. En cada una de estas imágenes, los algoritmos desarrollados deben extraer la situación de los posibles objetos sospechosos de ser abandonados, almacenándolos en una variable para que puedan ser evaluados posteriormente; una vez obtenidos los objetos del fotograma actual, se estudian todos los objetos almacenados de *frames* anteriores, decidiendo en cada caso si deben ser eliminados (en caso de que dejen de ser sospechosos), si deben seguir siendo evaluados, o si cumplen las condiciones para activar una alarma, ya que podrían tratarse de Objetos Abandonados (OA). Al final del análisis de cada *frame* debe almacenarse la situación de los objetos detectados como abandonados, estos datos son utilizados para obtener los resultados obtenidos y poder comprobar la calidad de los mismos.

La estructura de ambos métodos es similar, pero a la hora de la realización varía tanto el lenguaje de programación como la estructura interna que se desarrolla en cada uno de los bloques. De esta manera, los resultados obtenidos con cada uno de los métodos difieren ante un mismo vídeo de entrada.

En la primera alternativa implementada se utiliza Matlab como lenguaje de programación. El algoritmo desarrollado se basa en la segmentación de imagen (estudiado en la sección 2.2) para obtener la separación fondo-objeto, resultando una imagen con la que distinguir qué partes pertenecen al fondo y cuáles al primer plano. Se continúa con una evaluación de los objetos pertenecientes al primer plano a lo largo del tiempo, concluyendo que el objeto ha sido abandonado si permanece más de un umbral definido de tiempo (o de imágenes) en la misma ubicación. Como veremos en su momento, esta forma de detección tiene algunas problemáticas, la validez de los resultados está directamente relacionada con la calidad del fondo que sea capaz de obtener en función de las características del vídeo.

En segundo lugar se aborda el problema presente mediante el empleo de una CNN, y más específicamente haciendo uso de YOLO (estudiado en la sección 2.3 para realizar la detección de objetos, almacenando únicamente objetos de interés como pueden ser mochilas, bolsos, personas, etc.A continuación se realiza un seguimiento de dichos objetos y personas a lo largo de varias imágenes consecutivas de la secuencia, para poder decidir si alguno de ellos ha podido ser abandonado, o en su defecto debe ser eliminado. Con este método se logra solucionar el problema de la obtención de fondo, pero aparecen otros inconvenientes.

3.2 Solución basada en segmentación de fondo mediante métodos clásicos

Como se ha comentado en la introducción, para la implementación de esta primera propuesta se parte del trabajo disponible en [22], por lo que se usa Matlab 2020b como plataforma de programación. Esto facilita las tareas de implementación y mejoras debido al conocimiento previo del lenguaje y la cantidad de librerías y funciones existentes que facilitan el manejo con vídeos y su manipulación. En los siguientes apartados se explica cada una de las etapas necesarias para la implementación y el correcto funcionamiento de esta alternativa, indicando los razonamientos seguidos y las funciones que han sido empleadas.

En primer lugar se presenta un esquema general con las ideas principales que se emplean en el programa (figura 3.2).

Para comenzar el programa se debe introducir un vídeo o una secuencia de imágenes, por lo que será necesario comenzar con su lectura para posteriormente realizar un bucle que sea capaz de recorrer cada uno de los fotogramas analizando diferentes aspectos.

Mediante lo analizado en 2.2.3, se comienza con la extracción de fondo de la imagen; inicialmente se opta por un fondo estático, debido a la facilidad en su obtención, pero una vez se logra el objetivo final se sustituye por un fondo dinámico para conseguir mejorar los resultados.

Una vez obtenido el fondo de la imagen, el siguiente paso consiste en la obtención de objetos en primer plano, donde residen los objetos de interés. Para que el resultado obtenido sea el óptimo, se debe aplicar un tratamiento en el formato de la imagen, trabajando finalmente con imágenes binarias.

Tras la detección de los objetos se aplica un análisis de formas, que funciona “rastreando” cada silueta extraída, para almacenar su información a lo largo del tiempo. A partir de dichos datos, éstos se comparan con unas condiciones que se deberán cumplir para determinar que un objeto ha sido abandonado.

En la figura 3.1 se muestra un diagrama general con las diferentes etapas del algoritmo que se explican en detalle en los siguientes apartados.

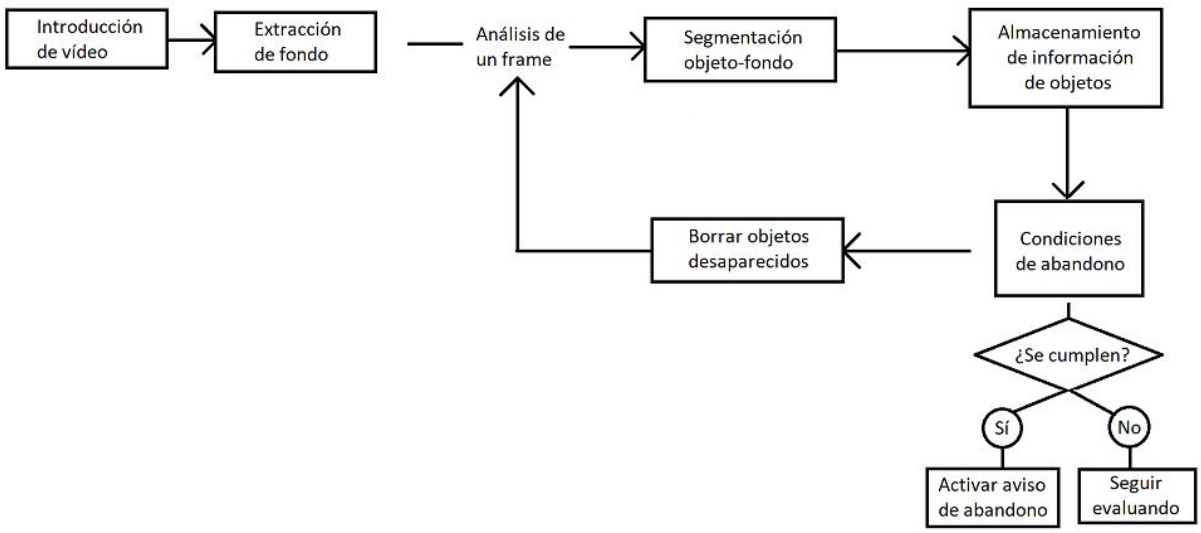


Figura 3.1: Esquema general de funcionamiento seguido para la programación.

3.2.1 Lectura y reproducción del vídeo

Al comenzar el programa, el primer paso necesario es la lectura del vídeo. Para ello se hace uso de la Toolbox[1] de Matlab denominada *Computer Vision Toolbox*. En ésta, es necesario definir una variable que vaya asociada con el archivo de entrada, seguidamente se transmite un mensaje por pantalla para que el usuario introduzca el nombre del vídeo a analizar.

Tras la selección del vídeo se recorren todos los fotogramas del mismo empleando las funciones de Matlab, como se puede observar en el código incluido en la figura 3.2.

```
90      % Lectura del video que se introduce
91 -    hVideoSrc = vision.VideoFileReader;
92 -    hVideoSrc.Filename = input('Enter the video to run:- ','s');
93 -    hVideoSrc.VideoOutputDataType = 'single';
94      % Análisis del video frame a frame
95 -    while ~isDone(hVideoSrc)
96 -        frame_no = frame_no+1;
97 -        Im = step(hVideoSrc); % Guardo el frame actual
98
```

Figura 3.2: Extracto del código desarrollado que permite la lectura de imágenes a partir de un vídeo de entrada.

3.2.2 Obtención del fondo

Una vez el programa está preparado para la lectura de vídeos el siguiente paso consiste en extraer el fondo de la imagen. Primeramente, y siguiendo la idea del proyecto de referencia [22], se almacena el primer *frame* del vídeo asumiendo que contiene el fondo de toda la secuencia de imágenes; después de realizar algunas pruebas, se observa que en numerosos vídeos se pueden obtener imágenes mejores como fondo, lo que conlleva a fallos en los resultados.

Estos fallos se producen al almacenar personas (que aparecen en el primer *frame*) en el fondo. Cuando la persona en cuestión se desplaza, y se compara la imagen actual con el fondo almacenado, se detecta

[1] Herramientas y librerías asociadas a Matlab para agregar funcionalidades sobre el programa base

una variación donde se encontraba la persona inicialmente, que lleva a analizarlo como un objeto en primer plano a pesar de contener únicamente el fondo de la imagen. Esta detección errónea, permanece estática con el tiempo (debido a que no se corresponde con ningún objeto) y se considera que se produce un abandono de la forma detectada.

Además del problema comentado anteriormente, en las zonas donde no se consigue almacenar el fondo, tampoco se logran detecciones; se consigue evitar las detecciones mencionadas, ya que todas ellas se producen en los primeros instantes del vídeo. A pesar de ello, en estas regiones se asume que existen objetos en primer plano, ya que nunca se obtiene el fondo de la imagen y no se pueden realizar detecciones lo que significa que tampoco se detectan abandonos en estas zonas.

Para minimizar los problemas que presenta el fondo estático, se implementa un fondo dinámico, el cual es capaz de seleccionar en los primeros *frames* del vídeo la imagen que menos objetos en primer plano contenga. Para ello, en primer lugar se almacena el primer fotograma de la imagen, y comparando con una función de Matlab, se almacena el número de píxeles en primer plano que contiene. Durante los primeros segundos de la grabación (400 imágenes), se evalúan el número de píxeles en primer plano que tiene la imagen actual, y si es significativamente menor que el fondo almacenado se reemplaza, quedando siempre el mejor fondo posible.

Como puede observarse en las siguientes capturas, en la figura 3.3a se representa el primer *frame* de un vídeo el cual se asume como fondo cuando se utiliza la extracción mediante fondo estático, mientras que en la figura 3.3b se aprecia una mejora de dicho fondo, ya que al evaluar varias imágenes se consigue una imagen donde las personas desaparecen de la cámara.

(a) fondo en el primer frame (b) fondo final de la imagen

Figura 3.3: Comparativa de los fondos almacenados al emplear la estrategia de fondo estático y fondo dinámico.

3.2.3 Segmentación de los objetos

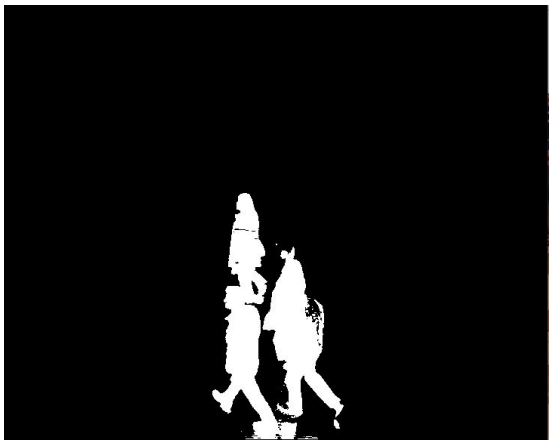
La siguiente tarea consiste en extraer todos los objetos que formen parte del primer plano, ya que en ellos se incluirán los OA que se desean detectar.

Para poder extraer la información necesaria se realiza la diferencia entre la imagen del *frame* actual y el fondo almacenado, pero previamente se deben tratar las imágenes para que presenten el formato adecuado. Esta operación podría realizarse directamente con las imágenes en formato RGB, pero se deciden transformar a YCbCr para reducir el efecto de la iluminación. El formato YCbCr almacena la componente de luminosidad o luma y las señales encargadas de almacenar la información de los colores

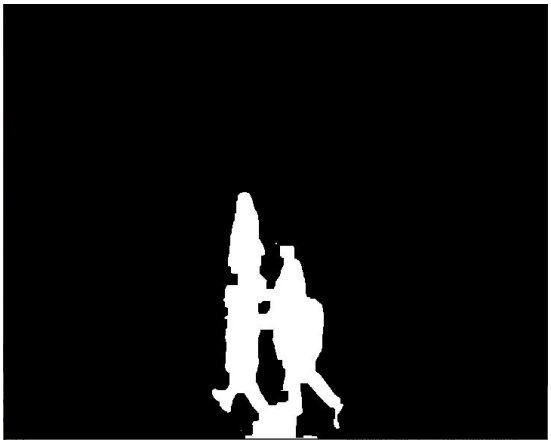
(también denominada crominancia); éstas son representadas mediante las letras Cb y Cr y almacenan la diferencia de azul y diferencia de rojo respectivamente. En la imagen 3.4b se aprecia la transformación realizada de la imagen original al formato YCbCr; la propia librería de *Computer Vision* contiene una función que realiza directamente la transformación. Se crea una variable multidimensional, en la cuál se almacenan las componentes Cb y Cr de todos los píxeles de la imagen, que se utiliza posteriormente para fijar qué valores pertenecen al fondo según el umbral establecido.

(a) Imagen original en RGB

(b) Imagen modificada en formato YCbCr



(c) Imagen binaria



(d) Imagen con operación de cierre

Figura 3.4: Tratamiento de imagen para la obtención de fondo.

A continuación se elimina el fondo de la imagen, realizando la resta en valor absoluto de cada uno de los píxeles de las imágenes en formato YCbCr, y el resultado obtenido se convierte en escala de grises para poder binarizar la imagen posteriormente; de esta forma se obtiene un 0 cuando la diferencia de ambas imágenes sea nulo (o tenga un valor muy bajo), es decir, si coincide con el fondo almacenado. Se obtiene un 1 cuando los valores difieran significativamente y se trate de objetos de interés. Para que este proceso se realice de forma adecuada, en el proceso de binarizar la imagen se debe añadir un umbral para indicar que valores son aceptados como fondo y como objeto. Para seleccionar dicho umbral se emplea la función “adaptthresh” que varía en función de la luminosidad detectada en la imagen bajo estudio.

Este tratamiento de imágenes suele resultar bastante común en aplicaciones similares, ya que permite reconocer y localizar fácilmente las zonas donde se encuentran los objetos que posteriormente se analizan, independientemente del propósito final. En la figura 3.4c se representa un ejemplo del resultado obtenido al binarizar la imagen de referencia (3.4a) según los objetos en primer plano.

Como se puede observar, el resultado es el esperado, pero no se logran extraer figuras cerradas. Para mejorar el resultado se aplica una operación de cierre a la imagen (como se pudo ver en el apartado 2.2). Se define una forma cuadrada de 10 píxeles de tamaño, ya que supone un compromiso que consigue rellenar los huecos de cada forma sin unir formas que pertenecen a objetos diferentes. Esta forma se introduce a la función encargada de realizar la operación de cierre y se consiguen formas cerradas sin huecos en su interior. Esta mejora se observa en la imagen 3.4d, en la cuál ya se aprecian con mayor claridad, las siluetas que se han podido extraer.

3.2.4 Análisis de formas

Para continuar se realiza un análisis de las formas de la imagen obtenida. Aprovechando que se trata de una imagen binaria, se emplea la función *vision.BlobAnalysis* para facilitar el trabajo; se crea un objeto de análisis, indicando que no analice los objetos que se encuentran en el borde de la imagen y el número máximo de detecciones en la imagen, en este caso se sitúa en 200 para no aumentar el uso de memoria, pudiendo modificarlo si fuera necesario. También se inicializa el tamaño mínimo y máximo de los objetos, que puede variar dependiendo de las imágenes introducidas, en función de las dimensiones y la cercanía de los objetos. Aplicar estos límites evita detecciones de objetos que puedan resultar erróneos. Como ejemplo de lo anterior, en la imagen 3.4a se puede apreciar un tren en el fondo, al aparecer en la imagen se trata como un objeto en primer plano ya que no forma parte del fondo, pero al realizar el análisis de formas es demasiado grande, por lo que el sistema no lo tiene en cuenta.

Una vez inicializado el objeto de trabajo se utiliza la función en cada imagen del vídeo, logrando extraer todos los objetos presentes; de esta forma se puede conocer el área, el centroide y el cuadro delimitador[2] de cada objeto/persona. En la figura 3.5 se incluye un ejemplo extraído de un *frame*, donde se representan todas las formas encontradas.

Figura 3.5: Ejemplo del análisis de formas realizado en un fotograma, con la representación mediante cuadros delimitadores.

[2]Proviene de la palabra *Bounding Box*, y contiene las coordenadas de las esquinas de un rectángulo que delimita la forma

3.2.5 Detección de abandonos

Una vez conocida la información de cada uno de los objetos en un determinado *frame*, ésta se almacena en una variable. Posteriormente se realiza un proceso de asociación para objetos que se encuentran parados, ya que son candidatos a objetos abandonados; cabe destacar que en este caso no se almacena la información actual del objeto, únicamente se modifica la ya existente, aumentando un contador que representa el número de *frames* que lleva detenido.

Para poder asumir que un objeto ha sido abandonado se imponen las condiciones que se deben cumplir:

1. En primer lugar, el objeto debe permanecer en el mismo lugar durante un cierto tiempo, que puede variar en función de las imágenes. Este umbral se encuentra por defecto en 100 frames, lo que equivale a poco más de 3 segundos. Al aumentar este valor, el tiempo para detectar el abandono aumenta, mientras que si se reduce, aparecen verdaderos negativos (detecciones de objetos no abandonados).

2. El objeto debe aparecer en alguna de las últimas 7 imágenes. De esta forma se puede asegurar que el objeto continúa en la imagen. Si no se cumple esta condición puede deberse a que el objeto ya no permanezca estático o la cámara no pueda percibirlo si otro elemento (persona u objeto) está ocultándolo.

3. El fondo debe ser el definitivo, por lo que mientras se evalúa la imagen para obtener un fondo dinámico no se detectarán objetos abandonados. Si el fondo no es el definitivo pueden detectarse formas estáticas que produzcan verdaderos negativos en los resultados, por lo que no se tiene en cuenta.

4. Se añade una funcionalidad para definir un área de la imagen de interés, por lo que si el centroide del objeto sospechoso se encuentra fuera de dicha región no se tiene en cuenta en los resultados. Resulta útil dependiendo de la orientación de la cámara para centrar el estudio según las zonas. Por ejemplo, en una estación de tren en la que se encuentra una cafetería, sería interesante eliminar el seguimiento en la cafetería, ya que aparecen numerosos objetos estáticos que no se desea que se evalúen como abandonos.

Para visualizar los resultados del programa, se introducen siluetas de color verde en todos los objetos detectados en la imagen actual para conocer las capacidades de detección de objetos y verificar cuáles de ellos es capaz de detectar. Además, cada vez que un objeto cumple las condiciones de abandono, la silueta pasa a color rojo y el rectángulo definido se rellena en el mismo color. También se añade un aviso auditivo que se emite mientras el objeto está presente y abandonado. En la figura 3.6 se observa un ejemplo de su representación al producirse el abandono de un objeto.

Eliminación de objetos que dejan de detectarse

A la par que se almacena la información de todos los objetos que aparecen en las imágenes, se debe eliminar la información innecesaria, ya que puede ralentizar el proceso de detección. Para ello, cuando no se detecta un objeto, se aumenta un contador que contiene el número de *frames* que lleva sin aparecer; cuando este contador alcance el valor de 40 el objeto se elimina asumiendo que ha desaparecido de la escena o se encuentra ocluido por otro objeto.

Este valor se puede modificar para evitar que un objeto que ha salido de la imagen ocupe espacio innecesario, pero el valor no puede ser muy reducido, ya que puede ocurrir que el objeto se encuentre oculto detrás de otra silueta (de un objeto o persona durante un tiempo), por lo que en caso de estar abandonado se asuma que ha desaparecido y se borre, teniendo que repetirse todo el proceso cuando vuelva a ser visible.

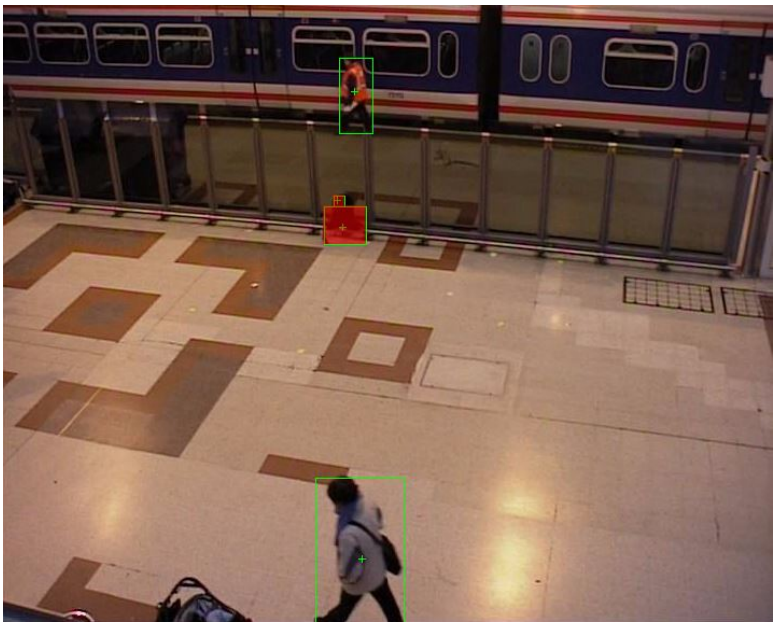


Figura 3.6: Ejemplo de representación de resultados en el momento que se produce un abandono.

3.3 Solución basada en redes neuronales convolucionales

Tras analizar las debilidades del método basado en técnicas clásicas, debidas a la dependencia del fondo obtenido, se decide realizar un programa con la misma funcionalidad pero aplicando redes neuronales para aprovechar las ventajas que presentan frente a los métodos clásicos.

Tras analizar las alternativas disponibles se elige el lenguaje de programación Python, siendo el más extendido para la programación de CNN e inteligencia artificial en general [38]. Se escoge la red neuronal de YOLO gracias a la facilidad de puesta en marcha y la ejecución en tiempo real; con la red neuronal escogida, se decide utilizar el sistema operativo Linux y la aplicación de Pycharm como plataforma de programación.

A continuación se presenta la estructura general del programa seguido de las explicaciones de cada una de las partes que lo componen.

3.3.1 Estructura general

De manera análoga a la sección anterior (3.2), el comienzo del código se emplea para introducir el archivo de vídeo que se desea analizar. Seguidamente se recorre en bucle cada uno de los fotogramas del vídeo, en busca de la posición de cada uno de los objetos que puedan ser útiles para la detección. Como se ha comentado anteriormente, se utiliza la red de YOLO en cada uno de los fotogramas, lo que devuelve a su salida todos los objetos que es capaz de extraer de la imagen de entrada, así como el factor de confianza, las coordenadas y el tamaño de cada uno de los objetos detectados. Una vez conocidos y almacenados los objetos de interés (personas, mochilas, bolsos y maletas), se dividen en 2 únicas clases: personas y objetos, a las cuáles se aplica un Filtro de Kalman para realizar un seguimiento de cada elemento mientras sea visible en la imagen. Con toda la información extraída se comparan los datos almacenados con unas condiciones que sirven para determinar cuando se abandona un objeto.

En la siguiente figura se aprecia de forma gráfica el esquema general del algoritmo implementado.

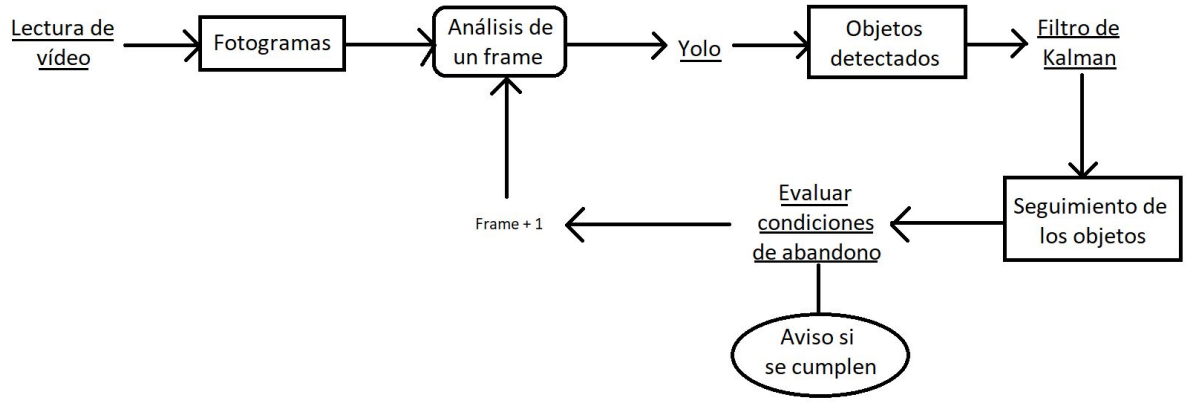


Figura 3.7: Esquema general del sistema para detección de objetos abandonados basado en YOLO.

Cada una de las etapa mostradas en la figura se describe en detalle a continuación.

3.3.2 Manipulación de vídeos

Para comenzar con la escritura del código se introduce la lectura de un vídeo seguido de su representación para comprobar su correcto funcionamiento; para ello se ha hecho uso de las librerías *OpenCV* (también representadas con la abreviatura cv2); estas librerías contienen numerosos algoritmos específicos para la manipulación de imágenes y detección de objetos que facilitan el desarrollo del trabajo.

Se utiliza la función *VideoCapture* para introducir la ubicación del archivo que se desea abrir, asociando dicha función a una variable donde se almacena el vídeo. Para evitar que el programa presente errores debido a archivos dañados, se introduce una condición con la función *Video.isOpened* para comprobar que la lectura de dicho archivo ha sucedido correctamente.

A continuación, se introduce un bucle que permanece ejecutándose hasta terminar de recorrer cada fotograma del vídeo. En su interior se realiza la lectura del archivo mediante la función *video.read* lo que devuelve a su salida 2 valores, el primero de ellos (*ret*) de tipo bool[3] toma el valor de *TRUE* cuando es capaz de capturar la imagen, cambiando su valor a *FALSE* cuando no sea capaz de capturarla; el segundo de los valores (*frame*), almacena la imagen actual con la cuál se trabaja a continuación.

Una vez realizados todos los pasos anteriores se puede visualizar en una ventana nueva el vídeo a procesar.

3.3.3 Puesta en marcha de YOLO

EL siguiente paso consiste en implementar la red de YOLO para poder realizar la detección de objetos en cada una de las imágenes que componen el vídeo.

Para comenzar, se descargan todos los archivos que componen la red [7]. Para lograr el funcionamiento óptimo se ajusta para que funcione sobre la tarjeta gráfica (o GPU) lo que agiliza el tiempo empleado para la detección; también se deben añadir las direcciones que ocupan los archivos (*yolov3.cfg* y *coco.data*) dentro del equipo. En la figura 3.8 se observa uno de los primeros resultados obtenidos, comprobando el correcto funcionamiento.

[3]Los datos de tipo bool o lógicos, son aquellos que pueden representarse mediante un valor binario, es decir, únicamente pueden tomar 2 valores que representan típicamente verdadero y falso.

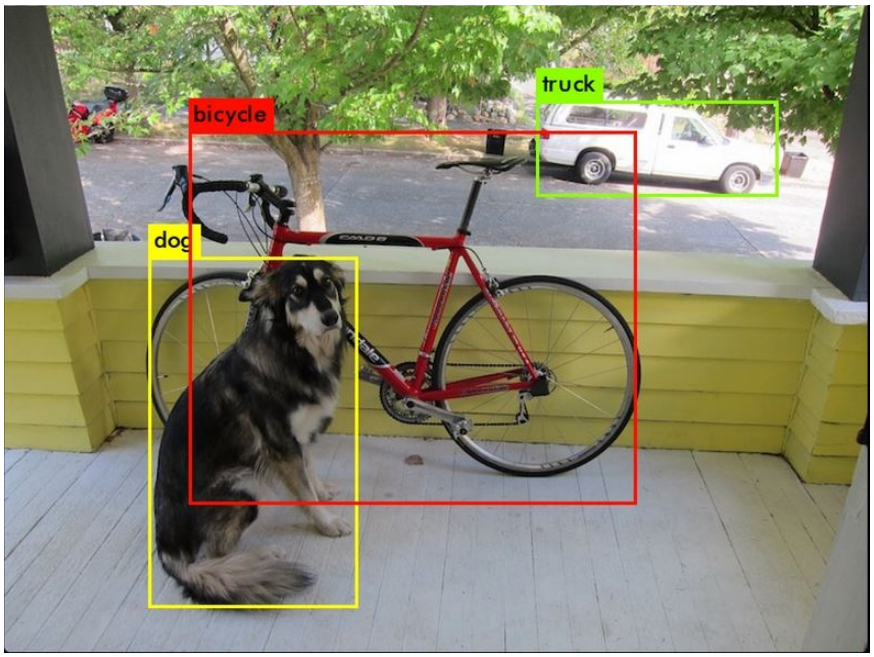


Figura 3.8: Resultado de la primera prueba de YOLO en una imagen mostrando las detecciones realizadas.

Posteriormente se ajusta el código existente para hacer posible su funcionamiento con vídeos, para ello se llama al documento que carga todas las capas de YOLO al comienzo, añadiendo la función de detección cada vez que se cambia de imagen a medida que transcurre el vídeo.

En este punto, se consigue obtener por cada *frame* del vídeo una lista que contiene todas las detecciones. Por cada detección se indica la clase a la que pertenece, el porcentaje de confianza, las coordenadas del centroide, el alto y ancho del objeto. Para facilitar posteriormente el trabajo se añaden como salidas el número de *frame* de la detección y el color que presenta el objeto en su centroide. En la figura 3.9 se observa el resultado devuelto por YOLO.

```
[(b'person', 0.8542858362197876, (523.1253662109375, 525.9746704101562, 69.22927856445312, 97.86112976074219))]
[(b'person', 0.845879316329956, (507.0492248535156, 524.6845703125, 98.56257629394531, 105.72979736328125))]
[(b'person', 0.7519022822380066, (506.27239990234375, 523.50341796875, 83.96092987060547, 106.28739929199219))]
[(b'person', 0.7517887949943542, (496.9775390625, 521.4517211914062, 96.3784408569336, 104.60314941406250))]
[(b'person', 0.8964788317680359, (502.2398681640625, 519.6934204101562, 75.58647155761719, 104.87992858886719))]
[(b'person', 0.7507040500640869, (498.95697021484375, 518.9138793945312, 73.62985992431640, 111.68364715576172))]
[(b'person', 0.8278431296348572, (485.291015625, 519.4611206054688, 84.18556976318360, 112.86874389648438))]
```

Figura 3.9: Ejemplo de la lista de salida devuelta por YOLO.

Con la red lista para la detección de objetos en vídeos, únicamente es necesario mostrar los resultados de forma visual, de esta forma se comprueban fácilmente los resultados:

Representación de los resultados de la detección de YOLO

Se utiliza una nueva hoja que dispone de la variable de salida de YOLO, para poder generar una secuencia de imágenes que reproduzcan de manera visual dichos resultados.

Para comenzar, se recorre en un bucle (*for*) todos los objetos del detector almacenando en diferentes variables la posición del objeto (centroide) y sus dimensiones (alto y ancho). Se realizan operaciones para calcular las coordenadas de la esquina superior izquierda e inferior derecha, ya que serán las necesarias para insertar una forma rectangular. Además se indica el color que tendrá su contorno y el grosor que se desea.

También se incluye la lectura de la clase a la que pertenecen para poder identificar si la detección se efectúa de forma correcta, y se visualiza el porcentaje de confianza para ver la calidad de la detección. Estas variables son de texto, por lo que se indican tanto las coordenadas donde se quiere representar como el color y el grosor de la letra.

Se comprueba el correcto funcionamiento de la representación y se repite el procedimiento pero representando únicamente los objetos de interés para lograr el objetivo final; de esta forma se puede seleccionar entre la representación de todos los objetos detectados o simplemente los objetos útiles para la detección de abandono. En la figura 3.10 se presenta un ejemplo de la representación final de las detecciones de interés. Se observan los cuadros delimitadores, además de la clase a la que pertenecen y la confianza de detección de YOLO.

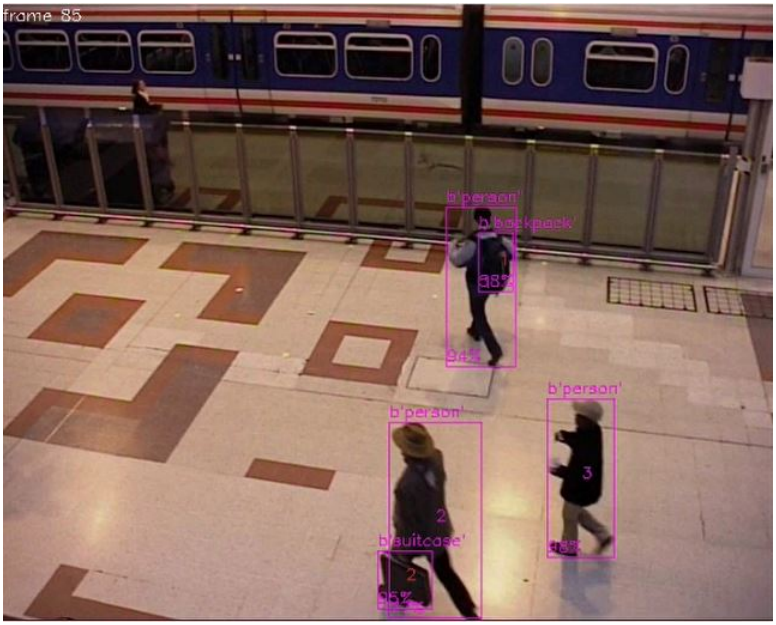


Figura 3.10: Ejemplo de la representación de las detecciones realizadas por yolo en una imagen.

3.3.4 Seguimiento de personas y objetos

Una vez almacenados los datos de detección necesarios para la aplicación, se implementa un sistema de seguimiento mediante un filtro de Kalman como se estudió en el apartado 2.5. Existen dentro de las librerías de *OpenCV* las funciones de predicción y corrección que facilitan el trabajo.

Para comenzar creando el filtro de Kalman, se crea una clase[4] que se asociará con una variable, al comienzo de la clase se inicializa el filtro, introduciendo las matrices características. Se introducen las dimensiones del filtro, indicando 4 variables para los estados: posición (x, y) y velocidad (v_x, v_y) con 2 coordenadas para cada variable) y 2 dimensiones para las entradas (coordenadas x e y de la posición), y a continuación, se inicializan las matrices de medida y transición.

En las expresiones 3.1 y 3.2 se observan las ecuaciones empleadas para el cálculo de estados y salidas del sistema. La variación de tiempo se establece en el tiempo de una imagen, como suceden 30 *frames* por segundo, esta variación equivale aproximadamente a 34 ms.

[4]Una clase provee una forma de empaquetar datos y sus funcionalidades. Una vez creado, se asociarán variables al nuevo tipo de objeto.

$$\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & 0 & \delta t & 0 \\ 0 & 1 & 0 & \delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_k \tag{3.1}$$

$$\begin{pmatrix} x \\ y \end{pmatrix}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_k \tag{3.2}$$

Para que la programación se realice de forma más simple se crean 2 funciones; la primera, para la fase de predicción, que hace la llamada a la función *predict* y devuelve los valores extraídos del filtro; la segunda función es la función asociada a la fase de corrección, en ella se almacena en una variable las coordenadas a introducir y se introduce la variable creada en la función *correct* del filtro.

Con el filtro inicializado y las funciones creadas, el siguiente paso es realizar un código que sea capaz de predecir las posiciones de los objetos. Para ello, la primera vez que aparezca un objeto, se crea y asocia un filtro de Kalman que ocupará una posición en una lista, si el objeto aparece anteriormente, se evalúa la situación de los objetos detectados de antemano para intentar asociarlos.

En el proceso de asociación se extraen las coordenadas de la detección, y se calcula la distancia entre centroides, y si la menor distancia con un objeto del fotograma anterior se sitúa por debajo de un umbral, se asume que se trata del mismo objeto, por lo que se realiza una corrección de dicho filtro con las coordenadas detectadas. En caso de que las distancias con todos los objetos anteriores se encuentren por encima del umbral establecido, se asume que el objeto no coincide con ninguno de los almacenados y se crea un nuevo filtro para esta detección, corrigiendo con las coordenadas existentes.

Una vez analizadas todas las detecciones actuales se predicen las posiciones de todos los filtros creados. En el caso del seguimiento de las personas, si alguno de los filtros no ha podido ser asociado con las detecciones, se almacenará la predicción; ésta predicción puede contener demasiado error si el número de medidas introducidas no es muy elevado, por lo que solo se almacena la predicción en caso que se hayan introducido 15 o más medidas anteriormente. El proceso de almacenar la predicción no se realiza en el caso de los objetos por razones que se exponen en el siguiente apartado referente a la detección de abandono.

En la figura 3.11 se representan las acciones realizadas ante la llegada de las detecciones de una imagen, en función de la asociación con los objetos almacenados previamente.

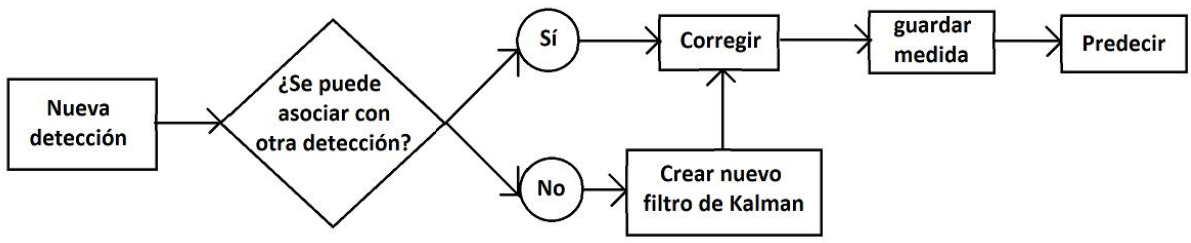


Figura 3.11: Diagrama para la representación de la secuencia seguida cuando se recibe una detección.

Para finalizar se eliminan todos los objetos que no se detectan en los últimos segundos del vídeo (se establece el límite en 80 fotogramas) ya que se supone que el objeto ha desaparecido de la imagen y debe ser borrado para no ocupar espacio en las variables.

En este momento del programa ya no se distingue entre las clases de cada objeto, únicamente se diferencia entre personas y objetos de interés, por lo que el seguimiento se realiza por separado. Primero se realiza el seguimiento de las personas, y al finalizar el seguimiento de los objetos, se introduce un proceso de asociación de objeto-persona, donde, calculando la distancia de cada objeto con todas las personas, se puede asumir que todos los objetos que aparecen en la imagen habrán sido introducidos por la persona más cercana en ese momento.

3.3.5 Detección de abandono

El proceso de detección de abandonos comienza en el último paso de seguimiento de los objetos, debido a que según la situación de estos se puede extraer la información necesaria.

Como ya se comentó anteriormente, en el caso de los objetos, nunca se almacena la predicción; en caso de almacenarla, aunque el objeto permanezca inmóvil, la predicción sufre un desplazamiento que puede causar un gran error a lo largo del tiempo, por lo que si no se detecta se asume que el objeto se encuentra detenido o ha desaparecido de la imagen.

Para poder definir si el objeto sigue en la imagen o ha desaparecido, se utiliza inicialmente la funcionalidad implementada del color del centroide a la hora de detectar el objeto; por un lado se conocen las componentes de color que tiene el centroide del objeto en la última imagen que fue detectado, por lo que se comparan dichos valores con los colores que existen en el *frame* actual. Evaluando la distancia cuadrática entre ambos valores se podrá asumir que el objeto sigue en la misma posición, mientras que si el color ha cambiado el objeto desaparece de la imagen.

Una vez conocidos todos los objetos que existen en la imagen que se está evaluando, se almacenan las características del objeto con el entorno que permiten decidir en que situaciones se encuentra abandonado. A continuación, se exponen los pasos seguidos para obtener sus características:

- En primer lugar se calcula la distancia que existe con la persona asociada al objeto, aumentando un contador si supera un umbral que varía en función del tamaño de la imagen. Este contador, representa el tiempo que permanece un objeto alejado de la persona responsable del mismo.
- El segundo paso se lleva a cabo cuando algún filtro asociado a las personas desaparece, es decir, una persona que se detectaba en la imagen, se deja de detectar. En esta situación, pueden darse 3 modificaciones en la persona asociada a un objeto:
    1. Si la posición en la lista de la persona asociada al objeto no se modifica, la asociación del objeto permanece intacto.
    2. Si la posición en la lista de la persona se ve desplazada, la asociación del objeto debe modificarse por la nueva posición que adopta.
    3. Si la persona eliminada se encuentra asociada a un objeto, se deberá almacenar la información, ya que podría tratarse de un abandono.
- Una vez analizada la información necesaria se evalúa para poder detectar un abandono. Si la persona asociada a un objeto se encuentra a una distancia lejana durante continuados *frames*, o desaparece de la imagen, se evalúa mediante el color, para concluir que si este difiere del almacenado en la

última detección habrá desaparecido, mientras que si el color permanece similar, el objeto continúa en la imagen y se encuentra abandonado.

Al obtener los resultados se desea probar otro método para comprobar el abandono una vez que desaparece la persona asociada; para ello, en caso de que el objeto se detecte en la misma posición que la última detección, se aumenta un contador que representa el tiempo que el objeto permanece estático; de esta manera si el objeto se pone en movimiento el contador pasará a 0, mientras que si la persona desaparece y el contador se sitúa por encima de un valor se asume que el objeto sigue presente y se encuentra abandonado. Este razonamiento es válido, ya que en el momento que una persona recoja el objeto, la red es capaz de detectarlo, y si presenta movimiento no puede ser considerado como abandono.

En la figura 3.12, se observa un ejemplo del abandono de un objeto. En primer lugar se aprecia la detención de la persona, para posteriormente depositar el objeto (mochila) que lleva en el suelo. Finalmente, se aleja del objeto abandonando el objeto que portaba.

Figura 3.12: Secuencia de abandono de un objeto. Contiene el momento de la detención de la persona, el depósito del objeto, y su posterior abandono.

Cuando se define un abandono se estudia cada cierto tiempo la situación del objeto para asegurarse que sigue abandonado. Para ello, se decide imponer que una vez que se detecta un objeto como abandonado, no deja de ser abandonado hasta que este objeto presente movimiento. En caso de que la posición del objeto se aleje de la posición de abandono se elimina el aviso de abandono, y el objeto se asocia de nuevo a la persona más cercana.

Capítulo 4

Resultados

Si buscas resultados distintos, no hagas siempre lo mismo.

Albert Einstein

4.1 Introducción

En este capítulo se muestran y analizan los resultados obtenidos para así ambos métodos, para cuantificar la calidad y eficacia que se puede obtener de cada uno, y posteriormente, poder compararlos.

4.1.1 Configuración experimental

Para observar los resultados que es capaz de obtener cada método en los vídeos de entrada, se realiza un código que nos permite comparar los resultados.

Para que este proceso sea posible se deben obtener los resultados del programa principal, guardando en un archivo de texto todos los objetos que han sido detectados como abandonados. Se almacena el fotograma en el que aparecen, y la posición y el tamaño del objeto.

Una vez que se tienen los resultados obtenidos, se introducen en el nuevo código los instantes y las coordenadas en donde se sitúa el OA. Se analiza la información real del objeto y la extraída del estudio del vídeo para poder clasificar los resultados según las siguientes categorías:

- **Verdadero Positivo (TP).** Se considera verdadero positivo cuando exista un objeto abandonado, y el programa sea capaz de detectarlo.
- **Verdadero Negativo (TN).** Dicha situación ocurre cuando se detecta un OA, pero en realidad no existe dicho abandono.
- **Falso Positivo (FP).** Se produce cuando existe un objeto abandonado, pero no se detecta.
- **Falso Negativo (FN).** Se da cuando no existe ningún objeto abandonado y el programa tampoco lo detecta.

Para poder cuantificar y obtener estadísticas se definen los parámetros de *precision* o precisión y *recall* o exhaustividad. La precisión de los resultados representa el porcentaje de abandonos que se han

detectado, es decir, la capacidad para detectar abandonos cuando se han producido, se calcula como el número de imágenes que detecta el objeto (TP)) entre el número total de imágenes que debería detectarlos (TP + FP). En cuanto a la exhaustividad, representa la relación entre medidas correctas (verdaderos positivos) y detecciones totales (TP + TN).

En 4.1 y 4.2, se aprecian las respectivas fórmulas empleadas para hallar cada uno de los valores. En el análisis de los vídeos no es necesario realizar el conteo de los falsos negativos, ya que se producen cuando no se detecta un objeto, y tampoco existe un objeto abandonado, pero no es necesario para el cálculo de los parámetros indicados.

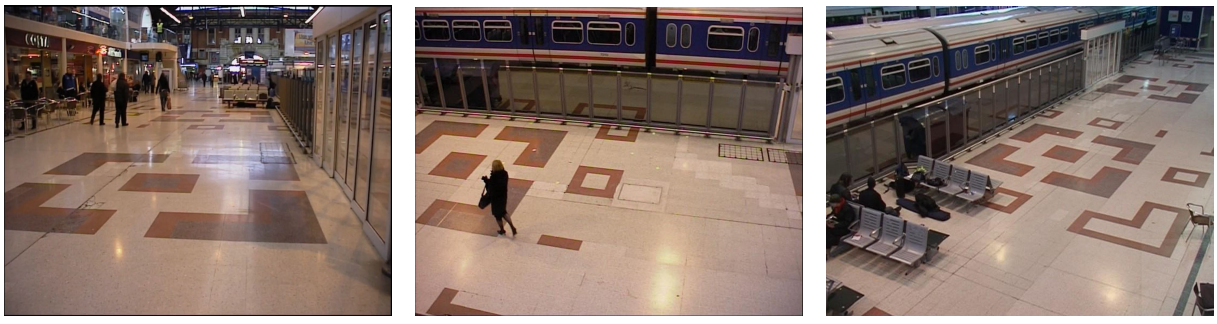
$$precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$recall = \frac{TP}{TP + TN} \tag{4.2}$$

4.1.2 Dataset PETS2006

Se utiliza la colección de vídeos de PETS06 [39], la cual incluye 7 vídeos de una estación de tren, grabada para la detección de objetos abandonados. Se analizan los datos de 3 cámaras diferentes (para cada uno de los vídeos) las cuales están sincronizadas y, conocidos los momentos en los que se producen los abandonos y la posición del objeto en el momento del abandono, se compara con los resultados de cada uno de los programas para extraer las estadísticas correspondientes.

Para la clasificación de resultados se permite un margen de error en la posición del centroide del objeto, además, se delimita una zona de interés en la imagen para facilitar el trabajo, evitando áreas donde se hace difícil la detección de objetos. En la figura 4.1 se aprecia la perspectiva y el campo de visión captado por cada una de las 3 cámaras que se analizan. Para evitar falsas detecciones en zonas que no tienen interés, se decide incluir unos límites delimitando una zona de interés. En la primera cámara (4.1a), se limita tanto la planta superior como la zona de las mesas situada en el borde de la imagen. En la segunda (4.1b), se tiene una vista bastante despejada que facilitará el análisis, para evitar detecciones innecesarias se limita la altura eliminando las cercanías de las vías del tren. Por último, en la figura 4.1c se tiene la vista de la última cámara, en ésta se reduce su altura para evitar de nuevo detecciones en las vías del tren.



(a) Cámara 1 (b) Cámara 2 (c) Cámara 3

Figura 4.1: Captura de imagen de las cámaras de la colección PETS06.

A continuación, en la tabla 4.1 se muestra la información sobre objetos abandonados proporcionada del dataset PETS06, para cada uno de los 7 vídeos. Para los objetos abandonados se indica el rango de imágenes en los que aparece dicho objeto así cómo las coordenadas donde se sitúa para cada una de las cámaras. El sistema de coordenadas se sitúa en la esquina superior izquierda, aumentando las coordenadas X e Y hacia la derecha y hacia abajo respectivamente.

Vídeo	Cámara	*Frame* inicial	*Frame* final	Coordenada X	Coordenada Y
1	2	2145	3021	327	228
	3	2145	3021	398	208
2	1	1550	2551	527	221
	2	1550	2551	338	223
	3	1550	2551	393	209
3	1, 2 y 3	No hay abandono			
4	1, 2 y 3	No hay abandono			
5	1	2000	3400	558	221
	2	2000	3400	322	224
	3	2000	3400	390	209
6	1	1700	2801	540	218
	2	1700	2801	315	213
	3	1700	2801	372	213
7	1	1700	3401	531	221
	2	1700	3401	283	225
	3	1700	3401	396	211

Tabla 4.1: Situación de los objetos abandonados para la colección de vídeos PETS06 para cada uno de los vídeos analizados. No aparece el vídeo 1 de la cámara 1 por aparecer dañado en su descarga.

4.2 Resultados obtenidos para el primer método

El nuevo código comentado en 4.1.1, compara los resultados obtenidos clasificándolos en Verdaderos Positivos, Verdaderos negativos y Falsos positivos, según el criterio que se expuso en el mismo apartado. Una vez obtenidos los resultados se calcula la precisión (*precision*) y exhaustividad (*recall*) de las medidas.

Conocidos los métodos para extraer resultados y cuantificarlos, se analizan los vídeos con el programa realizado en Matlab, para así compararlos con los resultados que se desea obtener, representados en la tabla 4.1.

A continuación, en la tabla 4.2 se exponen los resultados para cada vídeo así como el análisis global y las gráficas 4.2 y 4.3 donde se visualiza la precisión y la exhaustividad para cada vídeo.

Vídeo.Cámara	TP	TN	FP	Precision(%)	Recall(%)
1.2	876	0	0	100	100
1.3	774	10	102	88.3	98.7
2.1	755	154	246	75.42	83.05
2.2	767	1291	234	76.62	37.27
2.3	941	675	60	94	58.23
3.1	1	187	0	100	0
3.2	1	270	0	100	0
3.3	1	3211	0	100	0
4.1	1	473	0	100	0
4.2	1	1406	0	100	0
4.3	1	2593	0	100	0
5.1	0	0	1293	0	0
5.2	1012	0	281	78.3	100
5.3	964	1722	329	74.55	35.89
6.1	872	87	229	79.2	90.92
6.2	983	0	118	89.28	100
6.3	883	2038	218	80.19	30.23
7.1	500	387	1201	29.39	56.37
7.2	1253	29	448	73.66	97.74
7.3	492	0	1209	28.92	100
Total				**73.37 %**	**49,42 %**

Tabla 4.2: Resultados obtenidos para el primer método.

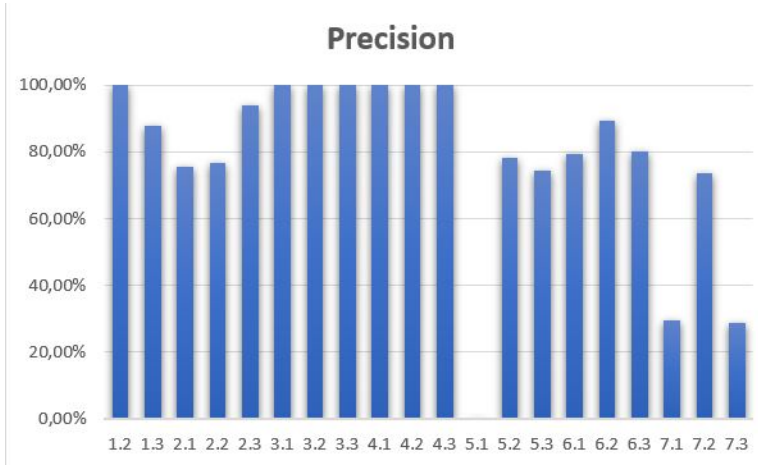


Figura 4.2: Precisión obtenida en los resultados del primer método para cada uno de los vídeos.

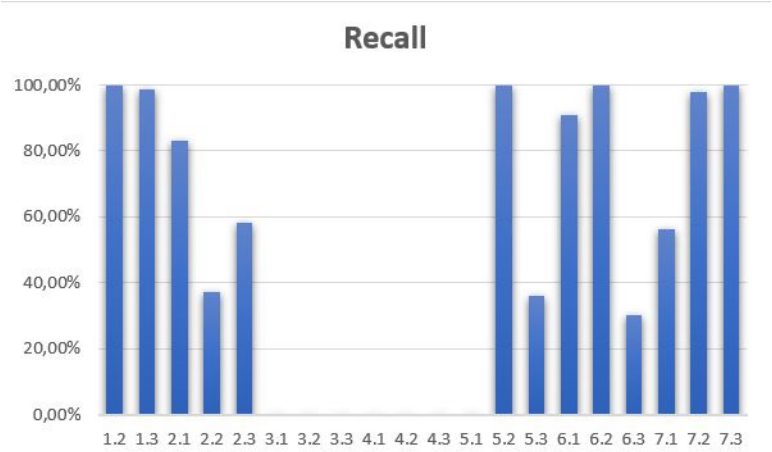


Figura 4.3: Recall obtenido en los resultados del primer método para cada uno de los vídeos.

De los resultados anteriores, se extrae una precisión aceptable, lo que significa que el programa detecta de manera correcta los abandonos que se producen. En cuanto al *recall*, se obtiene un valor bastante reducido, debido a la detección de abandonos de objetos que no lo son. Estos resultados se deben al problema que presenta el método. Como ya se ha comentado, dependen del fondo obtenido en cada situación. En el apartado 4.4 se explica con mayor detalle estos resultados.

En las siguientes figuras (4.4 y 4.5) se aprecian tanto resultados incorrectos como correctos para observar las debilidades del método empleado así como la correcta detección de los objetos. También aparece en la figura 4.6 el fondo que se ha almacenado para la secuencia.

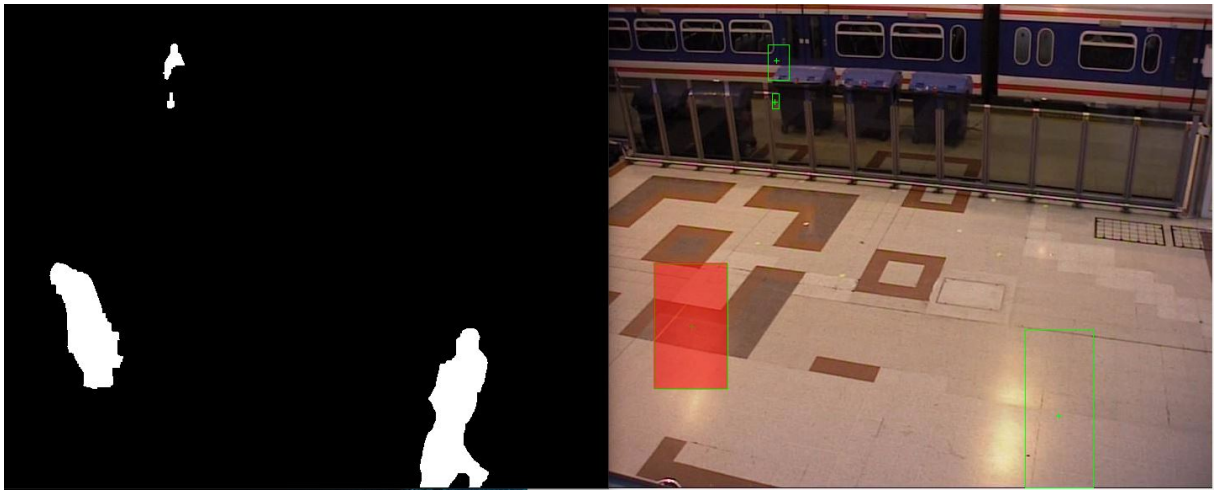


Figura 4.4: Ejemplo de detecciones erróneas debido a problemas al encontrar el fondo de la imagen.

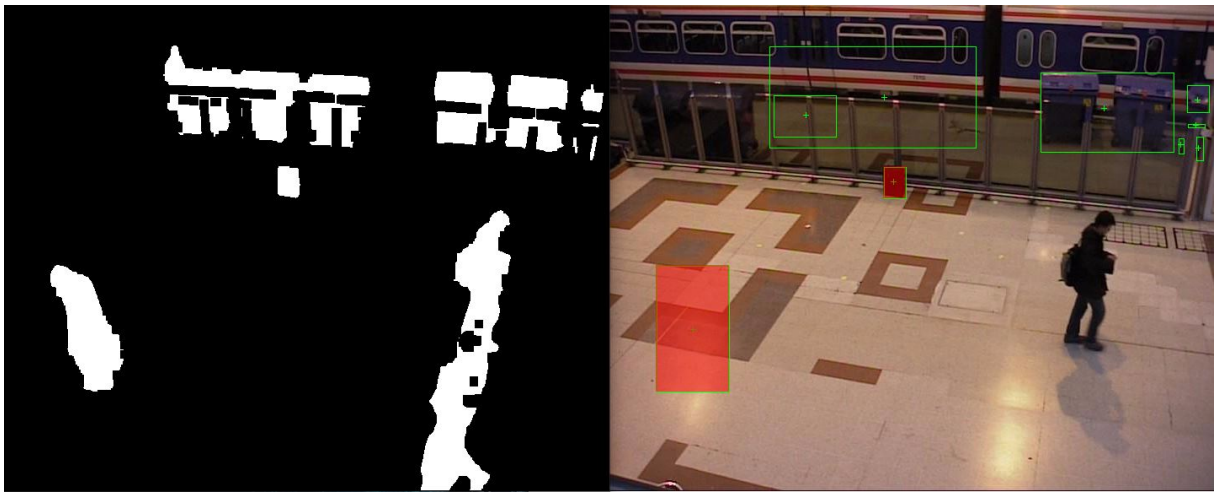


Figura 4.5: Ejemplo de detecciones de objetos abandonados con las condiciones favorables.

Figura 4.6: Fondo almacenado para extraer los resultados anteriores.

4.3 Resultados obtenidos para el segundo método

Una vez finalizado el segundo método se comprueban los resultados que es capaz de obtener el programa frente a los mismos vídeos. Para comenzar se extraen los resultados detectando los abandonos a partir del color que presenta el centroide del objeto cuando la red neuronal no es capaz de detectarlo, asumiendo que si el color es similar debería continuar en la misma posición. En las figuras 4.7 y 4.8 se muestran los resultados totales obtenidos en forma de gráficas, que representan el comportamiento del código para cada uno de esos vídeos.

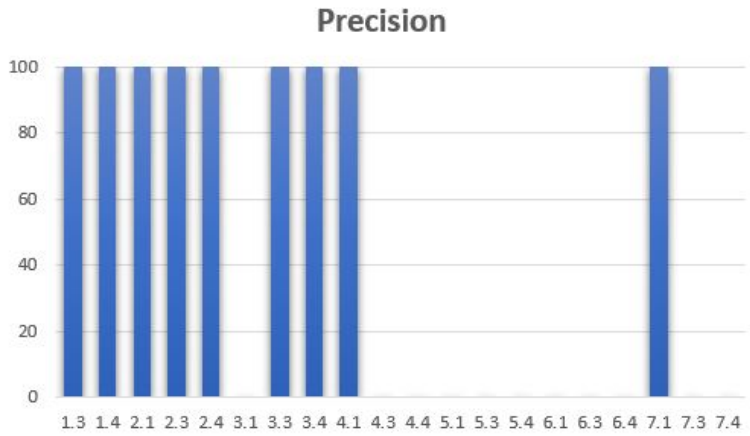


Figura 4.7: Precisión obtenida en el primer intento.

Precision	**62,84 %**

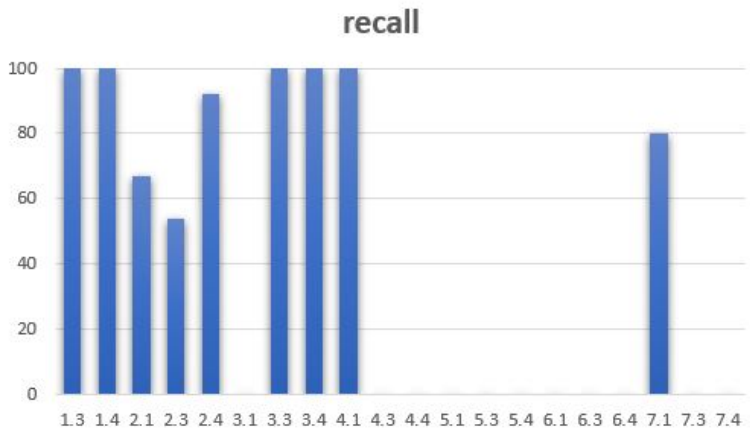


Figura 4.8: Recall obtenido en el primer intento.

Recall	**49,82 %**

En este momento se obtienen unos valores bastante reducidos para los resultados, por debajo de los conseguidos con el primer método. Por este motivo se modifica el código introduciendo el planteamiento explicado anteriormente en el desarrollo. Para que un objeto se considere abandonado debe permanecer estático en sus últimas detecciones, y además, la persona asociada debe alejarse o permanecer fuera de la imagen. Con estas modificaciones se vuelven a obtener resultados mejorando significativamente. En la tabla 4.3 se observan los resultados para cada uno de los vídeos y el total. A continuación, en las imágenes 4.9 y 4.10 aparecen gráficamente la precisión y *recall* de todos los vídeos.

Vídeo.Cámara	TP	TN	FP	Precision(%)	Recall(%)
1.2	987	0	0	100	100
1.3	987	0	0	100	100
2.1	1030	328	0	100	75,85
2.2	1030	422	0	100	70,94
2.3	1030	0	0	100	100
3.1	1	0	0	100	100
3.2	1	930	0	100	0
3.3	1	0	0	100	100
4.1	1	0	0	100	100
4.2	1	0	0	100	100
4.3	1	0	0	100	100
5.1	1	852	1401	0	0
5.2	1	0	1401	0	100
5.3	1401	365	0	100	79,33
6.1	1	80	398	0	0
6.2	1	0	398	0	100
6.3	1	0	398	0	100
7.1	1929	1752	0	100	52,4
7.2	1929	1011	0	100	65,61
7.3	1929	14	0	100	99,28
Total				**75.04 %**	**77.24 %**

Tabla 4.3: Resultados obtenidos para el segundo método.

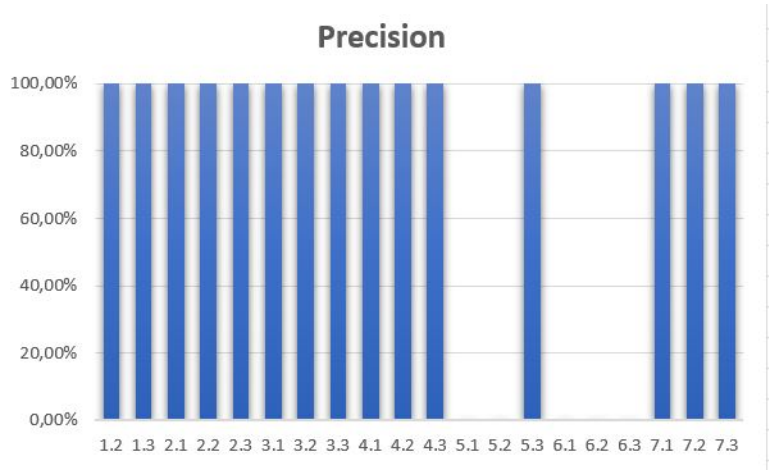


Figura 4.9: Precisión obtenida en el segundo intento.

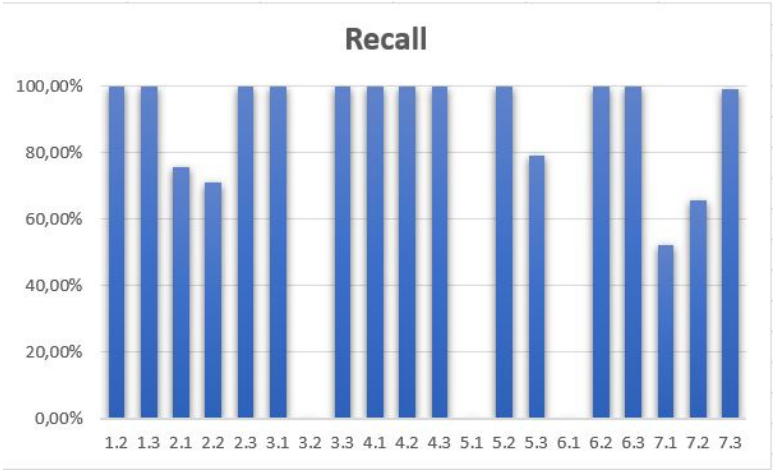


Figura 4.10: Recall obtenido en el segundo intento.

En este caso, se logra mejorar el punto débil del primer método, el *recall*. Además se mantiene la correcta detección de los objetos abandonados en la mayoría de las pruebas. En la figura 4.11 se aprecia la detección de un objeto abandonado, y su aviso visual en la imagen.

Figura 4.11: Ejemplo de detección de abandono de un objeto en la colección de PETS06.

A continuación se comentan particularmente los resultados, explicando concretamente los fallos encontrados en algunos vídeos.

4.4 Discusión

Una vez expuestos todos los resultados obtenidos a partir de ambos métodos de análisis, se analizan de manera particular cada uno de ellos para apreciar los beneficios e inconvenientes que presentan.

En la tabla 4.2 y las gráficas 4.2 y 4.3, se encuentran los resultados obtenidos a partir del primer método, los valores de precisión y exhaustividad de la totalidad de los vídeos, son de 73 % y 49 % respectivamente.

En cuanto a la precisión, el programa realizado es capaz de captar de manera correcta los abandonos realizados, únicamente en 3 vídeos de toda la colección se obtienen valores reducidos. Estos valores se deben a 2 motivos diferentes: Solo se detecta una parte del objeto, por lo que sus coordenadas difieren respecto a las de referencia y no se puede dar por válido el resultado; También puede ocurrir que se detecte el objeto correctamente en algunas imágenes, pero debido a las circunstancias de los vídeos se encuentran personas en las cercanías del objeto que hagan creer al programa que el objeto se encuentra en movimiento.

En relación con el *recall*, se obtiene un valor más reducido. Esto se produce por el principal inconveniente del método, cuando no es capaz de encontrar el fondo completo del vídeo. En estos casos, el fondo almacena alguna forma que no pertenece a éste, y, si se trata de una persona, al desplazarse detectará que el fondo ha cambiado, asumiendo que hay un objeto y al permanecer detenido detectará un falso abandono (Verdadero negativo).

A continuación, se revisan los resultados finales del segundo método, reflejados en la tabla 4.3 y las gráficas 4.9 y 4.10. Después de algunas modificaciones, se consigue obtener un valor del 75 % para la precisión, y un 77 % para el *recall*.

El valor obtenido para la precisión es similar al obtenido con el primer método pero presenta diferencias al analizar particularmente cada vídeo. Se observa en los resultados 2 opciones, o detecta todos los abandonos de un vídeo, o no es capaz de detectarlo. Esto se debe a la realización del código, donde una vez se detecta un abandono, no dejará de estar abandonado hasta que no presente movimiento, aunque el detector no sea capaz de encontrar el objeto, se supone que está detenido. En los casos donde la precisión es nula, se debe a diferentes motivos. En el caso de los vídeos 5.1 y 5.2, aparece una bolsa que el detector no es capaz de incluir en ninguna de las clases elegidas (mochilas, bolsos y maletas), y al no detectar el objeto no se puede identificar el abandono. En el caso del vídeo 6, en todas las cámaras se deja de detectar la persona asociada al objeto antes del abandono, y al evaluar las condiciones de abandono del objeto, se elimina el objeto ya que no se cumplen.

Analizando el valor de la exhaustividad obtenido en esta última prueba, se aprecia un aumento notable. En este párametro reside el principal beneficio del método, ya que sus resultados no dependen de la obtención del fondo de la imagen, reduciendo las detecciones erróneas. Los valores reducidos se producen en algunos de los vídeos donde no se producen detecciones, por lo tanto, y acudiendo a la fórmula 4.2, el numerador será nulo, ya que no se encuentran verdaderos positivos, y en el momento que hay una detección errónea el valor será nulo. En este método, los verdaderos negativos se producen por fallos en la asociación del objeto con una persona, o cuando el detector de YOLO no es capaz de detectar algún objeto.

Capítulo 5

Conclusiones y líneas futuras

En este capítulo, se resumen las conclusiones obtenidas y se proponen futuras líneas de investigación que se deriven del trabajo.

5.1 Conclusiones

En este TFG se han desarrollado y evaluado dos alternativas enfocadas a la detección de objetos abandonados en secuencias de imágenes.

En primer lugar, se ha realizado el estudio teórico necesario para conocer la situación actual, así como las técnicas existentes para abordar el problema con conocimiento.

Seguidamente se han realizado dos propuestas diferentes basadas en alternativas de la literatura. La primera de ellas basada en métodos clásicos para la detección de objetos; al observar su funcionamiento, destaca la capacidad de detección ante cualquier objeto, siempre y cuando se obtenga el fondo de la imagen de forma correcta. Otro de los inconvenientes que presenta es la dependencia de la iluminación, ya que si varía de forma significativa puede detectar objetos donde realmente no están.

Para evitar las desventajas de ésta primera, se ha propuesto la detección de objetos abandonados empleando redes neuronales.

En concreto, se ha utilizado YOLO debido a la capacidad de procesamiento y los resultados que es capaz de obtener.

Ambos métodos se han evaluado de forma exhaustiva empleando el dataset PETS06, que presenta una elevada dificultad debido a la ubicación de las cámaras, la complejidad del entorno y los cambios en la iluminación.

Al comparar los resultados conseguidos por los dos métodos, se aprecia que ambos son capaces de realizar la detección de abandono de una forma correcta, pero el segundo de los métodos realizados consigue mejorar para reducir significativamente el número de detecciones erróneas (verdaderos negativos) que se realizan. Esto es posible gracias a la información obtenida del detector, y en lugar de tener formas en la imagen, se tiene la información del objeto definiendo que solo algunos objetos podrán ser abandonados.

5.2 Líneas futuras

Una vez conocidos las ventajas e inconvenientes que presentan, se indican posibles líneas futuras que tienen como punto de partida este TFG:

- Empleo de otra CNN. Podría implementarse otra red para conseguir detectar los objetos en todo momento. El principal inconveniente es que se elige YOLO para que sea posible la ejecución en tiempo real, y existen redes que logran mejores resultados en detrimento del tiempo de ejecución.
- Entrenamiento de la CNN. Una posibilidad consiste en entrenar la red, en primer lugar, para que únicamente detecte las clases necesarias para la aplicación, lo que reduciría el tiempo de análisis. Por otro lado se podrían añadir más imágenes en el entrenamiento para intentar detectar los objetos de interés en todo momento, aunque no existan personas cerca.
- Unión de ambos métodos. Otro cambio que mejoraría los resultados consistiría en unificar las 2 técnicas empleadas, de esta forma se continuaría detectando los abandonos, pero se consigue reducir los inconvenientes. Por una parte se eliminarían las falsas detecciones producidas por la obtención de fondo, ya que YOLO no sería capaz de detectar nada en esa zona y no se almacenaría información. Por otra parte los problemas al detectar objetos desaparecerían, ya que con los métodos clásicos se podría comprobar si existe algún objeto en la posición almacenada. El inconveniente que puede aparecer, es la ralentización del proceso, pero se podría evitar realizando el análisis cada 2 o más imágenes.

Capítulo 6

Pliego de condiciones

En el capítulo actual se especifican los elementos más importantes empleados para la realización del TFG, indicando sus especificaciones técnicas, y el software utilizado.

6.1 Elementos necesarios para la realización mediante métodos clásicos

Para comenzar, se indican los elementos utilizados para su realización:

1. Ordenador portátil Medion Erazer con las siguientes especifiaciones:
    - Procesador Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
    - 16 GB de memoria RAM DDR4 SDRAM
    - 128 GB de almacenamiento SSD
    - 500 GB de almacenamiento SATA3
    - Sistema operativo Windows 10 de 64 bits
2. Compilador de Latex con Textmaker 5.1.2
3. Matlab 2020b con librerías de *Computer vision*

6.2 Elementos empleados en la realización mediante redes neuronales

Para conseguir el funcionamiento mediante la última versión se emplean los siguientes componentes:

1. Ordenador 'Z370 AORUS Ultra Gaming con las siguientes especificaciones:
    - Procesador Intel(R) Core(TM) i7-8700k CPU @ 3.70GHz
    - Tarjeta gráfica GeForce GTX 1080 TI
    - 32 GB de memoria RAM DIMM DDR4 2666MHz
    - 500 GB de almacenamiento SATA3

- Sistema operativo de 64 bits. Ubuntu 20.04.3

2. Compilador de Latex con Textmaker 5.1.2
3. Plataforma PyCharm 2022.1.3 (Community edition)
4. Lenguaje de programación Python 3.8.10
5. NVIDIA CUDA Toolkit 10.1.243

Capítulo 7

Presupuesto

Para finalizar, se incluye el presupuesto que supone la implantación del sistema desarrollado en un emplazamiento real. Este contiene todos los gastos, incluyendo el software y hardware, así como el coste de la mano de obra necesario para su desarrollo:

1. **Gastos de los recursos software del proyecto**

Concepto	Cantidad	Subtotal (euros)
Licencia PyCharm	1	200
Licencia Matlab	1	2000
Licencia TexMaker	1	0
Licencia Latex	1	0
Total software		**2.200**

2. **Gastos de los recursos hardware del proyecto**

Concepto	Cantidad	Subtotal (euros)
Ordenador personal	1	1800
Cámara de vigilancia	1	400
Total hardware		**2.200**

3. **Gastos de mano de obra**

Concepto	Cantidad(horas)	Coste unitario	Subtotal (euros)
Programación del código	350	65 €/hora	22.750
Realización de documentación	50	15 €/hora	750
Total			**23.500**

4. **Gastos totales**

Concepto	Subtotal
Software	2.200€
Hardware	2.200€
Mano de obra	23.500€
Total	**27.900€**

El importe total del proyecto asciende a la cantidad de : VEINTISIETE MIL NOVECIENTOS EUROS

En Alcalá de Henares, a ___ de _____ de 20___.

Fdo. Pablo Calvo del Castillo

Graduado en Ingeniería Electrónica y Automática Industrial

Bibliografía

[1] "Transformaciones morfológicas en imágenes." https://unipython.com/transformaciones-morfologicas/[Último acceso 28/junio/2022].

[2] "Segmentación de objetos basado en bordes." https://docplayer.es/15596519-Segmentacion-basada-en-bordes.html[Último acceso 24/junio/2022].

[3] "Segmentación de fondo de objetos." https://es.mathworks.com/solutions/image-video-processing/semantic-segmentation.html[Último acceso 24/junio/2022].

[4] "Funcionamiento de las redes neuronales." https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/[Último acceso 25/mayo/2022].

[5] "Redes neuronales convolucionales." https://ia-latam.com/2019/02/06/entendiendo-las-redes-neuronales-de-la-neurona-a-rnn-cnn-y-deep-learning/[Último acceso 25/mayo/2022].

[6] "Estructura empleada en yolov3." https://programmerclick.com/article/85171804594/[Último acceso 26/mayo/2022].

[7] "Yolo, página oficial," https://pjreddie.com/darknet/yolo/[Último acceso 13/marzo/2022].

[8] "Ejemplo de funcionamiento del filtro de kalman." https://www.researchgate.net/profile/Marcos-Fernandes-15/publication/313426875/figure/fig26/AS:459222321242122@1486498541679/Figura-30-Filtro-de-Kalman-com-diferentes-ponderacponderacponderacoes-na-saida-Fonte.png[Último acceso 26/mayo/2022].

[9] D. Liciotti, M. Paolanti, E. Frontoni, and P. Zingaretti, "People detection and tracking from an rgb-d camera in top-view configuration: review of challenges and applications," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 207–218.

[10] H. L. Kidane, "Comparative survey: People detection, tracking and multi-sensor fusion in a video sequence," *arXiv preprint arXiv:1806.06261*, 2018.

[11] I. Ahmed and G. Jeon, "A real-time person tracking system based on siammask network for intelligent video surveillance," *Journal of Real-Time Image Processing*, vol. 18, no. 5, pp. 1803–1814, 2021.

[12] T. Khuc and F. N. Catbas, "Completely contactless structural health monitoring of real-life structures using cameras and computer vision," *Structural Control and Health Monitoring*, vol. 24, no. 1, p. e1852, 2017.

[13] Y. Bao, Z. Tang, H. Li, and Y. Zhang, "Computer vision and deep learning–based data anomaly detection method for structural health monitoring," *Structural Health Monitoring*, vol. 18, no. 2, pp. 401–421, 2019.

[14] S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, "Trajectory-based surveillance analysis: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 7, pp. 1985–1997, 2018.

[15] R. Nayak, U. C. Pati, and S. K. Das, "A comprehensive review on deep learning-based methods for video anomaly detection," *Image and Vision Computing*, vol. 106, p. 104078, 2021.

[16] G. Sreenu and M. S. Durai, "Intelligent video surveillance: a review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–27, 2019.

[17] A. B. Mabrouk and E. Zagrouba, "Abnormal behavior recognition for intelligent video surveillance systems: A review," *Expert Systems with Applications*, vol. 91, pp. 480–491, 2018.

[18] S. Smeureanu and R. T. Ionescu, "Real-time deep learning method for abandoned luggage detection in video," in *2018 26th European Signal Processing Conference (EUSIPCO)*. IEEE, 2018, pp. 1775–1779.

[19] E. Luna, J. C. San Miguel, D. Ortego, and J. M. Martínez, "Abandoned object detection in video-surveillance: survey and comparison," *Sensors*, vol. 18, no. 12, p. 4290, 2018.

[20] J. Mudarra Luján, "Diseño, implementación y evaluación de una estrategia de detección de objetos abandonados en aplicaciones de videovigilancia," 2021.

[21] "Página web del grupo de investigación geintra de la universidad de alcalá," http://www.geintra-uah.org/ [Último acceso 20/diciembre/2021].

[22] "Código de inicio para detección de objetos abandonados en matlab." https://github.com/parasdahiya/abandoned-ObjDetect[Último acceso 10/junio/2022].

[23] "Aplicación de la visión artificil para la léctura de códigos de barras." https://www.bcnvision.es/blog-vision-artificial/herramientas-de-lectura-de-identificacion-industrial/[Último acceso 21/junio/2022].

[24] "Reconocimiento de actividad agrícola mediante detección de objetos en imágenes aéreas." https://digibuo.uniovi.es/dspace/handle/10651/58095?show=full[Último acceso 21/junio/2022].

[25] "Sistema de detección de objetos en el espacio de indra." https://www.minsait.com/es/actualidad/media-room/el-sistema-de-deteccion-de-objetos-en-el-espacio-de-indra-supera-las-pruebas[Último acceso 21/junio/2022].

[26] "You only look once." https://pjreddie.com/darknet/yolo/[Último acceso 3/mayo/2022].

[27] "Técnicas y algoritmos básicos de visión artificial." https://publicaciones.unirioja.es[Último acceso 20/mayo/2022].

[28] "Ejemplos de histograma." https://www.xatakafoto.com/guias/el-histograma-ii[Último acceso 21/mayo/2022].

[29] "Definición y aplicación de las redes neuronales," https://artyco.com/que-son-las-redes-neuronales-y-cual-es-su-aplicacion-en-el-marketing/[Último acceso 25/mayo/2022].

[30] "Algoritmo ssd de detección de objetivos." https://programmerclick.com/article/87881887111/[Último acceso 25/junio/2022].

[31] "Detección de objetos mediante retinanet." https://lamaquinaoraculo.com/computacion/deteccion-de-objetos-2-retinanet/[Último acceso 25/junio/2022].

[32] "Detección de objetos mediante facebook detr." https://huggingface.co/facebook/detr-resnet-50[Último acceso 25/junio/2022].

[33] "Funcionamiento y composición de yolo," https://es.wikipedia.org/wiki/Algoritmo_You_Only_Look_Once_(YOLO)[Último acceso 13/marzo/2022].

[34] "Yolo, detección de imágenes en python," https://www.aprendemachinelearning.com/deteccion-de-objetos-con-python-yolo-keras-tutorial/[Último acceso 13/marzo/2022].

[35] "Introducción de coco y sus aplicaciones," https://towardsdatascience.com/getting-started-with-coco-dataset-82def99fa0b8[Último acceso 13/marzo/2022].

[36] "Datos almacenados en coco," https://ichi.pro/es/seleccionar-y-preparar-un-subconjunto-especifico-de-imagenes-del- acceso 13/marzo/2022].

[37] "Aplicación del filtro de kalman en gps," https://cosasdeingenierossite.wordpress.com/2017/06/12/filtro-de-kalman-fusion-sensorial-de-acelerometros-y-gps/[Último acceso 23/abril/2022].

[38] "Por qué programar en python." https://blog.enzymeadvisinggroup.com/redes-neuronales-python[Último acceso 2/mayo/2022].

[39] "Colección de vídeos de pets2006." https://drive.google.com/drive/folders/1F3xAgdcmCZNEWpJY_HQkWcQneWvMg7xo[Último acceso 29/mayo/2022].

[40] "Página oficial de microsoft. descubre windows." https://www.microsoft.com/es-es/windows/[Último acceso 24/junio/2022].

[41] "Página de la aplicación matlab r2020b." https://es.mathworks.com/products/new_products/release2020b.html[Último acceso 24/junio/2022].

[42] "Computer vision toolbox." https://es.mathworks.com/products/computer-vision.html[Último acceso 24/junio/2022].

[43] "Información sobre gnu/linux en wikipedia," http://es.wikipedia.org/wiki/GNU/Linux [Último acceso 4/mayo/2021].

[44] "Página de la aplicación pycharm." https://www.jetbrains.com/es-es/pycharm/[Último acceso 24/junio/2022].

[45] "Página de la aplicación make," http://savannah.gnu.org/projects/make/ [Último acceso 4/mayo/2021].

[46] "Página de opencv." https://opencv.org/[Último acceso 24/junio/2022].

[47] L. Lamport, *LaTeX: A Document Preparation System, 2nd edition.* Addison Wesley Professional, 1994.

[48] "Página de microsoft excel." https://www.microsoft.com/es-es/microsoft-365/excel[Último acceso 24/junio/2022].

Apéndice A

Herramientas y recursos

Las herramientas necesarias para la elaboración se dividen según el método que se realiza. Los recursos necesarios para el primer método son los siguientes:

- PC compatible
- Sistema operativo Windows 10 [40]
- Entorno de desarrollo Matlab R2020b [41]
- Librerías de visión de Matlab [42]

Las herramientas necesarias para el segundo método son las siguientes:

- PC compatible
- Sistema operativo GNU/Linux [43]
- Entorno de desarrollo PyCharm [44]
- Gestor de compilaciones make [45]
- Librerías de OpenCV [46]

Para ambos métodos se utilizan las siguientes plataformas comunes:

- Procesador de textos LaTeX[47]
- Hoja de cálculos de Excel [48]

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá