ugr | Universidad de Granada

Trabajo de Fin de Máster

Máster Universitario en Ciencia de Datos e Ingeniería de Computadores

Sistema inteligente de videovigilancia para instalaciones críticas

---

Autor:

Juan Ignacio Isern Ghosn

Directores:

Francisco Barranco Expósito

Eduardo Ros Vidal

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

ETSIIT
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, septiembre 2019

Sistema inteligente de videovigilancia para instalaciones críticas

D. Juan Ignacio Isern Ghosn

Resumen: La degradación o el malfuncionamiento involuntario o deliberado de una instalación crítica puede tener un alto coste tanto coste a nivel económico como social, de ahí que haya que garantizar la integridad de este tipo de instalaciones de irrupciones externas o acciones que puedan ser prevenidas. Por su parte, los sistemas de videovigilancia clásicos tienen su eslabón más débil en el operador humano, que se encarga de revisar en ocasiones hasta decenas de distintos vídeos procedentes de cada una de las cámaras del circuito, estando el rendimiento del sistema de videovigilancia supeditado a la capacidad de atención del operador. Así, en este trabajo se propone un sistema de videovigilancia que pueda tanto detectar intrusiones en una instalación crítica como monitorizar el posicionamiento de los operarios que en ella trabajan. Para ello, se hace uso de aprendizaje profundo (*Deep Learning)* tanto para la detección como para la re-identificación de los sujetos humanos. Del mismo modo, se utilizan técnicas clásicas de visión por computador como la extracción de fondos para la selección de áreas de interés a analizar o filtros para la predicción de la trayectoria de los sujetos identificados. También se aplican transformaciones de planos para obtener la situación real de cada uno de los sujetos dentro de la instalación, a fin de salvaguardar los perímetros cuyo acceso no les son permitidos.

Palabras claves: Videovigilancia, instalaciones críticas, aprendizaje profundo, re-identificación, monitorización de perímetros.

Critical infrastructure smart video-surveillance system

D. Juan Ignacio Isern Ghosn

Abstract: The degradation or the involuntary or deliberate malfunctioning of a critical infrastructure can have a high cost, both at an economic and a social level, which is why it is necessary to guarantee the integrity of this type of installations from external intrusions or actions that can be prevented. On the other hand, traditional video surveillance systems have their weakest link in the human operator, who is responsible for reviewing dozens of different videos from each of the cameras in the circuit, with the performance of the video surveillance system depending on the operator's attention span. Thus, this work proposes a video surveillance system that can both detect intrusions in a critical installation and monitor the positioning of the operators working in it. For this purpose, Deep Learning techniques are used for both the detection and re-identification of human subjects. At the same time, classic computer vision techniques are used, such as the background subtraction for the selection of areas of interest to be analyzed, filters for the prediction of the trajectory of the identified subjects. Also, it is used plane transformations to obtain the real situation of each of the subjects within the installation, in order to safeguard the perimeters whose access is not allowed.

Keywords: Videosurveillance, Critical Infrastructures, Deep Learning, Re-identification, Perimeter Monitoring.

Yo Juan Ignacio Isern Ghosn, alumno de la titulación Máster Universitario en Ciencia de Datos e Ingeniería de Computadores de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 45354297E, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Máster en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

ISERN GHOSN JUAN IGNACIO - 45354297E

Firmado digitalmente por ISERN GHOSN JUAN IGNACIO - 45354297E
Fecha: 2019.09.11 19:32:54 +01'00'

Fdo.: Juan Ignacio Isern Ghosn

Granada a 11 de septiembre de 2019

Agradecimientos

A mis padres y mi hermana. Gracias por apoyarme, por permitir que me pueda dedicar a mi vocación y hacerme sentir afortunado de ser quien soy. Mis méritos son y serán siempre tan míos como de ustedes.

A mi amigo Daniel. Para dos chicos de isla como nosotros cualquier experiencia supone una aventura. Ha sido un placer descubrir Granada junto a ti y nuestra convivencia ha resultado ser un desarrollo personal y profesional constante.

A mis dos tutores, Francisco y Eduardo. Gracias por haber depositado vuestra confianza en mí y haber demostrado una gran calidad tanto profesional como humana.

Gracias a los que estuvieron y están, como a los que se fueron y no volverán. Porque todos son parte de mí e indirectamente, de este trabajo.

Índice de contenidos

Índice de tablas

Índice de figuras

1 Introducción

En este primer epígrafe del documento, se presenta el problema original y las motivaciones que han inspirado la realización del presente trabajo. Tras ello se exponen los objetivos concretos que se persiguen y que han sido establecidos a priori. Del mismo modo, también quedan en este punto recogidas las competencias cubiertas tras la realización del proyecto, la distribución del trabajo y la planificación de este y, finalmente, la estructura de la memoria que ante usted se presenta.

1.1 Presentación del problema y motivación

Recientemente, los sistemas de videovigilancia están adquiriendo una mayor importancia. Desde instituciones como el gobierno hasta comunidades residenciales utilizan estos sistemas para garantizar la seguridad de áreas sensibles (Rai, Asim Husain, Maity, & Kumar Yadav, 2019). Algunas de estas áreas sensibles se conocen habitualmente como instalaciones críticas. Estas comprenden aquellas instalaciones, habitualmente públicas, necesarias para la vida de la sociedad, comprendiendo desde instalaciones y redes de suministros, hasta edificios públicos o infraestructuras de servicios de emergencia (Gupta, Agarwal, & Goyal, 2014). Habitualmente, los sistemas de vigilancia de estas instalaciones dependían principalmente de los operadores humanos que las supervisaban, quedando estas sujetas a la capacidad su capacidad de atención (Troscianko et al., 2004). Así, en los últimos años se han preferido los sistemas automatizados debido a su mayor eficiencia y fiabilidad (Aldasouqi & Hassan, 2010).

Por su parte, el desarrollo de las tecnologías de procesamiento gráfico en los últimos años ha resultado en una mayor complejidad de las técnicas de aprendizaje profundo y en la mejora de sus resultados. Así, en los últimos años se han desarrollado nuevos enfoques en el aprendizaje profundo que han supuesto una mejoría en el campo de la detección de personas (Mateus, Ribeiro, Miraldo, & Nascimento, 2019). Del mismo modo, otras tareas como el *tracking* de múltiples objetos (MOT) (C. Zhang, Huang, Wang, Jiang, & Yan, 2018) o la re-identificación de personas (Ni, Ding, Chen, & Wang, 2018) están siendo líneas de investigación muy populares en los últimos años, estando en muchas

ocasiones aplicadas a tareas comprendidas dentro de la videovigilancia inteligente. Así, esto permite que se esté dando un contexto ideal para el desarrollo de este tipo de sistemas.

Por todo esto, este trabajo presenta un sistema íntegro de videovigilancia inteligente, que automatiza tareas como la detección y el seguimiento de uno o múltiples individuos presentes en escena, así como la ubicación de estos sobre el plano de la instalación que se encuentra monitorizada.

1.2 Objetivos del trabajo

El objetivo principal de este trabajo es la implementación de un sistema de videovigilancia para instalaciones críticas en tiempo real, capaz de llevar a cabo la detección, el seguimiento y posicionamiento de individuos humanos tanto en la escena que se analiza, como su situación específica dentro de la instalación. Los objetivos específicos para el desarrollo de este proyecto son:

- Llevar a cabo un **estudio del estado del arte de las técnicas existentes en cuanto a vigilancia de infraestructuras críticas**, haciendo hincapié en aquellas que supongan tratamiento del vídeo de grabaciones de instalaciones.
- Realizar un **estudio del estado del arte** de las técnicas existentes para **detección y seguimiento de personas** y valorar su aplicabilidad en un problema de videovigilancia de estas características.
- **Analizar el funcionamiento** de cada una **de estas técnicas** de acuerdo con las **métricas** que arrojan y el **coste computacional** que suponen.
- **Diseñar y desarrollar una solución software** íntegra de videovigilancia de reducido coste computacional, que funcione en **tiempo real**.
- Valorar la **futura integración** del sistema implementado en un **sistema embebido**.
- De acuerdo con las métricas de calidad de cada uno de los módulos implementados del sistema, **valorar la posible aplicación** de este sobre una **instalación crítica real**.

1.3 Distribución y planificación del trabajo

Para llevar a cabo el desarrollo del presente proyecto se efectuó previamente una **distribución de la carga de trabajo en tareas concretas**, para las cuales se estimó en un primer momento una cierta duración. Del mismo modo y una vez finalizada la realización de este trabajo, se han cotejado las estimaciones con las duraciones reales que se han dado para cada una de las tareas, a fin de detectar desviaciones en la temporalización. A continuación, en la Tabla 1 se muestran cada una **de las distintas tareas y sus duraciones, tanto estimadas como reales**.

Tabla 1 Temporalización del trabajo

Fases	Duración estimada (horas)	Duración real (horas)	Tareas
Estudio previo / Análisis del estado del arte	50	70	Tarea 1.1: Estudio bibliográfico de la seguridad en las instalaciones críticas.
			Tarea 1.2: Estudio bibliográfico de la detección de personas.
			Tarea 1.3: Estudio bibliográfico del análisis de regiones de interés (ROI).
			Tarea 1.4: Estudio bibliográfico del *tracking* de personas.
			Tarea 1.5: Estudio bibliográfico de la re-identificación de personas.
			Tarea 1.6: Estudio de la bibliografía para verificar la relevancia del proyecto.
			Tarea 1.7: Análisis de las herramientas informáticas a utilizar en el trabajo.
Diseño / Desarrollo / Implementación	100	130	Tarea 2.1: Establecimiento de requisitos del sistema.
			Tarea 2.2: Pruebas de concepto.
			Tarea 2.3: Diseño del sistema.
			Tarea 2.4: Implementación.
Evaluación / Validación / Prueba	50	60	Tarea 3.1: Test de rendimiento.
			Tarea 3.1: Test sobre escenas complejas.

			Tarea 3.2: Estudio de posibles mejoras.
Documentación / Presentación	50	65	Tarea 4.1: Elaboración de la memoria.
			Tarea 4.2: Preparación de la presentación.

En general, la duración real de cada una de las fases del proyecto se ha correspondido a las estimaciones iniciales efectuadas. Las desviaciones que se han dado han sido debidas en buena medida a la voluntad del autor del proyecto en cuanto al perfeccionamiento de muchos aspectos de este.

La fase del **estudio previo del estado del arte** de la temática tratada y del análisis de las herramientas informáticas a emplear en el trabajo **fue la que tuvo una mayor desviación** entre su duración estimada y real. En un primer momento, se pensó en el problema tratado en este trabajo como una temática individual, con línea de investigación propia y con un cierto recorrido y, por ende, con un estado del arte propio. Posteriormente, se advirtió que lejos de esto, en resumidas ocasiones se ha abordado el problema de la videovigilancia centrado en las instalaciones críticas. Por ello, fue necesario analizar por una parte la teoría de la seguridad de instalaciones críticas que sustenta este trabajo y por otra, las distintas aproximaciones técnicas que se han dado en problemas de similares características.

En cuanto a la **fase de diseño, desarrollo e implementación**, las principales desviaciones existentes en cuanto a la temporalidad se deben **a la cantidad de pruebas de concepto que se han realizado**. Se han uso de una gran diversidad de técnicas y algoritmos para cada uno de los módulos del sistema. Así, se ha podido comprobar aquellas técnicas que mejor se adecuen a las características del sistema y a los objetivos previamente establecidos para el proyecto.

El tiempo para el desarrollo de la fase de **evaluación, validación y prueba** ha sido **menor del esperado**, debido en buena medida a la implementación y automatización que se hubo realizado previamente de una gran parte de las tareas que en ella se ejecutan.

1.4 Estructura del documento

Este documento se corresponde con la memoria del Trabajo de Fin de Máster de mismo título. Esta memoria se encarga de documentar y justificar la metodología seguida durante el desarrollo de la totalidad del trabajo, desde que se planteara el problema que ha servido de motivación para la realización de este, hasta la obtención e interpretación de los resultados. Así, el presente documento ha sido distribuido en 6 epígrafes principales, a parte del correspondiente a las referencias bibliográficas.

Tras este epígrafe introductorio, se encuentran los **antecedentes**. En este apartado se expone el estado del arte del problema abordado, incidiendo en aquellos conceptos que han de resultar conocidos para el correcto entendimiento de la solución planteada en el trabajo. Por lo tanto, se presenta en este epígrafe una revisión de los conceptos de: La seguridad de las instalaciones críticas, la detección y el *tracking* de personas, la re-identificación de personas y la selección de áreas de interés de la imagen.

El tercer epígrafe por su parte trata los **recursos y tecnologías** que han sido necesario emplear para el desarrollo del proyecto. Entre ellos se puede encontrar el lenguaje de programación elegido y los paquetes o librerías empleados, los conjuntos de datos que han sido utilizados y los recursos hardware sobres los que se ha llevado a cabo el desarrollo de las utilidades y la memoria del presente trabajo.

En el cuarto apartado se tratan aquellos aspectos relevantes del proceso metodológico seguido durante el diseño y la implementación de la utilidad software. Así, en el apartado de **metodología** se presenta cual ha sido el planteamiento seguido en el desarrollo y se explican cada uno de los módulos del programa. En él, se hace hincapié en la utilidad de cada una de las funciones implementadas que integran cada uno de los módulos y su alineación con la consecución de los objetivos establecidos para este proyecto.

El quinto apartado se corresponde con el de **resultados**. En este epígrafe se incluyen las distintas métricas de rendimiento que han sido consideradas y el valor obtenido en las pruebas sobre cada uno de los módulos del sistema. Del

mismo modo, en este punto se incluyen ejemplos de funcionamiento del sistema en diversas situaciones, pudiendo así observar el desempeño del programa en distintas ubicaciones, como la presencia de diferente número de personas en escena o con diversidad de objetos en movimiento.

Por último, en el epígrafe de las **conclusiones** se lleva a cabo una revisión del proceso seguido durante el proyecto, a fin de comprobar la adecuación entre los objetivos inicialmente propuestos y los resultados obtenidos. Además, se exponen aquellos aspectos que requieren de revisión, mejora o simplemente que dan lugar a líneas de investigación que seguir en un futuro en caso de que se quisiera continuar el proyecto en el punto en el que se encuentra actualmente.

Para la corrección de este trabajo, se aconseja a los miembros del tribunal que, a fin de tener una total comprensión del proyecto y la problemática que abarca, se lleve a cabo lectura completa del documento.

2 Antecedentes

En este epígrafe de la memoria se lleva a cabo una síntesis del estado del arte actual de cada uno de los conceptos que han sido estudiados y que resultan necesarios para poder entender el trabajo y cada uno de los aspectos metodológicos que pueden resultar

2.1 Seguridad de infraestructuras críticas

Supervisar automáticamente infraestructuras críticas es un reto de gran importancia que abordar en los últimos años (Turchini, Seidenari, Uricchio, & Del Bimbo, 2018). Se define como **infraestructuras críticas aquellos recursos que se consideran esenciales para el mantenimiento de la sociedad** (Yusta, Correa, & Lacal-Arántegui, 2011). Cualquier sistema de una infraestructura crítica representa una enorme inversión pública. Incluso **una interrupción menor**, causada de forma involuntaria o deliberada, puede degradar el rendimiento del sistema e **infligir unas pérdidas económico-sociales muy significativas** (Brown, Carlyle, Salmerón, & Wood, 2006).

Cada país tiene un conjunto de infraestructuras críticas y estas varían de una nación a otra. Así, se pueden encontrar infraestructuras críticas dentro de los siguientes sectores (Gupta et al., 2014):

- Energía (incluyendo petróleo, gas natural y energía eléctrica)
- Banca y finanzas
- Transporte (Incluyendo transporte aéreo, de superficie y acuático)
- Tecnologías de la Información y la Comunicación (TIC)
- Sistemas de agua
- Servicios de emergencia gubernamentales y privados
- Instalaciones que producen, utilizan, almacenan o eliminan materiales nucleares

La automatización de la vigilancia de infraestructuras críticas se presenta como el paso lógico, pues **muchas áreas necesitan una vigilancia constante** para garantizar el cumplimiento de las normas y leyes, y la disminución del coste de los sensores favorece el despliegue de circuitos de cámaras de red (Turchini et

al., 2018). Actualmente, **los sistemas de vigilancia existentes se basan principalmente en el rendimiento de los operadores humanos**, los cuales en ocasiones deben observar un gran número de pantallas con el vídeo capturado por distintitas cámaras (Troscianko et al., 2004). Es sabido que, tras veinte minutos de trabajo continuo, **la atención del operador disminuye significativamente** suponiendo una merma del rendimiento del sistema de vigilancia en su totalidad. Las tareas típicas de los operadores incluyen la comprobación de la presencia o ausencia de personas y objetos en las zonas de seguridad, la garantía de que se respeta la capacidad máxima de un lugar, o la comprobación de eventos anómalos, como objetos colocados en lugares inesperados o personas en zonas prohibidas (Haering, Venetianer, & Lipton, 2008). Así, a fin de reducir la carga mental y aumentar la capacidad de atención, es deseable cualquier método automático que sea capaz de alertar de forma fiable y en tiempo real al operador de la presencia de personas, objetos o de cualquier situación anómala.

2.2 Detección de personas

La detección de personas (o detección de peatones, debido a su uso más tradicional en el campo de los sistemas de ayuda a la conducción) es una tarea muy importante en la visión por computador y tiene un gran potencial de aplicación en muchos campos. Debido a las **múltiples vistas**, las **diferentes iluminaciones**, las **múltiples escalas** y la **oclusión parcial de los objetivos** a detectar, supone un **gran reto** (Fang, Chen, Lu, & Hu, 2018). Así, en los últimos años se han realizado diversos esfuerzos para mejorar el rendimiento de la detección de peatones (S. Zhang, Benenson, Omran, Hosang, & Schiele, 2018). Para hacer frente a estos desafíos, existen distintas metodologías para resolver el problema de la detección de personas en una imagen.

2.2.1 Extracción de descriptores para la detección de personas

La metodología más habitual en los sistemas actuales la mostrada en la Ilustración 1, correspondiente a la **extracción de descriptores de la imagen**. Esta metodología clásica, considera que los descriptores de la imagen deben capturar la **información más discriminatoria de los peatones**, de forma que un algoritmo de aprendizaje automático pueda discernir las imágenes que

contienen un humano de las que no. Dentro de la diversidad de descriptores existentes en la literatura, los más conocidos son los del tipo *Haar* (Viola, Jones, & Snow, 2005), *Scale-invariant feature transform* (SIFT) (Lowe, 2004), *Histogram of oriented gradients* (HOG) (Dalal & Triggs, 2005), *Local binary patterns* (LBP) (Ojala, Pietikäinen, & Harwood, 1996), etc.

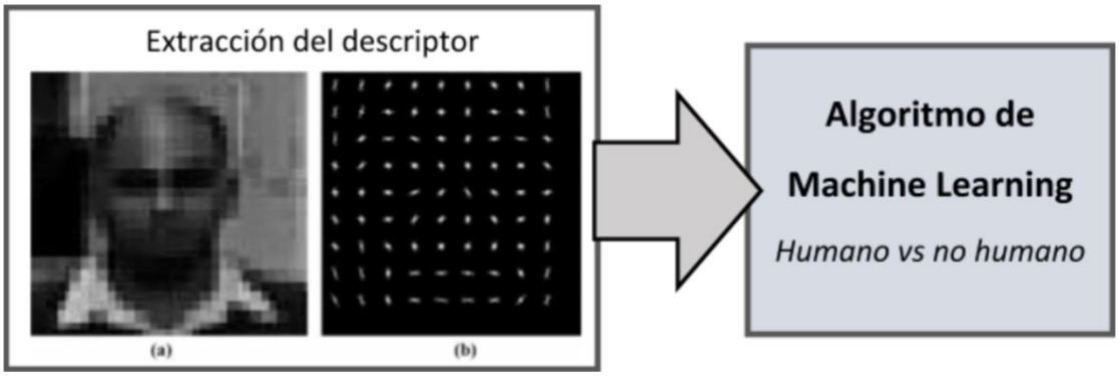

Ilustración 1 Detección de personas mediante descriptores
Fuente: Basado en Dadi, Pillutla, & Makkena (2018)

En la Ilustración 1, se muestra la imagen correspondiente a la persona (a) y su descriptor HOG asociado (b). El descriptor HOG se basa en la idea de que la **apariencia y la forma del objeto** local (humano o no) **pueden caracterizarse** a menudo teniendo en cuenta la **intensidad de los gradientes** o la distribución de las direcciones de los bordes. Cada fotograma de vídeo se divide en pequeñas regiones y se calcula un histograma 1-D local de las orientaciones de los bordes sobre los píxeles del bloque. **Cada histograma generado se considera una representación de imagen**. Así se utilizan clasificadores tales a máquinas de soporte vectorial (*Support Vector Machine* - SVM), redes neuronales artificiales (*Artificial Neural Networks* – ANN) y algoritmos de *boosting* para discriminar cada subregión e imagen (Brunetti, Buongiorno, Trotta, & Bevilacqua, 2018).

Así, si bien existen soluciones que mejoran el desempeño de los descriptores clásicos, el **alto requisito de potencia de procesamiento** de técnicas como el aprendizaje profundo, suponen una barrera para su adopción. Por esta razón y por la necesidad de procesamiento en tiempo real, se han propuesto nuevos planteamientos de descriptores para la detección en tiempo real de personas

(Benenson, Mathias, Timofte, & Van Gool, 2012; Bilal & Hanif, 2019; Quinton & Se, 2018).

2.2.1.1 Máquina de soporte vectorial

Una Máquina de Soporte Vectorial (SVM) es un algoritmo de clasificación introducido por Boser, Guyon, & Vapnik (1992). Por sus sólidos fundamentos teóricos, el clasificador SVM ha sido ampliamente utilizado en diversas áreas. Las principales características de este algoritmo es que normalmente es muy preciso y capaz de calcular y procesar datos de alta dimensionalidad. Intuitivamente el funcionamiento de una SVM es fácilmente comprensible. Una SVM **mapea los puntos de entrada a un espacio de una dimensión mayor** y encuentra un **hiperplano** (un espacio de tamaño $n - 1$) que **separa y maximiza el margen m entre las clases en este espacio**. En la Ilustración 2 se muestra un ejemplo de una SVM para clasificación binaria, con el hiperplano que separa las dos clases en negro.

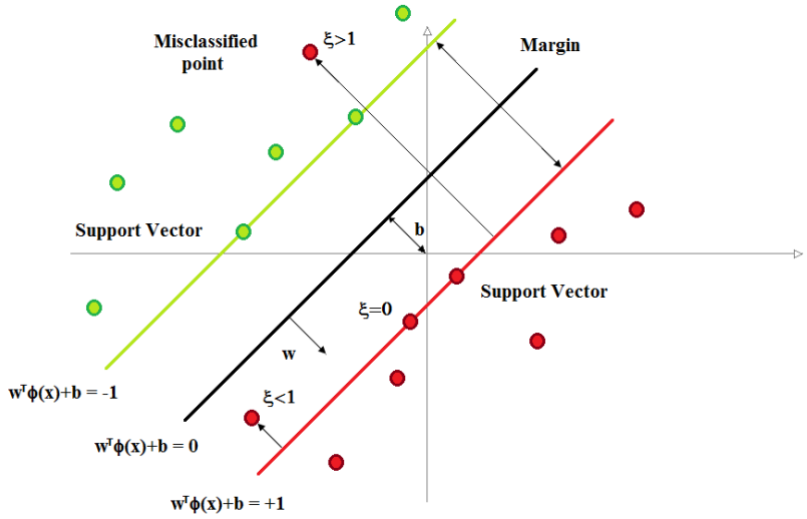

Ilustración 2 Ejemplo de SVM
Fuente: Sforza & Lippi, (2013)

Aunque el SVM se basa en un discriminador lineal, no se limita a realizar hipótesis lineales. Así, le es posible tomar decisiones no lineales mediante un mapeo no lineal de los datos a un espacio dimensional superior. La SVM es una formulación matemática no paramétrica que permite aplicar estas transformaciones de manera eficiente e implícita: el objetivo de una SVM es una función del producto escalar entre pares de vectores; la sustitución de los productos escalares originales por los calculados en otro espacio elimina la

necesidad de transformar explícitamente los puntos de los datos originales al espacio superior. El cálculo de este producto escalar entre vectores sin mapear explícitamente a otro espacio se realiza por medio de una función de kernel. Normalmente, se usan una gran variedad de funciones de kernel, siendo las más comunes la **lineal**, la **polinomial**, la **gaussiano** y la **sigmoidal** (Dalal & Triggs, 2005).

2.2.2 Aprendizaje profundo para la detección de personas

El desarrollo de las tecnologías de procesamiento gráfico en los últimos años y la disminución del precio de las GPUs han resultado en una **mayor complejidad de las técnicas de aprendizaje profundo** y en la **mejora de sus resultados**. Desde 2012, se han desarrollado nuevos enfoques basados en técnicas de aprendizaje profundo que han supuesto una mejoría en el campo de la detección de peatones, especialmente desde la llegada de AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), red entrenada con el conjunto de datos de **ImageNet** (Jia Deng et al., 2009), compuesto actualmente por más de **14 millones de imágenes** con los objetos en escena anotados, discriminando entre **más de 20.000 categorías de objetos** distintos.

2.2.2.1 Redes neuronales convolucionales (CNN)

Las redes neuronales convolucionales (*Convolutional Neural Network* - CNN) son un tipo especializados de redes neuronales que se utilizaron originalmente en **aplicaciones de procesamiento de imágenes**, si bien hoy en día también se utilizan con gran éxito en aplicaciones de procesamiento de vídeo y lenguaje natural. La principal fortaleza de una red neuronales convolucionales (*Convolutional Neural Network* - CNN) es que es capaz de extraer **información del contenido del píxel** y de **aprender características automáticamente** sin necesidad, como ocurre en el resto de métodos manuales tales a descriptores de imagen, de llevar a cabo una programación explícita sobre la red (Tomè et al., 2016).

La estructura de una red convolucional se compone típicamente de tres tipos diferentes de capas: Convolucional, de agrupación (*Pooling*) o totalmente conectada (*fully connected*). Cada tipo de capa tiene reglas diferentes para la

propagación hacia adelante y hacia atrás de errores que les permitan entrenarse. No existen reglas precisas sobre cómo debe organizarse la estructura de las distintas capas. Sin embargo, las CNN suelen estar estructuradas en dos partes. La primera parte, normalmente llamada extracción de características, consiste en utilizar combinaciones de capas convolucionales y de agrupación. La segunda parte llamada clasificación es el uso de capas completamente conectadas (Hui-huang Zhao & Liu, 2019). En la Ilustración 3 se muestra un ejemplo de estructura de CNN para el problema de la clasificación de caracteres manuscritos.

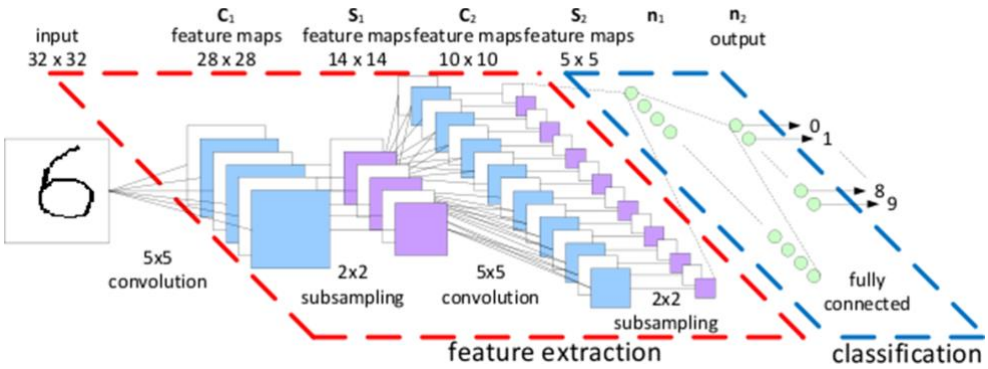

Ilustración 3 Estructura de una CNN
Fuente: Hui-huang Zhao & Liu, (2019)

Así, una arquitectura de red neuronal profunda se utiliza de múltiples capas, que son capaces de extraer características, tales como **bordes** o **patrones** en las **imágenes** y utilizar estas **características para clasificar un objeto**, pudiéndose, por lo tanto, entrenar estas redes para por ejemplo, reconocer peatones (Ren, He, Girshick, & Sun, 2017; Yang, Nguyen, San, Li, & Krishnaswamy, 2015).

En este trabajo se hace uso de aprendizaje automático para llevar a cabo la detección de personas, teniendo siempre en vista el requisito fundamental del funcionamiento en tiempo real del sistema, característica crítica en un sistema de vigilancia. Así, en nuestro caso se adopta una solución híbrida conectando en cascada un método más rápido que propone regiones de interés (ROI) prometedoras para las ubicaciones de los peatones (Mateus et al., 2019), cuya clasificación es refinada por la red neuronal convolucional, por lo que se mejora la precisión al eliminar los falsos positivos.

2.2.2.2 Transfer Learning

Debido al **gran número de parámetros de una CNN**, para **reducir el sobreajuste** durante el entrenamiento de un modelo, debe utilizarse un **conjunto de datos de tamaño sustancial y representativo** del problema a abordar (Krizhevsky et al., 2012). Sin embargo, en el contexto de la detección de peatones, no se dispone públicamente de un conjunto de datos anotado de unas dimensiones suficientes como para abordar directamente el problema. Sin embargo, aquellos modelos que como AlexNet, han sido entrenados con ImageNet han demostrado ser un buen punto de partida para realizar tareas diferentes a la clasificación de objetos. Así, en los últimos años esta limitación aparente se ha resuelto por medio del *Transfer Learning*, **utilizando los parámetros de un modelo CNN entrenado con un conjunto de datos perteneciente a otro problema distinto**. De forma que solo es necesario llevar a cabo un refinamiento del modelo con el conjunto de datos del problema que nos concierne, para especificar el funcionamiento de la CNN sobre el problema en cuestión (Weiss, Khoshgoftaar, & Wang, 2016).

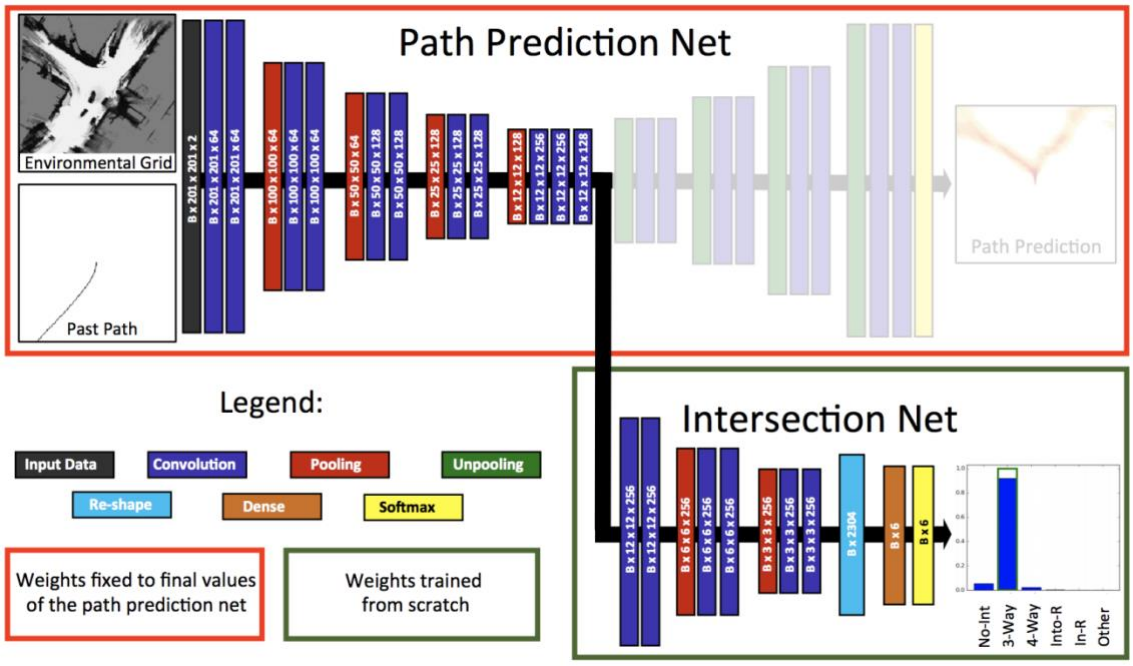

Ilustración 4 Transfer Learning en la clasificación de intersecciones
Fuente: Baumann et al., (2018)

Un ejemplo de aplicación de Transfer Learning al problema de la clasificación de intersecciones de carreteras (Baumann et al., 2018) se puede observar en la

Ilustración 4. En el caso específico de nuestro trabajo se considera tanto la elección y composición de un conjunto de datos grande y diverso, como la utilización de Transfer Learning para llevar a cabo el entrenamiento del detector de humanos.

2.3 Análisis de regiones de interés (ROIs): Extracción de fondos

Tal y como se anticipó anteriormente, a fin de mejorar el rendimiento de nuestro sistema de detección para cumplir el criterio del funcionamiento en tiempo real, se utiliza primeramente un método que identifica **áreas de la imagen donde hay más probabilidad de encontrar a las personas** (regiones de interés), para posteriormente discernir por medio del uso del modelo CNN en qué regiones se encuentran las personas.

Así, **las regiones de interés (*Regions of interest* – ROIs)**, también conocidas como regiones propuestas, se considera **el primer y más importante paso un sistema de detección** (Ren et al., 2017). En esta etapa se aplican algunas técnicas de procesamiento de imágenes para facilitar la búsqueda de ROIs (Brunetti et al., 2018). El enfoque habitual para **la detección y seguimiento de objetos en movimiento** en un flujo de vídeo obtenido por una cámara estática consiste en la **extracción del fondo** (*Background Substraction* - BS). Esta técnica permite la identificación de objetos en movimiento dentro de una escena generando un modelo del fondo de la imagen (Piccardi, 2004). Aunque los algoritmos basados en extracción de fondo son bastante sencillos de implementar, estos no son robustos en cuanto a la variabilidad de la iluminación, un fondo que no permanece estático, sombras o el ruido, lo que limita su uso principalmente en entornos controlados (Brutzer, Hoferlin, & Heidemann, 2011). En la Ilustración 5 se observa un ejemplo de la aplicación de extracción de fondos sobre el fotograma de uno de los vídeos.

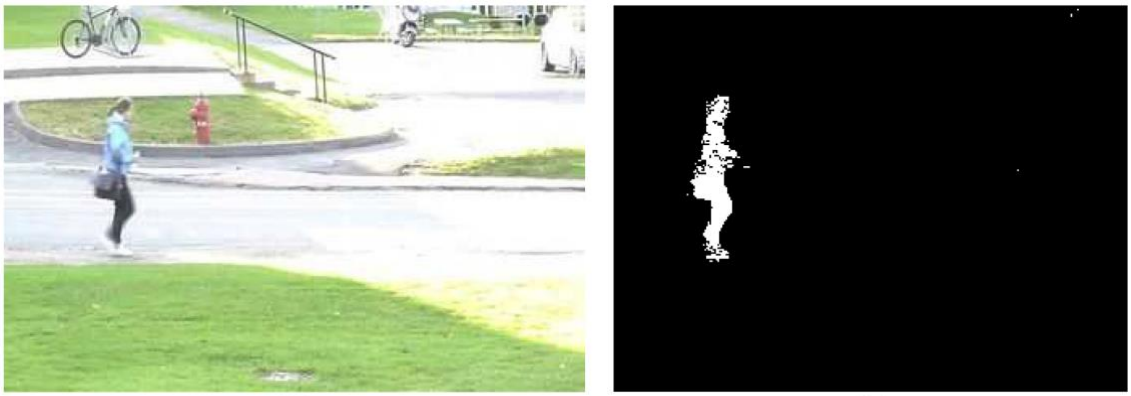

Ilustración 5 Ejemplo de aplicación de extracción de fondo
Fuente: Chiu, Chiu, & Xu, (2018)

En la literatura se han propuesto diversos métodos para realizar la extracción del fondo. Todos estos métodos **tratan de estimar el modelo del fondo a partir de la secuencia temporal de los fotogramas de un flujo de vídeo** (Piccardi, 2004). Los algoritmos más populares para la extracción de fondos que se encuentran en la literatura son el modelo de mezcla de gaussianas (*Mixture of Gaussians* - MOG) (Stauffer & Grimson, 1999; Zivkovic & Van Der Heijden, 2006) y el modelo *codebook* (Kim, Chalidabhongse, Harwood, & Davis, 2005). La metodología MOG modela el historial de cada píxel como un conjunto de distribuciones gaussianas y utiliza un sistema para ir actualizando sus parámetros en tiempo real. Esta metodología va mejorando en términos de rendimiento al considerar ecuaciones recursivas que adaptativamente actualizan los parámetros del modelo gaussiano (Zivkovic & Van Der Heijden, 2006). En cuanto al modelo *codebook* (Kim et al., 2005), los valores de cada uno de los píxeles del fondo son discretizados en grupos de similar color, de forma que se almacena una forma comprimida del fondo para una secuencia de imágenes. Esta metodología constantemente se enriquece con nuevas palabras clave (*codewords*) en presencia de un nuevo color que no puede ser asignado a los grupos preexistentes. En la Ilustración 6 se muestra un ejemplo de la aplicación de estos dos modelos de extracción de fondos comentados sobre distintas escenas en la grabación de autopistas.

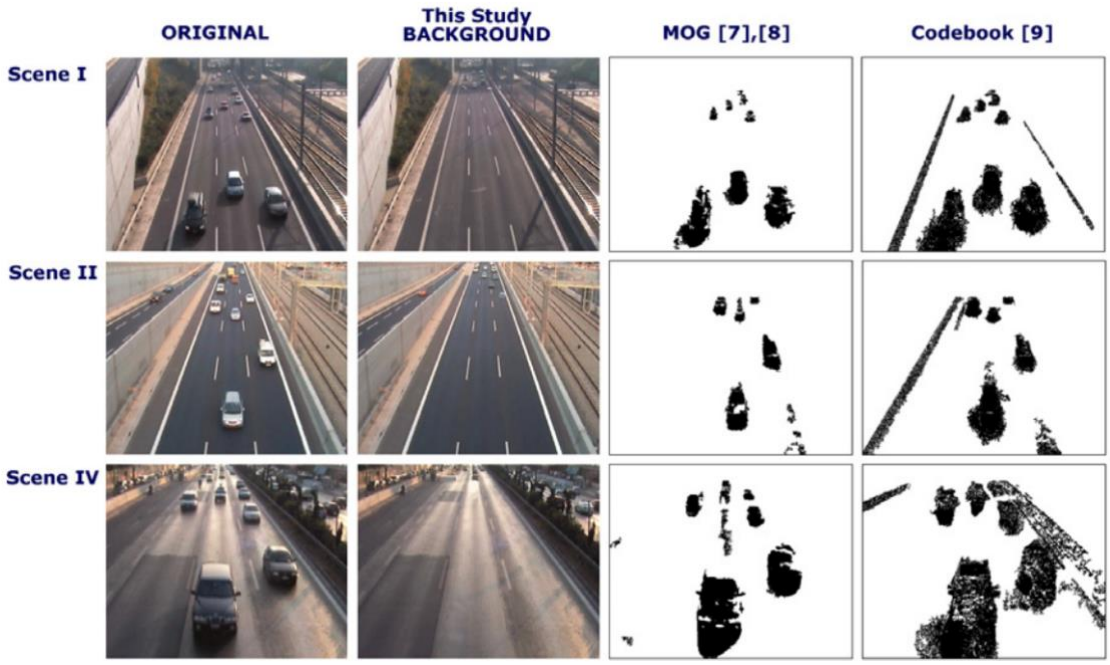

Ilustración 6 MOG y Codebook sobre la grabación de una autopista
Fuente: Adaptado de Mandellos, Keramitsoglou, & Kiranoudis, (2011)

En este trabajo se utiliza la extracción de fondos como método de selección de áreas de interés para el módulo de detección por varias razones: En primer lugar, se trata de modelos de diverso grado de complejidad, si bien el desempeño de todos ellos es similar, pudiendo encontrar **métodos simples, que dan buen resultado y cumplen con creces el requisito del funcionamiento en tiempo real**. En segundo lugar, **cualquier sujeto que aparece en escena** y que, por lo tanto, es susceptible de ser vigilado, **ha entrado en escena habiéndose movido previamente**, por lo que de cualquier manera este modelo será capaz de captar aquellas áreas de interés donde pueda encontrarse un sujeto potencialmente detectable. Por último, si bien se ha comentado previamente la no robustez de estos modelos en aspectos como la variabilidad de la iluminación, las sombras o el ruido en la imagen, la utilización de un algoritmo de selección de áreas de interés tiene la única finalidad de **acotar el espacio de búsqueda en la imagen a la hora de identificar a una persona**. Así, será el detector de humanos (el modelo CNN en el caso del presente trabajo) quien en último momento discierna si el área propuesta por el extractor de fondos se corresponde con un sujeto humano o, por el contrario, se trata de un falso

positivo derivado de las situaciones en las que el funcionamiento del modelo se ve en entredicho.

2.4 *Tracking* de múltiples objetos: *Tracking* de múltiples personas

El *tracking* de múltiples objetos (*Multiple Object Tracking* - MOT) es una línea de investigación muy popular en los últimos años, y **se utiliza ampliamente en muchas aplicaciones como la videovigilancia inteligente**, el reconocimiento de gestos y acciones, y el reconocimiento de eventos (C. Zhang et al., 2018). Por su parte, el seguimiento de varias personas es un problema crucial, desafiante y frecuente en los sistemas de videovigilancia inteligentes debido a la **existencia de escenas complejas** en las que se dan **oclusiones debido a objetos** que resultan obstáculos y a la influencia **otros sujetos humanos en escena** (Ju et al., 2015). El seguimiento de múltiples objetos tiene como objetivo **encontrar las trayectorias de los objetos en movimiento en una secuencia de vídeo**. Este problema normalmente se trata como una tarea de asociación. Así, un detector genérico localiza el rectángulo que delimita cada uno de los objetos (*bounding box*) en cada fotograma y luego un algoritmo de asociación se encarga de asociar cada uno de los correspondientes a los mismos objetos a través de los distintos fotogramas. En la Ilustración 7, se puede apreciar un ejemplo de *tracking* de personas a lo largo de una secuencia de fotogramas, en la que se preserva la identidad del objeto seguido.

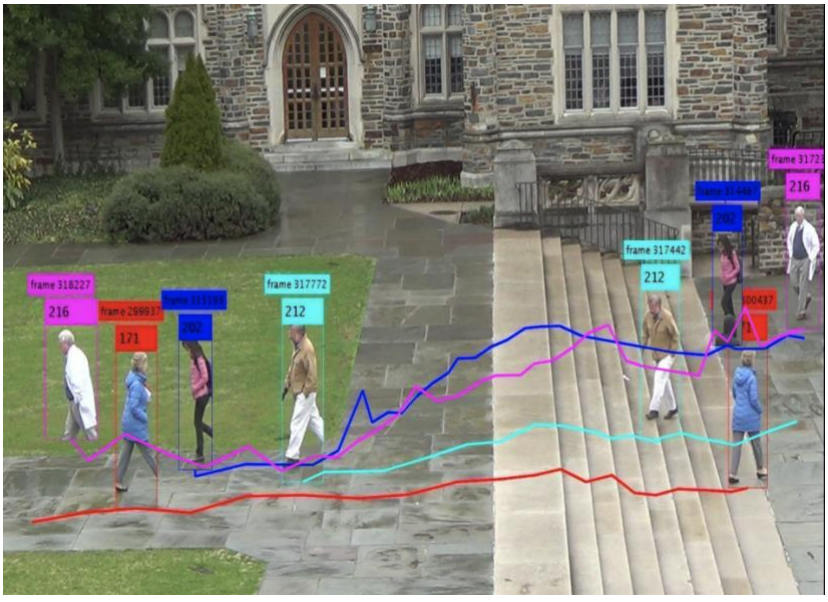

Ilustración 7 *Tracking* de múltiples personas
Fuente: Tesfaye, Zemene, Prati, Pelillo, & Shah, (2019)

Una distinción importante en los modelos de *tracking* es la de los modelos en línea (*online*) frente a los modelos fuera de línea (*offline*). Un **modelo en línea recibe la entrada de vídeo fotograma a fotograma**, y tiene que dar una salida para cada fotograma al momento. Esto significa que, además del **fotograma actual**, sólo se puede utilizar **información de los fotogramas anteriores**. **Los modelos offline**, al contrario, tienen acceso **a todo el vídeo**, lo que significa que además del fotograma actual, pueden utilizar información de fotogramas anteriores y futuros. De esta forma, la tarea puede ser vista como un problema de optimización, donde el objetivo es encontrar un conjunto de caminos que minimicen una función de pérdida global del seguimiento de los objetos (Smeulders et al., 2014). En una tarea de videovigilancia, nos encontramos en la necesidad de utilizar un modelo en línea al tratarse de un problema que ha de ser resuelto en tiempo real y en el que disponemos exclusivamente de la información recogida por las cámaras hasta el momento. Así, puede considerarse que **un algoritmo de *tracking* funciona en tiempo real si puede procesar más fotogramas por segundo que la tasa de fotogramas del vídeo de entrada** (Berclaz, Fleuret, Türetken, & Fua, 2011). Sin embargo, en sistemas en los que se necesita tanto llevar a cabo las detecciones como realizar el seguimiento, como es el caso de un sistema embebidos, los recursos son necesariamente compartidos entre las dos tareas, lo que reducirá la velocidad a la que el sistema puede operar en tiempo real (Bewley, Ge, Ott, Ramos, & Upcroft, 2016).

Los algoritmos tradicionales de *tracking* de objetos incluyen principalmente el desplazamiento medio o *mean shift* (Comaniciu & Ramesh, 2000), el filtro de partículas (*particle filter*) (Hue, Le Cadre, & Pérez, 2002), los algoritmos de diferencia entre fotogramas, filtro de Kalman (Kalman, 1960), etc. que **siguen la ubicación del objeto mediante múltiples iteraciones consecutivas**. En algunos escenarios simples, estos métodos tradicionales pueden lograr resultados de seguimiento precisos. Sin embargo, **el error acumulado de las iteraciones** a menudo lleva a que estos métodos **no puedan adaptarse a movimientos rápidos** o a cuando el **seguimiento se lleva a cabo sobre múltiples objeto**s. Así, en los últimos años, cada vez más investigadores han empezado a mostrar más atención a métodos de seguimiento por detección al

afrontar un problemas de *tracking* (Hao, Wang, Wu, & Sun, 2018). Así, **podemos considerar los problemas de *tracking* y detección análogos** en cierto sentido.

2.5 Re-identificación de personas

Recientemente, la re-identificación de personas ha atraído más y más la atención en el campo de la visión por computador debido a su importancia en aplicaciones reales como la videovigilancia, la robótica o la conducción automática (Ni et al., 2018; Sun, Jiang, Song, Lu, & Men, 2018). Habitualmente, la re-identificación de personas se refiere a la **relación de un sujeto humano determinado a lo largo del tiempo de una grabación o en distintas tomas de cámaras no solapadas de un circuito**. En la práctica, un sistema de re-identificación consiste en un detector de personas, un algoritmo de seguimiento o *tracker* y un emparejador de personas (*matcher*) (J. Zhang, Yuan, & Wang, 2019). Tal y como ya hemos visto anteriormente, la detección de personas y el *tracking* de personas son problemas independientes dentro de la visión por computador. Así, la re-identificación de personas se centra normalmente en otra tercera parte y **se considera un problema de recuperación y asociación**. En la Ilustración 8 se ejemplifica el problema de la re-identificación de personas en dos escenarios distintos: (a) de día, (b) de noche.

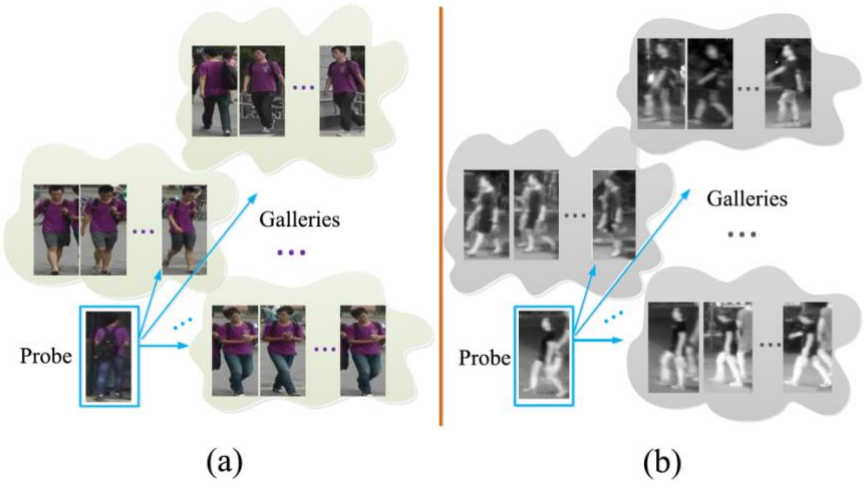

Ilustración 8 Problema de la re-identificación en dos escenarios distintos
Fuente: J. Zhang et al., (2019)

Antes del surgimiento de los métodos de aprendizaje profundo, había dos métodos principales para la re-identificación de personas: los métodos basados

en descriptores diseñados a mano y métodos de aprendizaje basado en métricas. **Los enfoques basados en descriptores** se centran en el diseño de descriptores de imagen que permitan **actuar de forma discriminante y robusta** ante la identificación de distintos individuos. Ejemplos de los métodos más conocidos son: *Local Maximal Occurrence* (LOMO) (Liao, Hu, Zhu, & Li, 2015), *Ensemble of Localized Features* (ELF) (Gray & Tao, 2008), *Gaussian Of Gaussian* (GOG) (Matsukawa, Okabe, Suzuki, & Sato, 2016) y *Symmetry-driven accumulation of local features* (SDALF) (Farenzena, Bazzani, Perina, Murino, & Cristani, 2010). Entre estos métodos, ELF es la más utilizado y tanto LOMO como GOG son considerados como los métodos más avanzados. En la Ilustración 9 se muestra de forma gráfica y resumida el esquema de funcionamiento que sigue el método de *Local Maximal Occurrence*.

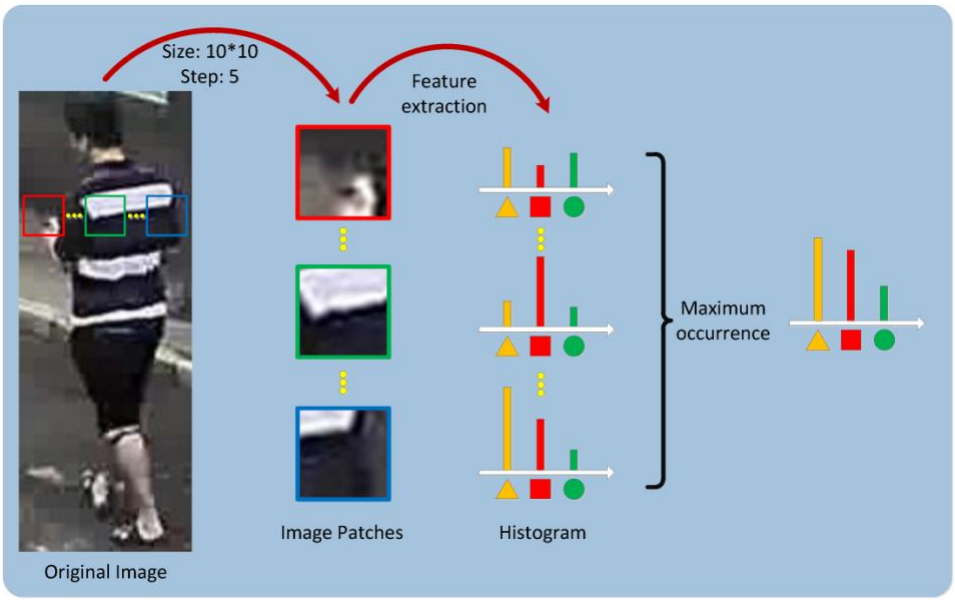

Ilustración 9 Extracción de características por el método LOMO
Fuente: Liao et al., (2015)

Los métodos de aprendizaje basado en métricas tienen, como su nombre indica, el objetivo del **diseño de métricas que resulten discriminatorias para el emparejamiento de imágenes de sujetos humanos**. La mayoría de estos métodos se basan en la distancia de Mahalanobis. Los métodos que se consideran los mejores de acuerdo a la literatura son: *Information-Theoretic Metric Learning* (ITML) (Davis, Kulis, Jain, Sra, & Dhillon, 2007), *Large Margin Nearest Neighbor Learning* (LMNN) (Weinberger & Saul, 2009), *Logistic Discriminant-based Metric Learning* (LDML) (Guillaumin, Verbeek, & Schmid,

2009), KISSME (Kostinger, Hirzer, Wohlhart, Roth, & Bischof, 2012), *Pairwise Constrained Component Analysis* (PCCA) (Mignon & Jurie, 2012), *Local Fisher Discriminant Analysis* (LFDA) (Pedagadi, Orwell, Velastin, & Boghossian, 2013), *Locally-Adaptive Decision Functions* (LADF) (Z. Li et al., 2013), *Cross-view Quadratic Discriminant Analysis* (XQDA) (Liao et al., 2015) y MLAPG (Liao & Li, 2015).

Al igual que en muchas otras tareas de visión por computador, **los métodos de aprendizaje profundo se han desarrollado y aplicado con éxito en la re-identificación de personas**. Así los métodos de aprendizaje profundo existentes para la re-identificación de personas se consideran un problema de ranking o de clasificación (J. Zhang et al., 2019). Los métodos basados en ranking toman tres imágenes como entrada donde dos de ellas son de la identidad correcta y otra es de una identidad diferente (*triplet loss*), de forma que se intenta **minimizar la distancia intra-clase** y **maximizar la distancia inter-clase**, tal y como se ejemplifica en la Ilustración 10.

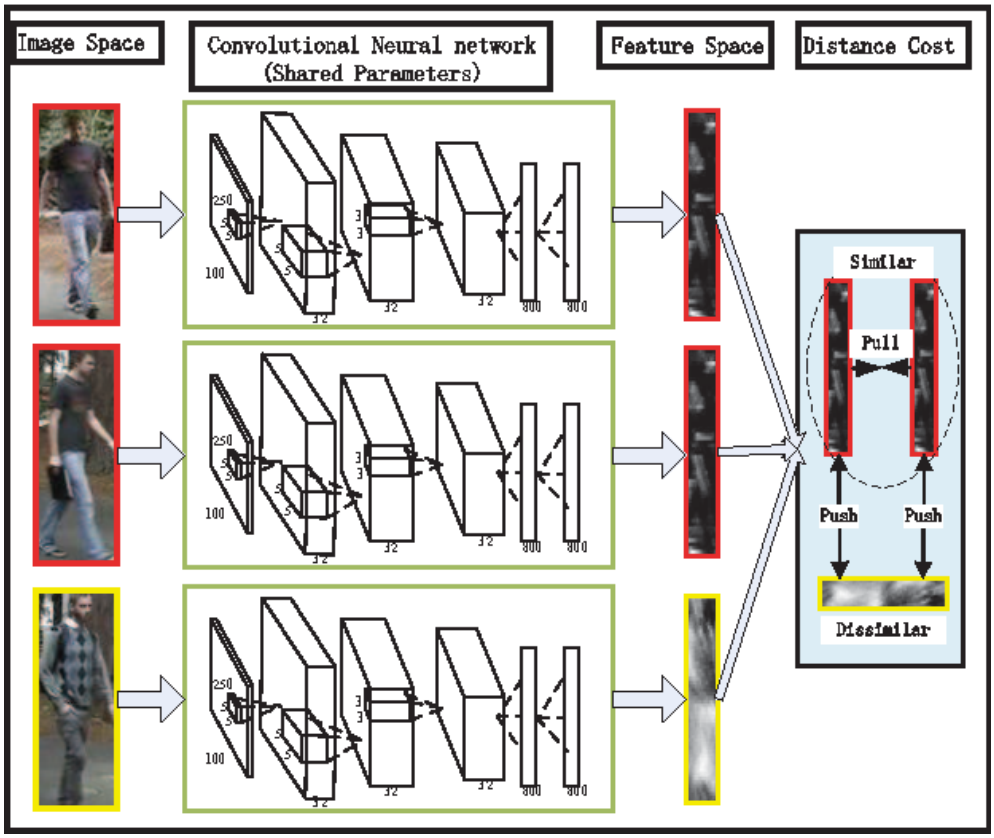

Ilustración 10 Ejemplo de la aplicación del triplet loss
Fuente: Cheng, Gong, Zhou, Wang, & Zheng, (2016)

Por su parte, los métodos basados en la clasificación para la re-identificación de personas se utilizan de estructuras complejas en las que interviene el coste de la clasificación. Así, encontramos desde métodos que utilizan clasificación binaria como el propuesto por Ahmed, Jones, & Marks (2015) o el propuesto por (Varior, Haloi, & Wang, 2016), hasta otros métodos que utilizan clasificación multiclase como SpindleNet (Haiyu Zhao et al., 2017), *Pose Sensitive Embedding* (PSE) (Hembury, Borovkov, Lintuluoto, & Inoue, 2003) o *Harmonious Attention Network* (HACNN) (W. Li, Zhu, & Gong, 2018).

3 Recursos y tecnologías

En este epígrafe se llevará a cabo la exposición de aquellos recursos y tecnologías que han sido requeridos durante la elaboración del trabajo.

3.1 Lenguaje de programación Python

Python es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y con semántica dinámica. Sus estructuras de datos de alto nivel, combinadas con la escritura y el enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripting o lenguaje de pegado para conectar componentes entre sí. Por su sintaxis simple, Python enfatiza en la legibilidad y por lo tanto reduce el coste de desarrollo y mantenimiento del programa. Del mismo modo, Python soporta módulos y paquetes, lo que fomenta la modularidad de los programas escritos en este lenguaje y la reutilización del código. El intérprete de Python y la su extensa librería estándar están disponibles en forma de código fuente de forma gratuita para todas las plataformas principales, además de que se pueden distribuir libremente (Python, 2017). Los motivos que han supuesto la elección de Python como lenguaje de programación para nuestro proyecto son los siguientes:

- **La gran variedad de librerías para IA**: Es una de las razones principales por las que Python es el lenguaje de programación más popular utilizado para la IA. Las librerías Python proporcionan elementos base para que los desarrolladores no tengan que codificarlos desde el principio cada vez. El aprendizaje automático requiere un procesamiento continuo de datos, y las bibliotecas de Python permiten acceder, manejar y transformar datos.
- **Flexibilidad**: Python posee ciertas características que permite que sea especialmente flexible y cómodo:
    - Se puede tanto programar utilizándolo como un lenguaje orientado a objetos (OOP) o como lenguaje de scripts. Esto permite realizar pruebas en la consola del intérprete sin necesidad de estructurar innecesariamente el código.

    - No hay necesidad de recompilar el código fuente, se puede implementar cualquier cambio y ver rápidamente los resultados.
    - Se puede combinar Python con otros lenguajes de programación. En caso de que por un criterio de rendimiento se requiera por ejemplo del uso de un lenguaje compilado, desde Python puede incluirse código y, por ende, características de otros lenguajes de programación.

- **Independencia de la plataforma**: Utilizar Python para el desarrollo de proyectos de aprendizaje automático tiene la ventaja de la garantía de su funcionamiento en cualquier otra plataforma.
- **Muchas opciones para visualización**: En la inteligencia artificial, el aprendizaje profundo y el aprendizaje automático, es vital poder representar los datos en un formato legible por el ser humano. Las bibliotecas de visualización de las que dispone Python permiten a los científicos de datos construir gráficos, histogramas y diagramas para una mejor comprensión de los datos y una presentación efectiva.

3.2 Bibliotecas utilizadas

Como se ha comentado, el lenguaje Python es considerado el más importante de todos los lenguajes de desarrollo de AI debido a su simplicidad. La sintaxis propia de Python es muy simple y se puede aprender fácilmente, por lo que muchos algoritmos de IA se implementan fácilmente en este lenguaje. A esta característica, se le une su simplicidad y potencia en la manipulación de datos y en la representación gráfica. Además, tanto estas propias funcionalidades, como otras nuevas pueden ser extendidas fácilmente mediante la instalación de diversos paquetes y librerías. Así pues, los paquetes y librerías que ha sido necesario instalar para el desarrollo del trabajo se presentan a continuación:

- **OpenCV** (Bradski, 2000): OpenCV es una biblioteca que proporciona una infraestructura común para las aplicaciones de visión artificial y para acelerar el uso de la percepción artificial en los productos comerciales. La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluyen

un conjunto completo de algoritmos de visión por computador y de aprendizaje automático, tanto clásicos como de última generación.

- **Keras** (Chollet, 2015): Keras es una API de redes neuronales de alto nivel, escrita en Python y capaz de funcionar sobre los frameworks TensorFlow, CNTK y Theano. Keras tiene el objetivo de permitir una rápida experimentación, de forma que se pueda pasar de la idea al resultado en el menor tiempo posible. Así, esta biblioteca permite un prototipado fácil y rápido (a través de la facilidad de uso, modularidad y extensibilidad). Keras soporta tanto redes convolucionales como redes recurrentes, así como combinaciones de ambas y se puede ejecutar tanto en la CPU como en la GPU.

- **Pandas** (McKinney, 2010): Pandas es una biblioteca que proporciona estructuras de datos de alto rendimiento y fáciles de usar y herramientas de análisis de datos para el lenguaje de programación Python.

- **Scikit-Learn** (Pedregosa et al., 2011): Scikit-learn es una biblioteca de aprendizaje automático de software libre para el lenguaje de programación Python. Cuenta con varios algoritmos de clasificación, regresión y clustering, incluyendo máquinas de soporte vectorial, random forest, gradient boosting, k-means y DBSCAN, y está diseñado para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy de Python.

- **NumPy** (Oliphant, 2006): NumPy es el paquete fundamental para la computación científica con Python. Contiene, entre otras cosas: un poderoso objeto array N-dimensional, funciones sofisticadas, herramientas para integrar código C/C++ y Fortran, herramientas útiles para álgebra lineal, transformada de Fourier, etc. Además de sus usos concretos, NumPy también puede ser utilizado como un contenedor multidimensional de datos genéricos muy eficiente, en el que se pueden definir tipos de datos arbitrarios. Esto permite que NumPy se integre sin problemas y rápidamente en una amplia variedad de bases de datos.

- **SciPy** (Jones, Oliphant, Peterson, & others, 2001): La biblioteca SciPy proporciona muchas rutinas numéricas eficientes y fáciles de usar, tales como rutinas de integración numérica, interpolación, optimización, álgebra lineal y estadística.

- **FilterPy** (Labbe, 2018): FilterPy es una biblioteca de Python que implementa una serie de filtros Bayesianos, haciendo especial hincapié en los filtros de Kalman.

3.3 Conjuntos de datos

Tanto para llevar a cabo el entrenamiento de los distintos algoritmos de aprendizaje automático utilizados en este trabajo, como para probar las diversas técnicas de visión por computador empleadas y para medir el rendimiento del sistema en su conjunto, ha sido necesario disponer de conjuntos de datos cuyos dominios se enmarcan en las características del trabajo. A continuación, se presentan cada uno de los conjuntos de datos que han sido empleados durante el desarrollo del presente proyecto:

- **INRIA Person Dataset** (Dalal & Triggs, 2005): Este conjunto de datos se recopiló como parte del trabajo de investigación sobre la detección de personas en posición erguida en imágenes y vídeo. El conjunto de datos se divide de dos formas: (a) imágenes originales con los correspondientes archivos en los que se incluyen las anotaciones, y (b) imágenes positivas en formato normalizado de 64x128 píxeles con las imágenes negativas originales. En total, este conjunto de datos incluye **3.542 imágenes positivas y 1.671 imágenes negativas**. El conjunto de datos INRIA ha sido utilizado en este proyecto para el entrenamiento de los distintos algoritmos de aprendizaje automático que han sido probados para la clasificación de personas dentro de la tarea de detección.

- **VIRAT Video Dataset** (Oh et al., 2011): Este conjunto de datos se creó para el programa de Recuperación y Análisis de Vídeo e Imágenes (VIRAT), proyecto de videovigilancia financiado por la Oficina de Tecnología de la Información (IPTO) de la Agencia de Proyectos de Investigación Avanzada de la Defensa (DARPA). Incluye **25 horas de**

grabación de cámaras estacionarias en 16 escenarios distintos, entre ellos: aparcamientos, calles, construcciones, etc. Este conjunto de datos está diseñado para ser realista, natural y desafiante en términos de resolución, ruido de fondo y diversidad de escenas y de categorías de actividades humana, así como de eventos. En comparación con los conjuntos de datos existentes, las características distintivas de este conjunto de datos son las siguientes:

- Realismo y escenarios naturales: Los datos se recogieron en escenas naturales que mostraban a personas que realizaban acciones normales en contextos estándar, con fondos no controlados y desordenados.
- Diversidad: Los vídeos fueron recolectados en múltiples sitios distribuidos a lo largo de los Estados Unidos. Se incluyeron una variedad de puntos de vista y resoluciones de cámara, y las acciones son realizadas por muchas personas diferentes.
- Cantidad: Se incluyen diversos tipos de acciones humanas e interacciones persona-vehículo, con un gran número de ejemplos (>30) por clase de acción.
- Amplia gama de resoluciones de imagen y de tasas de refresco de los vídeos.

Este conjunto de datos ha sido empleado para llevar las pruebas de funcionamiento del sistema en su conjunto y la evaluación de su rendimiento por medio del cálculo de las métricas de calidad que pueden ser observadas en el epígrafe de Resultados de este mismo documento.

- **Oxford Town Centre Dataset** (Harvey Adam. LaPlace, 2019): El conjunto de datos de *Oxford Town Centre* es un conjunto de vídeos obtenidos por un sistema de videovigilancia de peatones en una zona céntrica y concurrida de Oxford que se utiliza para la investigación y el desarrollo de sistemas de reconocimiento de actividades y rostros. El vídeo se obtuvo a través de una cámara de vigilancia en la esquina de *Cornmarket* y *Market St.* En Oxford, Inglaterra, e incluye a unas **2.200 personas**. Este

conjunto de datos es el único que utiliza imágenes de una **cámara de vigilancia pública** que, de otro modo, se destinarían a la seguridad pública. Así, este conjunto de datos es interesante por que muestra a los peatones actuando con normalidad y sin ensayar, indicando que ni conocían ni consintieron su participación en la grabación. Este conjunto de datos es utilizado en nuestro proyecto para las diversas pruebas de concepto que se han considerado en cada uno de los módulos del sistema, así como para probar la robustez del sistema en su conjunto y la interacción de los módulos entre sí.

3.4 Recursos *hardware*

La elaboración del presente trabajo se ha llevado a cabo en dos ordenadores personales distintos. Tratándose uno de estos de un ordenador de sobremesa, donde se ha realizado la mayor parte del trabajo, mientras que el otro se corresponde con un ordenador portátil.

3.4.1 Ordenador de sobremesa

En el ordenador de sobremesa es sin lugar a duda donde se ha llevado a cabo la mayor parte de la elaboración de este trabajo. Esto ha sido debido al *hardware* de este que, en comparación al del ordenador portátil, le permite llevar a cabo una ejecución de los procesos necesarios para el desarrollo del proyecto, disminuyendo el tiempo de ejecución de estos. A continuación, en la Tabla 2 se presentan aquellas características o especificaciones técnicas del ordenador de sobremesa más significativas.

Tabla 2 Especificaciones del ordenador de sobremesa

Característica	Valor
Sistema Operativo	Ubuntu 18.04 bionic (64 bits)
Procesador	Intel Core™ i5-8400 @ 2.80GHz x 6
Memoria RAM	2 x 16,0 GB DDR4
Disco duro 1 (SSD)	256 GB
Disco duro 2 (HDD)	2 TB
Tarjeta gráfica	Nvidia RTX2080 (8GB)

Fuente: Elaboración propia

3.4.2 Ordenador portátil

Por cuestiones de disponibilidad horaria y la consiguiente flexibilidad necesaria, para la elaboración del presente trabajo, fue necesario realizar parte de este en un equipo secundario, tratándose este de un ordenador portátil de la compañía Apple, concretamente un modelo MacBook pro. A continuación, en la Tabla 3 se presentan aquellas características o especificaciones técnicas más significativas de este ordenador portátil. Cabe destacar que, en cualquier caso, el uso de este ordenador ha estado principalmente enfocado a la conexión con el ordenador sobremesa anteriormente presentado para trabajar en este de forma remota.

Tabla 3 Especificaciones del ordenador portátil

Característica	Valor
Sistema Operativo	macOS High Sierra 10.13.6
Procesador	Intel Core™ i5-6360U @ 2.0 GHz x 2
Memoria RAM	2 x 4,0 GB LPDDR3
Disco duro 1 (SSD)	256 GB
Tarjeta gráfica	Intel Iris™ Graphics 540

Fuente: Elaboración propia

4 Metodología, desarrollo e implementación

En el presente epígrafe se tratarán aquellos aspectos relevantes de la sistemática seguida a raíz de la metodología elegida para el desarrollo de este trabajo. A su vez y conjuntamente, detalla el conjunto de tareas y el flujo de ejecución realizado para la implementación de la solución propuesta a la problemática planteada en este trabajo y de acuerdo con los objetivos concretos previamente definidos.

4.1 Requisitos del sistema

Como ya se ha comentado previamente, el objetivo fundamental de este trabajo es la **implementación de una utilidad software para la videovigilancia inteligente**. Este sistema deberá cumplir con cada uno de los objetivos planteados inicialmente en este trabajo, lo que en gran medida supondrá que se vea condicionada tanto la estructura como las características de la solución que en este epígrafe se presenta.

Así, el primer requisito que se espera de la solución software propuesta es a nivel funcional, esperándose que esta sea capaz de llevar las tareas de **detección de personas**, **re-identificación**, seguimiento o ***tracking*** y **posicionamiento** con un nivel de precisión y sensibilidad acorde a las exigencias de un sistema de vigilancia para que pueda ser considerado como inteligente.

El otro requisito fundamental que cumplir es la **minimización del coste computacional** necesario para que este sistema pueda funcionar, permitiendo que su despliegue se encuentre en un rango presupuestario realista y que pueda funcionar sobre un sistema embebido. Por ello, es necesario garantizar el requisito fundamental del funcionamiento en tiempo real del sistema en su conjunto sobre un sistema de altas prestaciones.

4.2 Solución propuesta

Teniendo en mente los requisitos anteriores, la solución software que en este trabajo se presenta se compone de distintas funcionalidades. En primer lugar, se lleva a cabo por medio de extracción de fondos una selección de las áreas de interés (ROI) a analizar dentro de la imagen, correspondientes a zonas en las

que ha habido un movimiento de alguno de los objetos presentes en escena. El análisis de dichas regiones se lleva a cabo con un clasificador CNN, que comprobará si el movimiento viene dado por la existencia de sujetos humanos en escena, realizando la detección de estos en caso afirmativo. A continuación, se lleva a cabo el seguimiento en la imagen de cada una de las detecciones realizadas durante un cierto número de fotogramas, re-identificando en primer lugar cada una de dichas detecciones por medio de la apariencia de las personas en cuestión en caso de que fuera posible y alternativamente, haciendo uso de la predicción de la trayectoria de esta a fin de emparejar la detección actual con la predicción de la última detección del sujeto para este momento. Una vez confirmada la presencia de una misma persona en escena durante varios fotogramas consecutivos, se inicia un seguimiento en firme de esta, dotando de identificación al sujeto y guardando un historial de la cada de las detecciones llevadas a cabo sobre el mismo, de forma que la re-identificación resulte cada vez más precisa. Para llevar a cabo la predicción de la trayectoria en el *tracking* se harán uso de las coordenadas reales de la situación de cada una de las detecciones dentro de la instalación. Esto permite tanto llevar a cabo los cálculos pertinentes en términos de la situación real de cada una de las detecciones dentro del mundo, como la posibilidad de la representación de dicha situación sobre un plano 2D a vista de pájaro de la escena, facilitando la tarea del control de perímetros de la instalación videovigilada.

4.3 Descripción de los módulos

Al tratarse de un **sistema de naturaleza modular**, con cada uno de estos módulos correspondientes a sus distintas funcionalidades interactuando entre sí, se ha considerado a efectos didácticos, la consiguiente división que permite analizar su implementación de manera individualizada. El esquema de dichos módulos y de la interacción entre ellos puede apreciarse en la Ilustración 11.

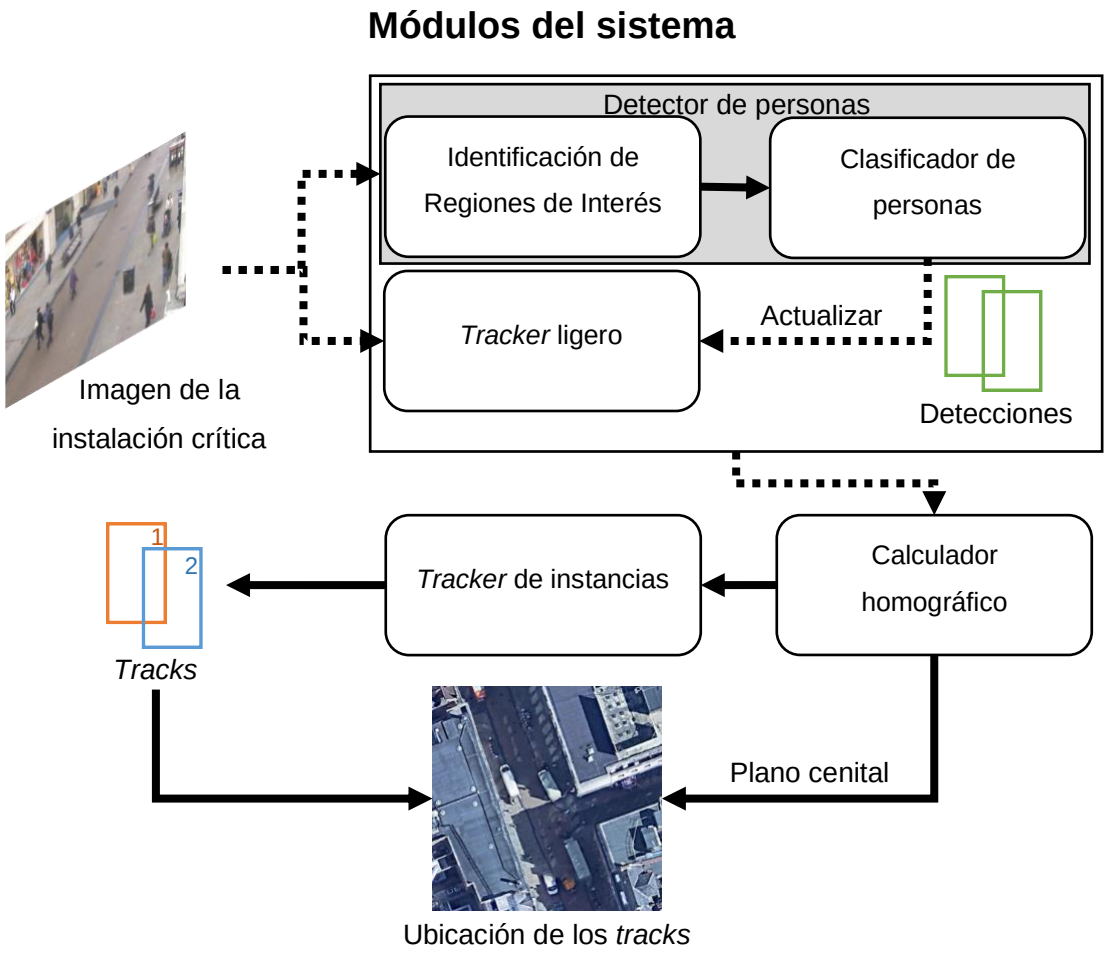

Ilustración 11 Módulos del sistema e interacción entre ellos
Fuente: Elaboración propia

Tal y como se puede observar en dicha figura, el proceso se inicia con la llegada al sistema de una imagen de una de las cámaras de la instalación videovigilada. Sobre esta imagen pueden realizarse **dos tareas análogas**, que son ejecutadas alternativamente a fin de disminuir el uso de recursos computacionales:

- **Detección de personas**: **Identificación de regiones de interés** (ROIs) en la imagen y **clasificación** de estas de acuerdo con la existencia o no de personas. Esta tarea tiene por resultado la obtención de las detecciones de personas en la imagen.
- ***Tracking* de las detecciones en la imagen**: Sobre las detecciones de personas anteriormente obtenidas, se lleva a cabo la predicción de la nueva ubicación de los sujetos en la nueva imagen. La detección de

personas permitirá actualizar la ubicación real de los sujetos de cara a efectuar el *tracking* en la imagen posteriormente.

Sea cual fuera la tarea ejecutada de las anteriormente presentadas, se tiene como resultado la ubicación de las personas presentes en la imagen. A continuación, se lleva el cálculo de las coordenadas reales que se corresponden a las ubicaciones de dentro de la imagen obtenidas anteriormente, por medio del **calculador homográfico**. A partir de entonces, el ***tracker* de instancias** se encarga de asociar las coordenadas con identidades concretas de personas que han sido seguidas a lo largo del tiempo. Si no se pudiera efectuar esta asociación, bien por la inexistencia del sujeto dentro del sistema previo a dicho momento o debido a que se ha dejado de detectarlo dentro de la escena, se iniciará un nuevo seguimiento o se dejará de realizarlo sobre el sujeto en cuestión, respectivamente.

A continuación, se explican en detalle todos aquellos aspectos relevantes del proceso de diseño e implementación de cada uno de los módulos anteriormente presentados.

4.3.1 Identificador de regiones de interés (ROI)

El módulo del identificador de regiones de interés tiene por finalidad **disminuir el espacio de búsqueda** dentro de cada una de las imágenes del flujo de la secuencia de vídeo analizada, a fin de identificar aquellas **áreas con mayor probabilidad de contener algún objeto de interés**. En nuestro caso, se ha utilizado como método de identificación la extracción de fondos, con el objetivo de obtener exclusivamente aquellos objetos en movimiento que hay en escena. La consideración de esta técnica se apoya en los siguientes criterios:

- **Cualquier objeto entra en escena moviéndose**: que inicialmente quiera ser detectado (en nuestro caso personas) ha de entrar en la escena habiéndose movido previamente, por lo que, en cualquier caso, su movimiento siempre quedará recogido.

- **Es una técnica computacionalmente eficiente**: Frente a la diversidad de algoritmos de extracción de fondos, cada cual con distinto coste computacional, encontramos un buen número de ellos muy eficientes.

Los algoritmos de extracción de fondos considerados en este trabajo son aquellos disponibles dentro de la librería de OpenCV (Bradski, 2000). En la Tabla 4 se presenta tanto un resumen de aquellos algoritmos disponibles dentro de esta biblioteca que han sido considerados en este trabajo.

Tabla 4 Algoritmos de extracción de fondos disponibles

Algoritmo	Descripción
Mixture of Gaussians (MOG) (KaewTraKulPong & Bowden, 2002)	Utiliza un método para modelar cada píxel de fondo mediante una mezcla de K distribuciones gaussianas (K = 3 a 5). Los pesos de la mezcla representan las proporciones de tiempo que esos colores permanecen en la escena. Los colores de fondo más probables son los que permanecen más tiempo y más estáticos.
MOG2 (Zivkovic & Van Der Heijden, 2006)	También está basado en la mezcla gaussiana. Este algoritmo selecciona el número apropiado de distribuciones gaussianas para cada píxel. Proporciona una mejor adaptabilidad a las diferentes escenas debido a los cambios de iluminación, etc.
GMG (Godbehere, Matsukawa, & Goldberg, 2012)	Este algoritmo combina la estimación estadística del fondo y la segmentación bayesiana. Utiliza los primeros fotogramas (120 por defecto) para el modelado de fondo. Emplea algoritmos probabilísticos de segmentación de primer plano que identifican posibles objetos utilizando inferencia bayesiana. Las estimaciones son adaptables, de forma que las observaciones más recientes están más ponderadas que las antiguas para adaptarse a cambios en la iluminación.
Local SVD Binary Pattern (LSBP) (Guo, Xu, & Qiang, 2016)	Al contrario que la mayoría de los algoritmos que, utilizan descriptores de color y textura, este modelo de extracción de fondos adaptativo se utiliza un descriptor LBP conjuntamente a SVD. Este descriptor permite describir la estructura de las regiones locales en una imagen dada, por lo que puede mejorar la robustez en la detección del fondo ante variaciones de iluminación, ruido y sombras.
KNN (Zivkovic & Van Der Heijden, 2006)	Se utilizan ecuaciones recursivas para actualizar constantemente los parámetros de un modelo de mezcla de distribuciones gaussianas y para seleccionar el número apropiado de componentes para cada píxel. Se utiliza el método KNN para una mejor estimación de la densidad del kernel.

A fin de elegir aquel algoritmo que mejor se adapte a nuestros requisitos, se visualiza el funcionamiento de estos ante escena que presentan distintas situaciones. En la Ilustración 12 se recoge un fotograma de una de dichas pruebas y el resultado de la ejecución de cada uno de los algoritmos de extracción de fondos considerados.

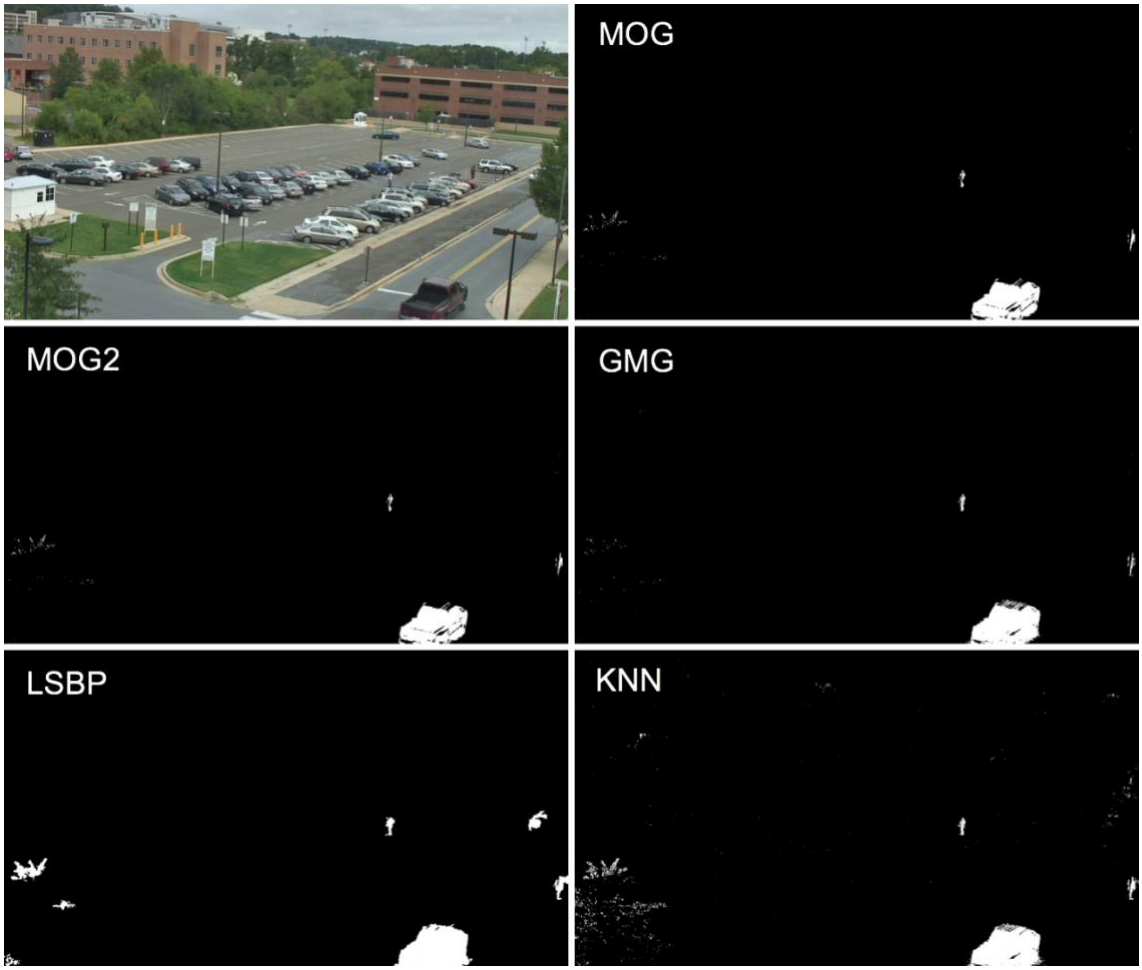

Ilustración 12 Prueba de algoritmos de extracción de fondos

Tal y como se puede apreciar en la imagen anterior, la capacidad de extracción de los componentes principales en movimiento es buena en todos los algoritmos, si bien el KNN capta más ruido que el resto y el LSBP considera elementos con un movimiento casi imperceptible, como las hojas de los árboles. Así, **se ha elegido algoritmo de extracción de fondos final el MOG2**, debido a su **buen desempeño en las pruebas realizadas**. Así, por encima de la prueba empírica anteriormente mostrada, esta decisión se ha visto influenciada por la prueba de eficiencia efectuada sobre cada algoritmo. En la Tabla 8 del epígrafe de Resultados se muestra el número de fotogramas por segundo que se pueden

procesar con un ordenador de las característica mostradas en el apartado de Ordenador de sobremesa, siendo **MOG2 aquel algoritmo más rápido**.

4.3.1.1 Transformaciones morfológicas

Tal y como se pudo apreciar en la Ilustración 12, la **extracción de fondos permite obtener una buena aproximación** de todos aquellos objetos en movimiento, si bien es verdad que hay áreas de dichos objetos que no son del todo captadas en su totalidad. Del mismo modo, ante la presencia de ruido en el fondo, causado por movimiento de objetos de pequeñas dimensiones no interesantes para el análisis (Por ejemplo: movimiento de hojas, cambios de iluminación, etc.), se ha de intentar eliminar la mayor parte de este.

Por medio de transformaciones morfológicas, **se consigue eliminar el ruido** y dar **mayor importancia a los objetos en movimiento de mayor tamaño detectados en escena**. Las transformaciones morfológicas son operaciones sencillas basadas en la forma de la imagen. Normalmente se realiza en imágenes binarias. Necesita dos entradas, una es nuestra imagen original, la segunda se llama elemento estructurador o *kernel* que decide la naturaleza de la operación. En la Ilustración 13 se pueden observar aquellas transformaciones morfológicas realizadas en este trabajo al realizar la extracción de fondos.

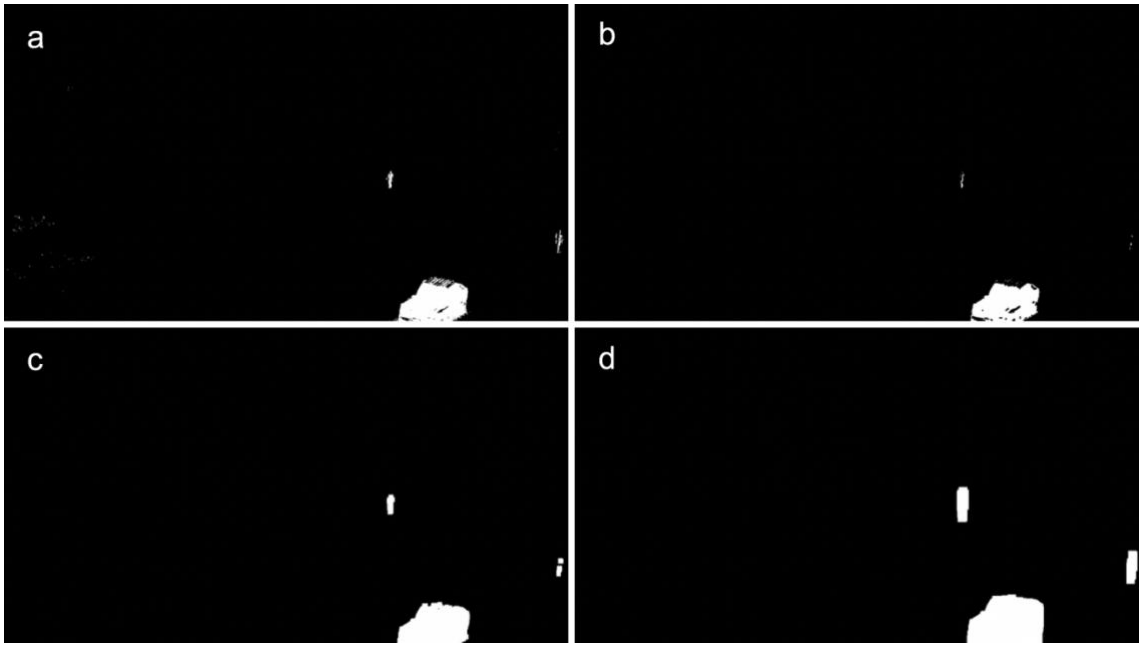

Ilustración 13 Transformaciones morfológicas realizadas
Fuente: Elaboración propia

El orden y la finalidad de cada una de las transformaciones morfológicas que se realizan tras el **algoritmo de extracción de fondos (a)** y para las cuales se muestra un ejemplo del resultado de su aplicación en la Ilustración anterior, quedan recogidas a continuación:

1. **Erosión (b)**: El kernel se desliza a través de la imagen (como en la convolución 2D). Un píxel en la imagen original (1 o 0) se considerará 1 sólo si todos los píxeles del núcleo son 1, de lo contrario se erosiona (se hace a cero). Así que lo que sucede es que, todos los píxeles cerca de los límites serán descartados dependiendo del tamaño del núcleo. Así que el grosor o el tamaño del objeto en primer plano disminuye o simplemente la región blanca disminuye en la imagen. Como puede apreciarse, esta transformación útil para eliminar pequeños ruidos blancos, separar dos objetos conectados, etc. En nuestro caso, el kernel que se ha utilizado para esta transformación tiene un tamaño de 3x3 píxeles.

2. **Elemento estructural en forma de cruz (c)**: En ciertas ocasiones, es deseable crear elementos estructurales con una cierta forma, como elíptica, circular, rectangular, etc. En nuestro caso, utilizamos un elemento estructural en forma de cruz, cuyo *kernel* tendrá un tamaño de 10x10 píxeles.

3. **Dilatación (d)**: Transformación de funcionamiento contrario a la erosión. En esta transformación, un elemento de píxel es 1 si al menos un píxel del *kernel* lo es. Por lo tanto, aumenta el área blanca en la imagen o el tamaño del objeto en primer plano. Normalmente y tal como consideramos en este proyecto, en casos como la eliminación de ruido, la erosión es seguida por la dilatación, ya que la erosión elimina los ruidos blancos, pero también encoge nuestro objeto, motivo por el que lo dilatamos, ya que el ruido se ha ido y no regresará, pero el área de nuestro objeto aumentará. Para esta transformación se ha considerado un kernel de tamaño 35x10 píxeles.

4.3.2 Clasificador de personas

Una vez que se han detectado aquellas áreas de dentro de la imagen que con mayor probabilidad pueden contener a una persona, en este siguiente módulo se llevará a cabo la clasificación entre aquellas que realmente las contienen y las que no. Para efectuar esta tarea se han considerado diversos algoritmos de *Machine Learning* y técnicas para poder **discriminar** de la mejor manera posible **entre las regiones de la imagen con y sin personas**. El conjunto de datos que se ha utilizado para el entrenamiento de cualquiera de estos algoritmos es el *INRIA Person Dataset* (Dalal & Triggs, 2005), cuyas características pueden consultarse en el epígrafe correspondiente a Conjuntos de datos de esta memoria.

4.3.2.1 HOG + Máquina de soporte vectorial

En un primer momento se consideró la utilización de descriptores sobre las imágenes a clasificar y con ellos, entrenar una máquina de soporte vectorial (*Support Vector Machine* - SVM). Esta es una de las técnicas clásicas más habituales en los sistemas de detección de personas y resulta un buen punto de partida sobre el que evaluar el rendimiento de los algoritmos que utilicemos a partir de este.

El descriptor que se consideró en nuestro caso es el **Histograma de Gradientes Orientados** (*Histogram of Oriented Gradients* - HOG). Este descriptor asume que la apariencia y la forma de un objeto dentro de **una imagen pueden ser descritas** por la distribución de la intensidad de los gradientes o de **las direcciones de los bordes**. La implementación de este descriptor se basa en la división la imagen en pequeñas regiones conectadas (células). Para cada célula se calcula un histograma de las direcciones de los gradientes (es decir, las direcciones de los bordes) para los píxeles de dentro de la célula. La combinación de estos histogramas representa el descriptor (Dalal & Triggs, 2005). En la Ilustración 14 se muestra de forma gráfica un ejemplo del cálculo del descriptor de histograma de gradientes orientados sobre la imagen de una persona.

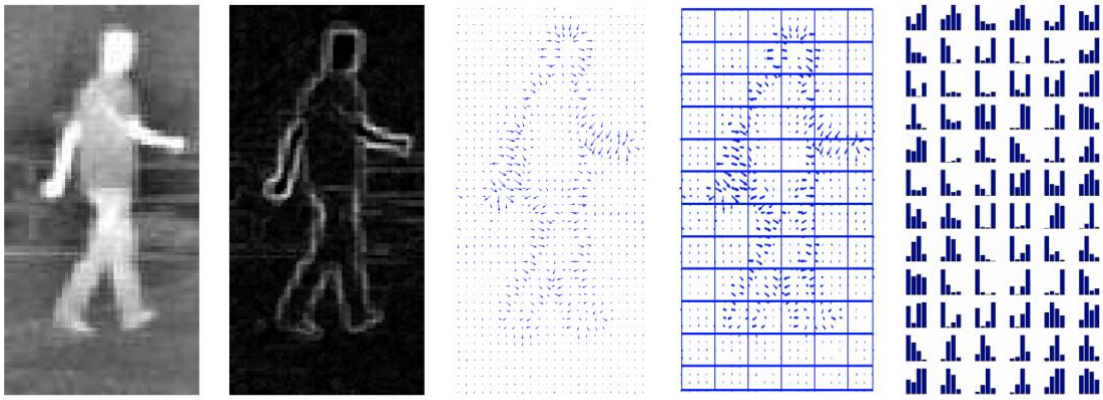

Ilustración 14 Ejemplo práctico del cálculo del descriptor HOG
Fuente: Bertozzi et al., (2007)

Un aspecto relevante para sacar el máximo partido al algoritmo de SVM es la **configuración de los parámetros** de este. Así, realizamos una búsqueda del mejor parámetro C (coste de la clasificación errónea), del parámetro gamma, cuando este sea necesario, y la elección del kernel a utilizar. En la Tabla 5 se muestran los valores de los distintos parámetros considerados en la búsqueda. Sobre ellos se ejecutan todas las combinaciones posibles y su resultado puede visualizarse en la Tabla 9 del epígrafe de Resultados. Sin embargo, y si bien los resultados obtenidos sobre el conjunto de datos en cuestión no son malos, se consideraron estas pruebas como un punto de partida y una referencia que poder utilizar para comparar el resto de las pruebas realizadas sobre algoritmos distintos a SVM.

Tabla 5 Combinaciones de parámetros de SVM considerados

Parámetro	Valor
Kernel	Lineal, gaussiano, polinómico, sigmoideo
C	1, 5, 10, 50, 100
Gamma	0.0001, 0.001, 0.01, 0.1

4.3.2.2 CNN: Transfer Learning

Las siguientes pruebas que se realizaron para la clasificación de personas se llevaron a cabo por medio de redes neuronales convolucionales (CNN). Como en el contexto de la **detección de personas, no se dispone públicamente de un conjunto de datos anotado de unas dimensiones suficientes** como para

abordar directamente el problema y así entrenar un modelo íntegramente, se optó por **reutilizar una red ya entrenada** para la detección de objetos sobre un conjunto de datos significativamente grande. Aquellos modelos de redes neuronales que han sido entrenados con ImageNet han demostrado ser un buen punto de partida para realizar tareas diferentes a la clasificación de objetos. En nuestro caso, realizamos ***Transfer Learning*** sobre uno de estos modelos pre-entrados, para así llevar a cabo la clasificación binaria entre personas y no-personas.

El modelo base pre-entrenado que se ha considerado es *MobileNetV2* (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018), sucesor de *MobileNetV1* (Howard et al., 2017). **Este es un modelo que mejora el estado del arte** en cuanto a las métricas de rendimiento de los modelos móviles en múltiples tareas, especialmente en **clasificación de imágenes y en tareas de detección para aplicaciones móviles**. Una desventaja que ha ido experimentando los últimos años es que las redes resultantes terminan siendo muy complejas y computacionalmente mas costosas. Por esta razón, los modelos móviles persiguen diseños lo más sencillos posible.

Así, la idea principal detrás de los modelos *MobileNet* es que es que las capas convolucionales, que son esenciales para las tareas de visión por computador, pero son computacionalmente caras de calcular, pueden ser reemplazadas por las llamadas convoluciones separables en profundidad (*depthwise separable convolutions*), que hacen aproximadamente la misma labor, pero de forma mucho más rápida. Esto se debe a que estas convoluciones **factorizan el kernel de la convolución**, de forma que se realizan dos convoluciones de menor tamaño: una a nivel de cada canal de la imagen y otra en profundidad. En la Ilustración 15 se puede apreciar la estructura del bloque principal de la red MobileNetV2.

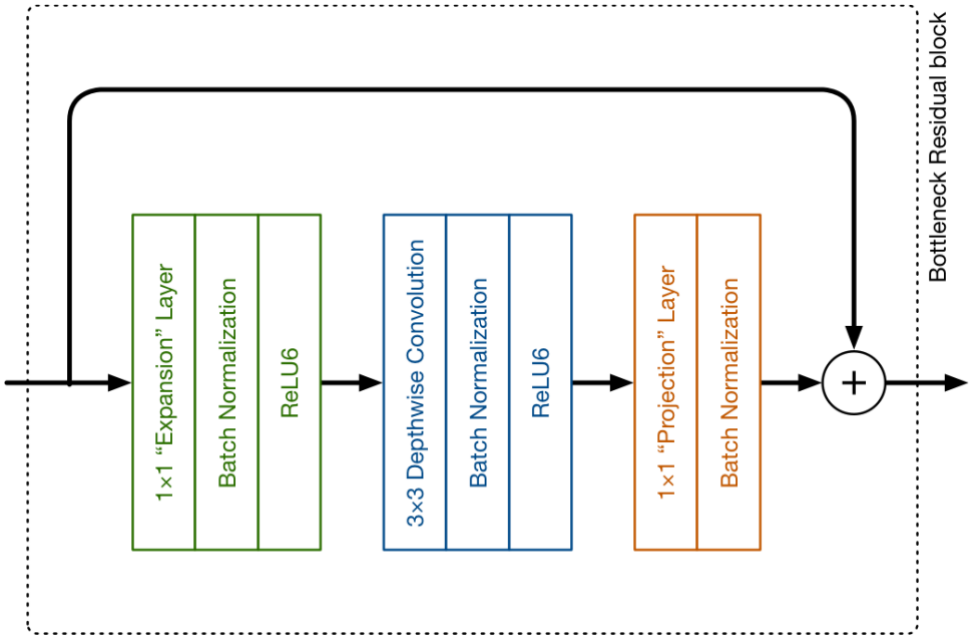

Ilustración 15 Bloque principal de MobileNetV2
Fuente: Hollemans, (2018)

Tal y como se puede observar en la figura anterior, el bloque se compone de tres capas: La primera capa (*expansion layer*) es una convolución de 1×1 que tiene por objetivo expandir el número de canales de los datos. La segunda capa (*depthwise convolution*) realiza una convolución en cada uno de los canales de forma independiente. La tercera y última capa (*projection layer*), lleva a cabo una proyección de los datos con un número elevado de canales en un tensor con un número mucho menor de dimensiones. En cuanto a la conexión residual, esta ayuda con el flujo de gradientes a través de la red y sólo se utiliza cuando el número de canales que entran en el bloque es el mismo que el número de canales que salen de él. **La arquitectura completa de *MobileNetV2* consta de 17 de estos bloques**, seguidos por una convolución 1×1, una capa de global average pooling y una capa de clasificación.

Como la elección de cada una de las tecnologías que se utilizan en nuestro sistema vienen dadas por el requisito del funcionamiento en tiempo real, el principal motivo que sustenta la elección de un modelo con esta arquitectura fue la eficiencia que trae consigo. Tal y como se ha comentado, *MobileNetV2* está diseñado para proveer una mayor velocidad en el cálculo que el resto de las arquitecturas existentes hasta el momento. Esto, unido al bajo número de parámetros de la red y la baja carga en memoria que supone, resultó definitivo

para su elección. En la Tabla 6 se exponen las características de cada uno de los modelos principales entrenados a partir de *MobileNetV2*.

Tabla 6 Principales modelos entrenados con Transfer Learning

Modelo	Características del modelo	Parámetros	Peso (MB)
MNV2_1	• Arquitectura MobileNetV2 original • Tres capas densas con *Dropout* de 0.5 entre 1ª y 2ª • Reentrenamiento de las capas densas • Tamaño de *Batch*: 16 • Optimizador Nadam con *Learning rate* de 0.0001	4.1 M	31.60
MNV2_2	Igual que MNV2_1 pero: • Dos capas densas con *Dropout* de 0.5 • Tamaño de *Batch*: 32 • Optimizador Nadam con *Learning rate* de 0.00001	2.9 M	17.40
MNV2_3	Igual que MNV2_2 pero: • Dos capas densas con *Dropout* de 0.3	2.9 M	17.40
MNV2_4	• Dos primeros bloques de MobileNetV2 • Convolución y capa de normalización extra • Dos capas densas con *Dropout* de 0.5 • Reentrenamiento de las capas añadidas • Tamaño de *Batch*: 32 • Optimizador Nadam con *Learning rate* de 0.0001	75.186	0.88
MNV2_5	Igual que MNV2_4 pero: • Tres primeros bloques de MobileNetV2	102.210	1.14

En cuanto a las pruebas realizadas, los primeros modelos generados (de los cuales se muestran **MNV2_1**, **MNV2_2** y **MNV2_3**) se basaron exclusivamente en la variación del número de capas densas y de dropout del modelo. Una vez que se obtuvo un resultado que hiciera pensar en la elección de esta arquitectura de red como un acierto, se llevaron a cabo pruebas para disminuir el tamaño de esta y así alinearla aún en mayor medida a los objetivos específicos de eficiencia de nuestro sistema (de las cuales se muestran **MNV2_4** y **MNV2_5**). Así, se disminuyeron el número de bloques de la arquitectura *MobileNetV2* de 17 a 2, variando a continuación el número de capas densas y su tamaño, así como el número de capas de *dropout* y su valor. Se puede apreciar esta disminución del

tamaño en el número de parámetros de cada uno de los modelos, por ende, de su peso en disco.

Las métricas de calidad de cada uno de estos modelos pueden consultarse en la Tabla 10 del apartado de Resultados. El modelo que se ha escogido finalmente para llevar a cabo la clasificación de personas dentro de este módulo es el **MNV2_5**, debido a su buen intercambio entre número de parámetros y calidad de sus resultados.

4.3.3 *Tracker* ligero

Este módulo tiene por finalidad predecir dentro de una imagen las nuevas ubicaciones de aquellas personas que ya han sido detectadas previamente. Por ello, este módulo se encarga, alternativamente con el selector de regiones de interés y el clasificador de personas (que conjuntamente realizan la tarea de detección de personas), de **ubicar a todas las personas dentro de la imagen en cada momento**. Así, si bien es cierto que la detección es más precisa que el *tracking*, los motivos que han llevado a la consideración de un algoritmo de estas características son los siguientes:

- **El *tracking* es más rápido que la detección**: Por lo general, los algoritmos de seguimiento son más rápidos que los de detección. Cuando se realiza el *tracking* de un objeto que fue detectado en un fotograma anterior, se sabe mucho acerca de la apariencia del objeto, su ubicación, velocidad, dirección, etc. Por lo que se puede utilizar toda esta información para predecir la ubicación del objeto en el siguiente cuadro y hacer una pequeña búsqueda alrededor de la ubicación esperada del objeto para localizar con precisión el objeto. Así, de cada n fotogramas, se ejecuta la detección una única vez, al ser más costosa, mientras que en los n-1 fotogramas restantes se ejecuta el algoritmo de *tracking*. Es cierto que la ejecución de forma iterativa de un algoritmo de *tracking* supone que se puedan ir acumulando errores y es por esto por lo que la utilización de la detección permite reiniciar el *tracking*.

- **Conocimiento de la instancia del objeto que se sigue**: Así como un algoritmo de detección está entrenado con un gran número de ejemplos

del objeto que se quiere detectar y, por lo tanto, tienen más conocimiento sobre la clase general del objeto, un algoritmo de *tracking*, por otro lado, saben más sobre la instancia específica de la clase que se sigue.

- **El *tracking* apoya ante fallos de la detección**: En caso por ejemplo de una oclusión que no permita llevar a cabo la detección en algún momento, un algoritmo de *tracking*, por su parte, podrá gestionar tal situación, permitiendo predecir la ubicación del objeto que se encuentra ocluido.

Los distintos algoritmos de *tracking* considerados para este módulo son los provistos dentro de la biblioteca de OpenCV (Bradski, 2000). Así, en la Tabla 7 se muestra una pequeña descripción de cada uno de ellos.

Tabla 7 Algoritmos de *tracking* considerados

Algoritmo	Descripción
Boosting (Grabner, Grabner, & Bischof, 2006)	Este algoritmo se basa en una versión on-line de *AdaBoost*. Este algoritmo de *tracking* se entrena en tiempo de ejecución con ejemplos positivos y negativos del objeto que se sigue. Así, la región de la imagen cuya ubicación se le provee al algoritmo se utiliza como ejemplo positivo, mientras que el resto de las regiones de la imagen se consideran ejemplos negativos. La nueva ubicación del objeto es aquella en la que la puntuación es máxima, que sirve a su vez para refinar el entrenamiento del algoritmo.
Multiple Instance Learning (MIL) (Babenko, Ming-Hsuan Yang, & Belongie, 2009)	Tiene funcionamiento similar al algoritmo Boosting anteriormente descrito. La diferencia es que, en lugar de considerar sólo la ubicación actual del objeto como un ejemplo positivo, busca en un pequeño vecindario alrededor de la ubicación actual para generar varios ejemplos positivos potenciales, si bien estos no tienen porqué serlo realmente. Así, en MIL se especifican "bolsas" positivas y negativas de ejemplos. En la bolsa positiva, ante la posibilidad que el objeto no esté bien centrado, la consideración de ejemplos vecinos permite que haya una alta probabilidad que al menos uno de los ejemplos lo esté.
MedianFlow (Kalal, Mikolajczyk, & Matas, 2010)	Este algoritmo de *tracking* sigue al objeto tanto hacia adelante como hacia atrás en el tiempo, de forma que mide las discrepancias entre estas dos trayectorias. Minimizar el error entre estas trayectorias le permite

	detectar de forma fiable los fallos de seguimiento y seleccionar aquellas más fiables.
Kernelized Correlation Filters (KCF) (Henriques, Caseiro, Martins, & Batista, 2012)	Este rastreador se aprovecha del hecho de que las muestras positivas utilizadas en el algoritmo de *tracking* MIL tienen grandes regiones superpuestas. Esta superposición de datos conduce a algunas propiedades matemáticas que son explotadas por este algoritmo para hacer que el *tracking* sea más rápido y preciso.
Tracking, learning and detection (TLD) (Kalal, Mikolajczyk, & Matas, 2012)	Como su nombre lo indica, este rastreador descompone la tarea de rastreo en tres componentes: *tracking*, aprendizaje y detección. Así, el *tracking* sigue al objeto fotograma a fotograma, mientras que el detector localiza al objeto con todas las apariencias que se han observado hasta el momento y corrige el *tracking* en caso de que fuera necesario. El módulo de aprendizaje estima los errores del detector y los actualiza para evitarlos en el futuro.
Minimum Output Sum of Squared Error (MOSSE) (Bolme, Beveridge, Draper, & Lui, 2010)	Este algoritmo utiliza filtros de correlación adaptativa para el modelado de la apariencia de los objetivos, mientras que el *tracking* se realiza por medio de la operación de convolución. Así, un objeto se sigue calculando la correlación entre su filtro y una búsqueda en el fotograma siguiente, de forma que la ubicación correspondiente al valor máximo en la salida de correlación indica la nueva posición del objetivo.
CSRT (Lukežič, Vojíř, Čehovin Zajc, Matas, & Kristan, 2018)	Este algoritmo de *tracking* funciona entrenando un filtro de correlación con descriptores como HOG. Este filtro se utiliza para buscar el área alrededor de la última posición conocida del objeto en fotogramas sucesivos.

Tras haber analizado cada uno de estos métodos de *tracking*, se llevó una prueba de concepto sobre cada uno de ellos. El resultado para una de las escenas sobre las cuales se ejecutaron cada uno de los algoritmos es el mostrado en la Ilustración 16. Como bien se puede apreciar, el comportamiento entre los distintos algoritmos es similar. Por ello, y de acuerdo con la prueba de eficiencia, cuyo resultado puede verse en la Tabla 11 del epígrafe de Resultados, el algoritmo que se ha considerado para este módulo ha sido **MOSSE**.

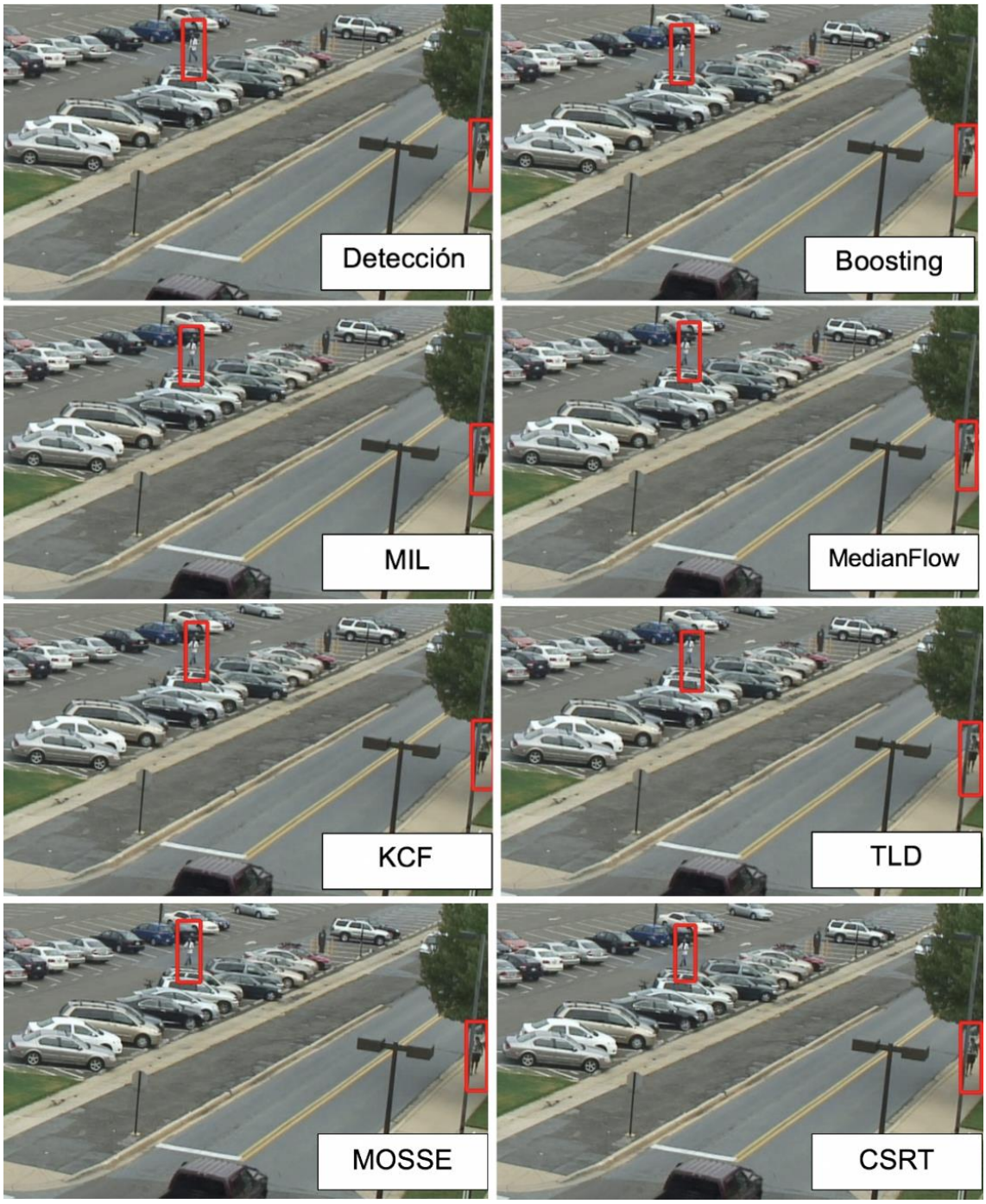

Ilustración 16 Prueba de los *trackers* ligeros considerados

4.3.4 Calculador homográfico

En este módulo se lleva a cabo la transformación entre las ubicaciones de las personas detectadas dentro de la imagen a coordenadas correspondientes al mundo real. Para llevar a cabo esta función, este módulo se ayuda de transformaciones homográficas. En geometría, se denomina **homografía** a toda transformación proyectiva que determina una correspondencia entre dos figuras

geométricas planas, de forma que a cada uno de los puntos y las rectas de una de ellas le corresponden, respectivamente, un punto y una recta de la otra.

En nuestro sistema de vigilancia, consideramos que, por la distancia entre la cámara y el objeto a seguir, se puede suponer que el objeto se mueve en un plano dominante. El **plano dominante** es un área que ocupa el dominio más grande de una imagen. En nuestro caso, consideramos como plano dominante el plano de tierra o suelo (véase la Ilustración 17), que será aquel en el que se proyecten las coordenadas reales del mundo. Así, se puede recurrir a la homografía entre la vista que se obtiene por medio de la cámara y el plano del suelo para obtener una **correspondencia entre puntos de la imagen y coordenadas del mundo real**.

Para estimar la homografía se realiza lo siguiente: Dado un conjunto de puntos correspondientes entre si $x_i \leftrightarrow x_i'$, donde x_i proviene de la vista de la cámara, mientras que x_i' es la coordenada correspondiente a ese punto sobre el plano del suelo y escribiendo $x_i' = (x_i', y_i', w_i')^T$ con coordenadas homogéneas, podemos estimar la matriz de homografía H entre la vista y el plano como $\boldsymbol{x_i' \times Hx_i = 0}$. Así, Para cada par de puntos correspondientes, se escriben tres ecuaciones lineales de la manera mostrada en la Ecuación 1, siendo h_i, $i = 1,2,3$ un vector de 3×1 hecho de los elementos de la íesima fila de H (Zhanfeng, Zhou, & Chellappa, 2004).

Ecuación 1 Cálculo de los elementos de la matriz de homografía

Fuente: (Hartley & Zisserman, 2000)

$$\begin{bmatrix} 0^T & -w_i' x_i & -y_i' x_i \\ w_i' x_i{}^T & 0^T & -x_i' x_i \\ y_i' x_i & x_i' x_i & 0^T \end{bmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0$$

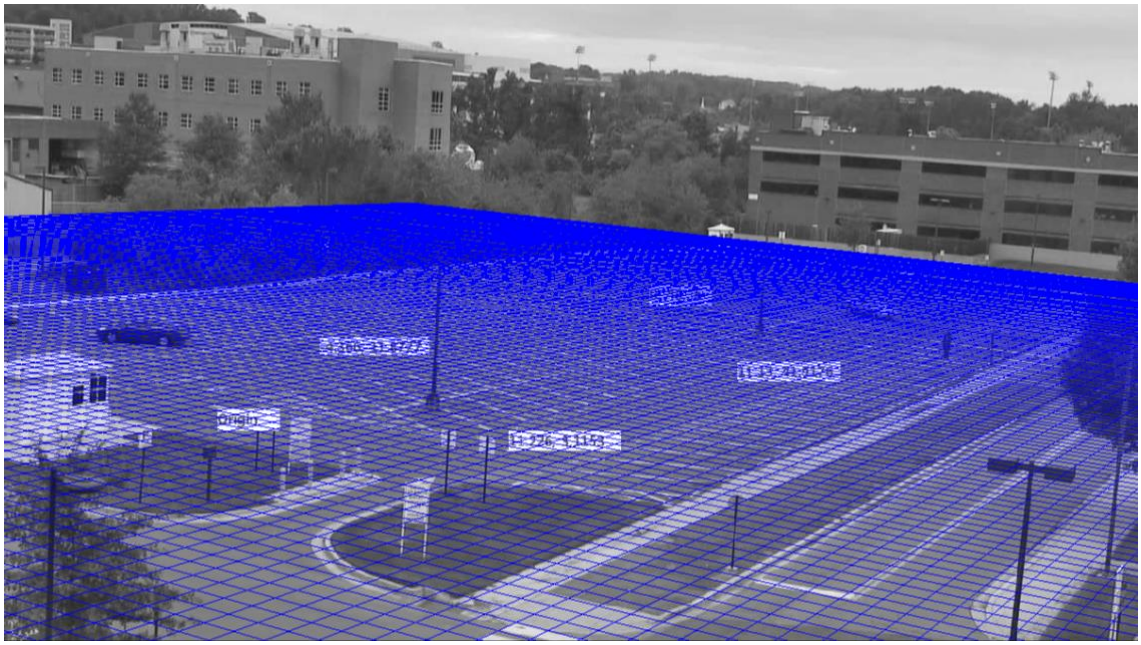

Ilustración 17 Plano dominante para el cálculo homográfico
Fuente: Oh et al., (2011)

Para este módulo y por criterios de temporalidad, hacemos uso de la información homográfica provista en el conjunto de datos VIRAT (Oh et al., 2011), que establece la correspondencia de la ubicación de un píxel de la imagen sobre las coordenadas del plano dominante del suelo. Para llevar a cabo las transformaciones usando la homografía, se hace uso de las funciones provistas por la librería OpenCV (Bradski, 2000), implementadas con dicha finalidad. Del mismo modo, efectuamos transformaciones homográficas sobre toda la imagen de la cámara, a fin de obtener un **plano en vista cenital de la escena**, lo que nos permite visualizar de mejor manera la situación específica con respecto a la instalación de los sujetos que están siendo seguidos. En la Ilustración 18 se muestra el cálculo del plano cenital efectuado por este módulo sobre grabaciones de distintas escenas.

Ilustración 18 Obtención de vista cenital en distintas escenas

4.3.5 *Tracker* de instancias

Las funciones que este módulo tiene dentro del sistema son las siguientes:

- **Mantiene la identidad de las personas**: En todo momento es capaz de mantener cada una de las detecciones de personas identificadas de acuerdo con su apariencia, ubicación y trayectoria. Esto se realiza con el fin de poder relacionar estas detecciones con apariciones anteriores o futuras de los mismos sujetos.

- **Tratamiento de falsos positivos**: Ante objetivos que han sido detectados por primera vez, el *tracking* de instancias se encargará de evaluar si a medida que pasa el tiempo siguen apareciendo, a fin de considerarlos humanos a identificar, o por el contrario resultan detecciones provenientes de falsos positivos y que, por lo tanto, han de ser ignorados.

- **Contingencia ante fallos en la detección de personas:** En caso de que alguno de los sujetos humanos que se están siendo seguidos y aun estando expuestos en la escena, no fueran detectados por el módulo del sistema encargado de dicha tarea, este *tracker* se encarga de predecir la ubicación de dicho sujeto no detectado y dar fe de que sigue estando presente en la escena. Del mismo modo y ante una **oclusión parcial o total** de un sujeto que deriva en la imposibilidad de su detección, este módulo y la predicción de la trayectoria de las personas que ejecuta permite mostrar su ubicación más probable.

Así, las funciones que anteriormente se han descrito se encuentran implementadas en módulos concretos dentro del *tracker* que se propone en este proyecto. En la Ilustración 19 se muestra el esquema de funcionamiento del *tracker* a raíz de la detección de uno o varios sujetos humanos, de forma que se muestran los componentes que lo conforman y la interacción que hay entre ellos.

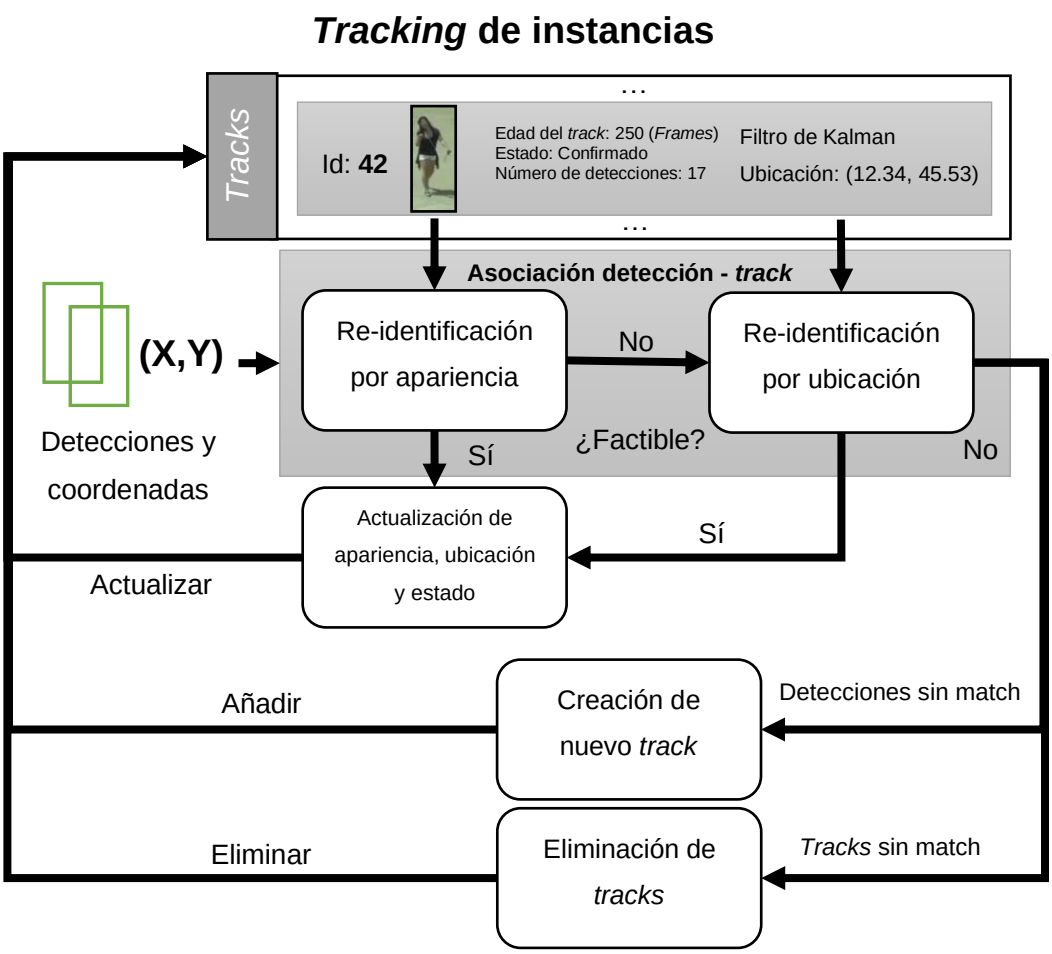

Ilustración 19 Funcionamiento y módulos del *tracking* de instancias

De acuerdo con el esquema anterior, puede observarse como el funcionamiento de el *tracking* de instancias se inicia con aquellas detecciones de personas que se han realizado y las coordenadas provistas por el calculador homográfico. A partir de entonces se inicia un proceso, cuya primera fase consta de **la asociación de dichas detecciones con alguno de los *tracks* ya guardados**. Esta asociación se puede realizar de dos formas:

1. **Re-identificación por apariencia**: Con las imágenes de las detecciones se lleva a cabo una búsqueda en la base de datos de aquellos *tracks* con apariencia similar.

2. **Re-identificación por ubicación**: Aquellas detecciones que no se han podido emparejar por medio de la apariencia, se asocian por medio de las coordenadas de la ubicación de las detecciones y las de los *tracks*.

En caso de que las detecciones hayan podido emparejarse con los *tracks* de la base de datos con alguno de los métodos anteriores, se **actualiza** dicho *track* con la ubicación actual y su apariencia dentro de la base de datos, a fin de poder refinar el emparejamiento para futuras ocasiones. En caso de que no se haya podido llevar a cabo el emparejamiento para alguna detección o *track*, se llevan a cabo alguna de estas dos opciones:

- En caso de que se trate de **detecciones que no se corresponden con ningún *track***, estas han de ser guardadas para futuras posibles detecciones del mismo objeto. Por ello, se crea un nuevo *track* con misma apariencia y ubicación a la detección, dotándolo de un identificador propio.
- Si son ***tracks* que no han podido ser emparejados con las detecciones** actuales, ha de ser evaluado si han de borrarse los *tracks*, pues puede que el sujeto humano seguido hasta el momento haya dejado de encontrarse en escena.

A continuación, se especifica la implementación realizada de cada uno de estos módulos presentados.

4.3.5.1 Re-identificación por medio de la apariencia

Para llevar a cabo la asociación entre detecciones y *tracks* por medio de la apariencia, se considera una **métrica de similaridad** entre un descriptor de características de la imagen correspondiente a la persona detectada y el historial de descriptores de cada uno de los *tracks*. Este procedimiento es el propuesto por Wojke, Bewley, & Paulus (2017). Estos autores proponen para la obtención de un descriptor de cada una de las imágenes de las personas el uso de una red neuronal convolucional, en lo que definen como un **descriptor de apariencia profundo**. Para su cálculo, hacen uso de un modelo de red con la arquitectura presentada en la Ilustración 20 y que previamente ha sido entrenado sobre **1.100.000 imágenes de 1.261 peatones distintos**. Tal y como se puede apreciar, el descriptor generado (salida de dicha red) tiene un tamaño de 128. Así, nosotros haremos uso de este modelo para obtener un descriptor de apariencia fiable.

Name	Patch Size/Stride	Output Size
Conv 1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv 2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

Ilustración 20 Arquitectura de la CNN para el descriptor profundo
Fuente: Wojke, Bewley, & Paulus (2017)

A partir de la obtención del descriptor para cada una de las imágenes de las detecciones, se puede establecer una similaridad entre estos y los descriptores de los *tracks* guardados. Esta similaridad, en nuestro caso será calculada por medio de la medida de la **similitud coseno**, cuya fórmula se muestra en la Ecuación 2. Del mismo modo, en la Ilustración 21 se enseña un ejemplo real del cálculo de esta medida para la re-identificación de personas detectadas en el conjunto de datos VIRAT (Oh et al., 2011).

Ecuación 2 Fórmula de la similitud coseno

$$sim(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

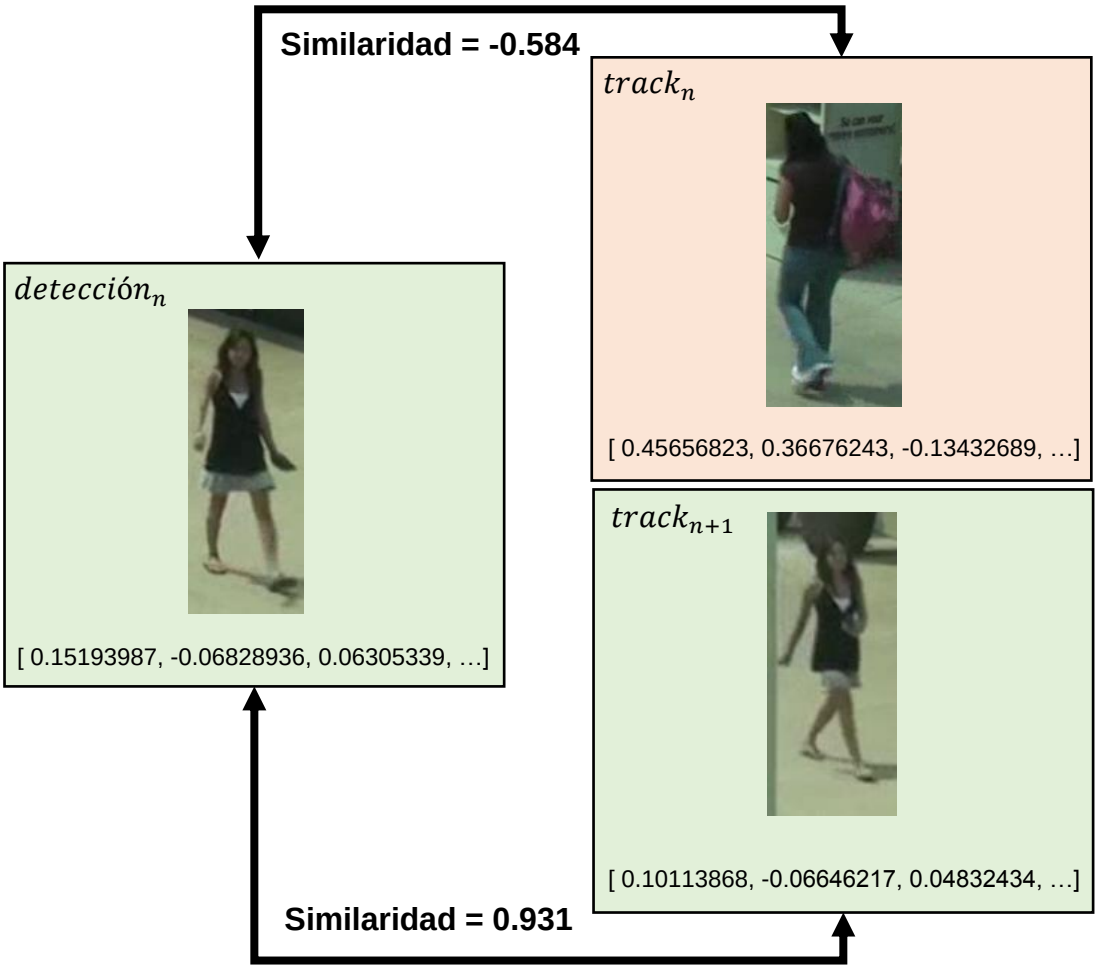

Ilustración 21 Ejemplo de re-identificación por apariencia

Una vez que se dispone de una medida de similaridad entre los descriptores de las detecciones y los de los *tracks*, el problema será la asignación entre pares de detección – *track*. Así, a raíz de la similitud, invertimos dicho valor para generar una matriz de costes y aplicamos el ***algoritmo húngaro*** (Kuhn, 1955) para llevar a cabo la asignación. Así, este algoritmo modela un problema de asignación como una matriz de costes de $n\,x\,m$, donde cada elemento representa el coste de asignar la enésima detección al emésimo *track*. Adicionalmente, aquellas asignaciones potenciales cuyo valor de coste supere un umbral preestablecido, no se realizarán, quedando la detección y el *track* en cuestión pendientes de asignar.

4.3.5.2 Re-identificación por medio de la ubicación

En caso de que **no fuera posible** llevar a cabo la asignación por medio de la **re-identificación a partir de la apariencia**, se lleva a cabo la misma tarea, pero considerando las **coordenadas de la ubicación actual de cada una de las**

detecciones a asignar y las coordenadas que se han predicho de cada uno de los *tracks* para el momento en cuestión, dado sus trayectorias y velocidades. Esta predicción de la ubicación de cada una de los *tracks* se lleva a cabo por medio de un filtro de Kalman que guarda cada uno de los *tracks*. Su funcionamiento se muestra en el apartado 4.3.5.3 de esta memoria. Con las coordenadas de la ubicación actual de las detecciones y la predichas para cada uno de los *tracks*, se lleva a cabo una medición del coste entre ambas, por medio de la distancia euclídea, cuya fórmula se muestra en la Ecuación 3.

Ecuación 3 Fórmula general de la distancia euclídea entre dos puntos

$$d(P,Q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

Al igual que en el caso de la re-identificación por medio de la apariencia, se utiliza el **algoritmo húngaro** para la asignación entre pares de detección – *track*, mientras que el coste de aquellas potenciales asignaciones que superen un umbral preestablecido hará que estos queden pendientes de asignar.

4.3.5.3 Actualización de apariencia, ubicación y estado

Sobre aquellas detecciones que hayan sido identificadas dentro de la base de datos de *tracks* por medio de alguno de los dos métodos anteriormente presentados, se lleva a cabo la actualización de la apariencia, ubicación y estado a fin de realizar un refinamiento en estas características que **facilite la re-identificación de los *tracks* en futuras ocasiones**. Cada una de las tareas que este módulo realiza son descritas a continuación:

- **Actualización de apariencia**: Se añade el descriptor de la última imagen en la que se ha detectado al *track* en cuestión, a fin de que esta se encuentre lo más actualizada posible.

- **Actualización de la ubicación**: A raíz de la última ubicación conocida del *track* detectado y la información previa de la trayectoria y su velocidad, se lleva a cabo la predicción de la ubicación más probable que tendrá el sujeto una vez vuelva a ser detectado. Este cálculo de la ubicación por medio de la información de la trayectoria y velocidad se lleva a cabo

mediante un **filtro de Kalman** (Kalman, 1960). Este filtro se basa en las características estadísticas del ruido del sistema y del ruido de medición, de forma que utiliza las variables medidas se utilizan como señal de entrada, y las variables estimadas que necesitamos conocer son la salida del filtro. Todo el proceso de filtrado se compone de una fórmula para la predicción y otra para la actualización, que son mostradas respectivamente en la

- Ecuación 4 y en la Ecuación 5.

Ecuación 4 Kalman Filter: Fórmula para predicción
Fuente: Kalman, (1960)

$$\boldsymbol{X(n) = F \cdot X(n-1) + V_q(n-1)}$$

Ecuación 5 Kalman Filter: Fórmula para actualización
Fuente: Kalman, (1960)

$$Y(n) = H \cdot X(n) + V_p(n)$$

Donde $X(n)$ e $Y(n)$ son la variable de estado estimada y la medición en el momento n, respectivamente. F es la matriz de la transición de estados y H la matriz de mediciones. Por su parte, $V_q(n)$ y $V_p(n)$ representan el ruido del sistema y el ruido de las mediciones, respectivamente. Así el proceso que se sigue con el filtro de Kalman para llegar a cabo las predicciones y actualizaciones es el mostrado en la Ilustración 22.

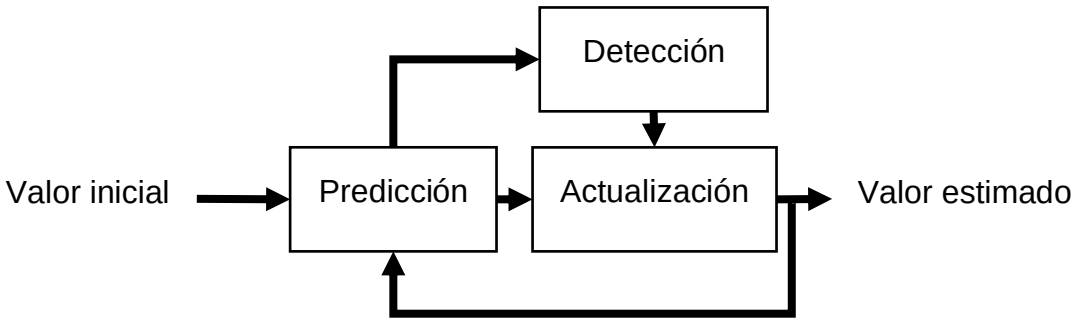

Ilustración 22 Flujo de ejecución del filtro de Kalman

- **Actualización del estado**: Para tener conocimiento de si un determinado *track* se encuentra en escena, ha dejado de estarlo o bien se trata de una detección de la cual aún no tenemos confirmación, se han determinado tres estados distintos:

- **Detectado**: En este estado se encuentran todas aquellas detecciones que se han identificado de forma inferior a un cierto número de veces. Esto sirve para poder lidiar con falsos positivos que el módulo de detección arroje y para tener un cierto conocimiento de las características de los *tracks* antes de realizar su seguimiento (trayectoria, apariencia, etc.).

- **Confirmado**: Este estado se corresponde con todos aquellos *tracks* que han sido detectados en un cierto número de veces como mínimo y que, por lo tanto, se encuentran actualmente en escena.

- **Expirado**: En este estado se encuentran aquellos *tracks* que no han sido identificado durante un cierto número de fotogramas consecutivos. Así, se interrumpe su seguimiento hasta que nuevamente se detecten en la escena.

5 Resultados

Este epígrafe comprende los resultados obtenidos tras las diversas pruebas de carácter cualitativo y cuantitativo realizadas y la muestra del estado final en que se encuentra el sistema implementado una vez finalizado el proyecto.

Así, si en apartados anteriores de este trabajo se han mostrado resultados de pruebas de concepto realizadas sobre distintas soluciones para cada uno de los módulos, en este epígrafe se presentan aquellas métricas que han dado pie a cada una de las decisiones tomadas durante el desarrollo del trabajo. Del mismo modo, en este apartado se analiza el desempeño del sistema en su conjunto una vez ha sido puesto a prueba sobre un conjunto de datos real de videovigilancia.

En cuanto a las pruebas de carácter cualitativo, en este epígrafe se valoran distintas situaciones potencialmente difíciles para un sistema de videovigilancia. Así, se exponen aquellas situaciones complejas en las que el sistema acierta y en las que falla, y se da una explicación al porqué de dicho comportamiento.

Por último, se presenta cual es el estado final y la apariencia del sistema una vez finalizado el proyecto.

5.1 Métricas de calidad

En primer lugar, se presentan cada una de las métricas de calidad que han sido posible extraer y que han resultado aclaradoras para facilitar la toma de decisiones acerca de qué soluciones escoger para cada uno de los módulos del sistema.

5.1.1 Resultados la selección de regiones de interés (ROIs)

En el epígrafe del Identificador de regiones de interés (ROI) se analizaron cada uno de los algoritmos de extracción de fondos considerados y se llevo a cabo una comparativa a nivel funcional de cada uno de ellos. Junto a esto, se ha obtenido la tasa fotogramas por segundo (FPS) para cada uno de estos algoritmos, en una prueba realizada para este proyecto sobre varias tomas del conjunto de datos de videovigilancia VIRAT (Oh et al., 2011). La tasa de fotogramas pone de manifiesto la eficiencia de cada uno de estos algoritmos, aspecto crucial a la hora de elegir uno de ellos para este módulo. En la Tabla 8

se muestra el resultado para cada uno de los algoritmos. El *hardware* utilizado para estas pruebas es el especificado en el apartado del Ordenador de sobremesa de esta misma memoria. Se puede apreciar una gran diversidad de rendimientos entre los distintos algoritmos, siendo necesario escoger alguno de ellos que garantice con margen el requisito fundamental del funcionamiento en tiempo real.

Tabla 8 Prueba de eficiencia de los algoritmos de extracción de fondos

Algoritmo	FPS en prueba
Mixture of Gaussians (MOG) (KaewTraKulPong & Bowden, 2002)	33.35
MOG2 (Zivkovic & Van Der Heijden, 2006)	125.12
GMG (Godbehere et al., 2012)	23.80
Local SVD Binary Pattern (LSBP) (Guo et al., 2016)	11.12
KNN (Zivkovic & Van Der Heijden, 2006)	124.16

5.1.2 Resultados del módulo de clasificación de personas

A continuación, se exponen los resultados obtenidos para el módulo del clasificador de personas. En primer lugar, se muestran las métricas de calidad obtenidas para el uso del descriptor HOG y el algoritmo SVM y seguidamente las métricas de cada uno de los modelos CNN entrenados. Al tratarse de un problema de clasificación binaria (Persona y no persona), se utilizarán las métricas propias para estos tipos de problemas.

5.1.2.1 Resultados de HOG + SVM

A fin de encontrar aquel modelo SVM con mejor configuración de parámetros, se llevó a cabo el entrenamiento y prueba de las distintas combinaciones de diversos valores de estos. Los valores de los parámetros considerados son los que se muestran en la Tabla 5. Así, en la Tabla 9 se muestran los resultados de las mejores combinaciones de parámetros que se han evaluado, ordenados por su *F1-score*. Tal y como se puede observar en esta tabla, tanto la elección de un kernel gaussiano como uno polinomial garantiza unos resultados similares. En cualquier caso, los mejores resultados se han conseguido para un valor de *Gamma* de 0.01. En el caso del kernel polinomial, un valor de *C* bajo (5 y 10) y un grado de 4 garantizan los mejores resultados. Por su parte y para los valores

de C escogidos, el kernel gaussiano obtiene resultados similares en la mayoría de sus combinaciones.

Las mejores ejecuciones del kernel polinomial consiguen los mejores resultados de precisión, mientras que las mejores del Gaussiano lo hacen en la exhaustividad (*recall*).

Tabla 9 Resultados para HOG + SVM

Parámetros	Mean accuracy	Mean precision	Mean recall	Mean f1
{'C': 5, 'degree': 4, 'gamma': 0.01, 'kernel': ‘poly’}	0.965925	0.971243	0.925070	0.947584
{'C': 10, 'degree': 4, 'gamma': 0.01, 'kernel': ‘poly’}	0.965569	0.973339	0.921859	0.946885
{'C': 5, 'degree': 3, 'gamma': 0.01, 'kernel': ‘poly’}	0.965141	0.966185	0.927853	0.946603
{'C': 5, 'degree': 5, 'gamma': 0.01, 'kernel': ‘poly’}	0.965284	0.974377	0.919932	0.946361
{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}	0.964785	0.961590	0.931492	0.946280
{'C': 50, 'gamma': 0.01, 'kernel': 'rbf'}	0.964785	0.961590	0.931492	0.946280
{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.964642	0.961167	0.931492	0.946075
{'C': 10, 'degree': 3, 'gamma': 0.01, 'kernel': ‘poly’}	0.964713	0.967150	0.925498	0.945849
{'C': 5, 'gamma': 0.01, 'kernel': 'rbf'}	0.964214	0.961925	0.929351	0.945337
{'C': 10, 'degree': 3, 'gamma': 0.1, 'kernel': ‘poly’}	0.964428	0.969433	0.922286	0.945255
{'C': 50, 'degree': 3, 'gamma': 0.1, 'kernel': ‘poly’}	0.964428	0.969433	0.922286	0.945255
{'C': 5, 'degree': 3, 'gamma': 0.1, 'kernel': ‘poly’}	0.964428	0.969433	0.922286	0.945255
{'C': 1, 'degree': 3, 'gamma': 0.1, 'kernel': ‘poly’}	0.964428	0.969433	0.922286	0.945255
{'C': 100, 'degree': 3, 'gamma': 0.01, 'kernel': ‘poly’}	0.964428	0.969433	0.922286	0.945255
{'C': 100, 'degree': 3, 'gamma': 0.1, 'kernel': ‘poly’}	0.964428	0.969433	0.922286	0.945255
{'C': 50, 'degree': 3, 'gamma': 0.01, 'kernel': ‘poly’}	0.964286	0.969003	0.922286	0.945048
{'C': 10, 'degree': 5, 'gamma': 0.01, 'kernel': ‘poly’}	0.964428	0.974306	0.917363	0.944962
{'C': 5, 'degree': 4, 'gamma': 0.1, 'kernel': ‘poly’}	0.964072	0.972349	0.918219	0.944493
{'C': 100, 'degree': 4, 'gamma': 0.1, 'kernel': ‘poly’}	0.964072	0.972349	0.918219	0.944493
{'C': 50, 'degree': 4, 'gamma': 0.1, 'kernel': ‘poly’}	0.964072	0.972349	0.918219	0.944493
{'C': 100, 'degree': 4, 'gamma': 0.01, 'kernel': ‘poly’}	0.964072	0.972349	0.918219	0.944493
{'C': 10, 'degree': 4, 'gamma': 0.1, 'kernel': ‘poly’}	0.964072	0.972349	0.918219	0.944493
{'C': 1, 'degree': 4, 'gamma': 0.1, 'kernel': ‘poly’}	0.964072	0.972349	0.918219	0.944493
{'C': 50, 'degree': 4, 'gamma': 0.01, 'kernel': ‘poly’}	0.964001	0.972131	0.918219	0.944390
{'C': 1, 'degree': 5, 'gamma': 0.01, 'kernel': ‘poly’}	0.963787	0.973593	0.916078	0.943945

5.1.2.2 Resultado de modelos CNN

En la Tabla 6 se presentaron los modelos de CNN más importantes que fueron entrenados por medio de *Transfer Learning* y sus características. Entre ellas se incluyeron aspectos como el número de parámetros del modelo, que está directamente proporcionado al espacio en memoria que este ocupará. Así, a fin de lograr los objetivos de eficiencia que se han impuesto a este proyecto, esto se presenta como fundamental, de forma que conjuntamente a las métricas de clasificación, son los dos aspectos sean quienes determinen que modelo se escogerá para el módulo de la clasificación de personas.

En la Tabla 10 se muestran las métricas de clasificación obtenidas para cada uno de estos modelos entrenados cogiendo como base *MobileNetV2* (Sandler et al., 2018) y los parámetros de dicha arquitectura, entrenados con el conjunto de datos *ImageNet* (Jia Deng et al., 2009).

Tabla 10 Resultados principales modelos CNN

Modelo	Mean accuracy	Mean precision	Mean recall	Mean f1
MNV2_1	0.973024	0.944642	0.983013	0.963446
MNV2_2	0.978900	0.981859	0.959380	0.970489
MNV2_3	0.977030	0.975976	0.960118	0.967982
MNV2_4	0.974626	0.972953	0.956425	0.964618
MNV2_5	0.976763	0.977393	0.957903	0.967549

Tal y como se observa en la tabla anterior, en general, todas las pruebas que se muestran no obtuvieron malos resultados, considerándose de primeras mejor opción para nuestro sistema la utilización de un modelo de red neuronal convolucional que la aproximación de HOG con SVM. En la Ilustración 23 se muestran las curvas ROC para cada uno de estos modelos considerados anteriormente. Finalmente, **el modelo que se ha elegido para usarse en el clasificador de personas ha sido el MNV2_5**. Esto se ha debido a que tal y como se mostró en la Tabla 6, se trata de un modelo con bajo número de parámetros y, por ende, un modelo que ocupa menos memoria. A su vez, esta decisión queda reforzada con las métricas anteriores al observar su buen balance entre precisión y exhaustividad.

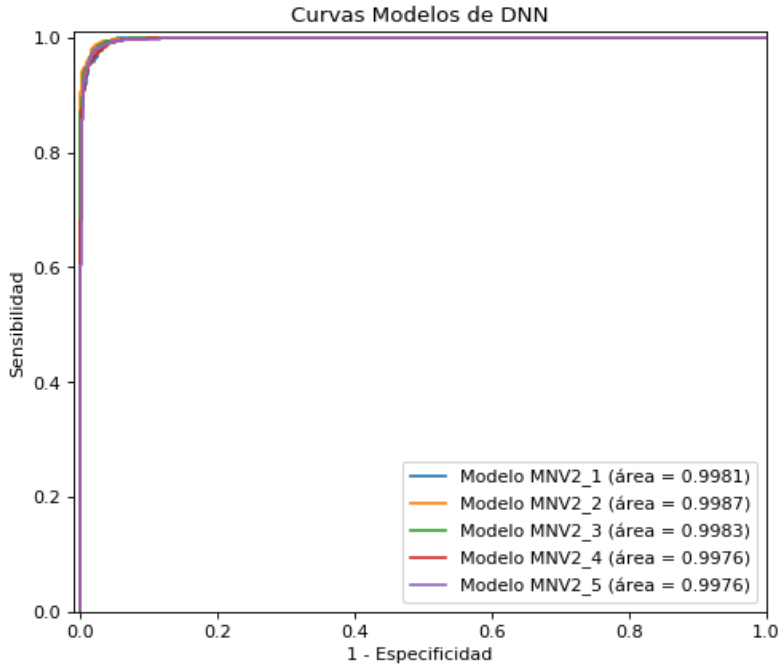

Ilustración 23 Curva ROC de modelos CNN para clasificación de personas

5.1.3 Resultados del módulo del tracker ligero

En el epígrafe del módulo del *Tracker* ligero, se analizaron cada uno de los algoritmos de *tracking* existentes dentro de las bibliotecas consideradas. Así, se llevo a cabo una comparativa a nivel funcional de cada uno de ellos. Junto a esto, se ha obtenido la **tasa de fotogramas por segundo** (FPS) para cada uno de estos algoritmos, en una prueba realizada para este proyecto sobre varias tomas de los conjuntos de datos de videovigilancia VIRAT (Oh et al., 2011) y el conjunto de datos de *Oxford Town Centre* (Harvey Adam. LaPlace, 2019). La tasa de fotogramas pone de manifiesto la eficiencia de cada uno de estos algoritmos, aspecto crucial a la hora de elegir uno de ellos para este módulo. En la Tabla 8 se muestra el resultado para cada uno de los algoritmos. El *hardware* utilizado para estas pruebas es el especificado en el apartado del Ordenador de sobremesa de esta misma memoria. Se puede apreciar una gran diversidad de rendimientos entre los distintos algoritmos, siendo necesario escoger alguno de ellos que garantice con margen el requisito fundamental del funcionamiento en tiempo real.

Tabla 11 Prueba de eficiencia de los algoritmos de extracción de tracking

Algoritmo	FPS en prueba
Boosting (Grabner et al., 2006)	27.02
Multiple Instance Learning (MIL) (Babenko et al., 2009)	16.64
MedianFlow (Kalal et al., 2010)	83.33
Kernelized Correlation Filters (KCF) (Henriques et al., 2012)	52.63
Tracking, learning and detection (TLD) (Kalal et al., 2012)	9.02
Minimum Output Sum of Squared Error (MOSSE) (Bolme et al., 2010)	322.58
CSRT (Lukežič et al., 2018)	20.01

Teniendo en consideración el requisito de funcionamiento en tiempo real de nuestro sistema, se ha de escoger cualquiera de los algoritmos que funcionen con una tasa de refresco mayor a 30 fotogramas por segundo, teniendo que dejar margen adicional para el cálculo del resto de tareas del sistema.

5.1.4 Resultados del sistema integrado

Una vez evaluados cada uno de los módulos de forma separada, se ha llevado a cabo una **prueba sobre el sistema en su totalidad** y para el conjunto de datos VIRAT (Oh et al., 2011). Esta prueba, por lo tanto, supone una evaluación colectiva de todos los módulos del sistema, valorando la aportación individual de cada módulo y su interacción con el resto del sistema.

Por medio de la elección de diversas métricas de calidad, se ha querido utilizar un marco de referencia que nos permita evaluar el rendimiento y que pueda ser comparable con otros métodos futuros. Actualmente, **no se tiene constancia de la existencia de aproximaciones similares a la propuesta** en este proyecto sobre el conjunto de datos en cuestión. Así, hemos utilizado entre otras, las métricas propuestas por Bernardin & Stiefelhagen (2008), evaluando aspectos como: falsos positivos y negativos, alineamiento de la ubicación estimada de los *tracks* respecto a la real, interrupciones del seguimiento, identificaciones erróneas de *tracks*. Todas las métricas que han sido consideradas son las siguientes:

1. Exactitud del seguimiento de múltiples objetos **(*Multiple Object Tracking Accuracy* - MOTA)**: Proporción media de la exactitud en términos de

falsos positivos, falsos negativos y cambios en la identidad de las personas.

2. Precisión del seguimiento de múltiples objetos **(*Multiple Object Tracking Precision* - MOTP)**: Proporción media de la precisión en la estimación de la posición de lo sujetos seguidos.
3. Mayoritariamente seguidos **(*Mostly Tracked* - MT)**: Porcentaje de *tracks* que tienen la misma etiqueta durante al menos el 80% de su vida.
4. Mayoritariamente perdidos **(*Mostly Lost* - ML)**: Porcentaje de *tracks* que son seguidos durante menos del 20% de su vida.
5. Cambios en la identidad **(*Identity Switches* – ID Sw)**: Número de veces que cambia la identidad de un *track*.
6. Fragmentación **(*Fragmentation* - FM)**: Número de veces que un *track* es interrumpido.
7. Falsos positivos **(*False Positives* – FP)**: Porcentaje de detecciones consideradas erróneamente.
8. Falsos negativos **(*False Negatives* – FN)**: Porcentaje de fallos en la detección de personas que si lo son.
9. Hz: Velocidad de procesamiento en fotogramas por segundo (FPS).

En la Tabla 12 se muestran todas las métricas de calidad probadas para nuestro sistema sobre el conjunto de datos de VIRAT (Oh et al., 2011) (ver apartado correspondiente a Conjuntos de datos). Cada una de estas medidas han sido calculadas de acuerdo con el **intervalo de fotogramas cada el cual se ejecuta la tarea de detección**. Se ha tenido esta consideración por la importancia de la **búsqueda del balanceo o equilibrio entre precisión y velocidad**. Al tratarse de una tarea computacionalmente más costosa, la detección no debe ejecutarse de forma continuada si queremos un sistema más eficiente. Sin embargo, también es cierto que la precisión provista por la detección es mayor que la estimación que pueda realizar el algoritmo de *tracking*.

Tabla 12 Métricas de rendimiento del sistema

Intervalo Detección-Tracking (*Frames*)	MOTA(1)	MOTP(2)	MT(3)	ML(4)	ID Sw(5)	FM(6)	FP	FN	Hz
0	79.23%	86.27%	89.02%	9.21%	154	361	**2.24%**	**2.56%**	5.21
2	80.64%	87.13%	89.53%	9.10%	157	310	2.33%	2.73%	11.67
4	81.11%	87.02%	91.51%	8.51%	**148**	298	2.37%	2.85%	21.49
6	**81.32%**	86.96%	91.79%	8.46%	149	270	2.91%	2.89%	26.34
8	81.28%	**87.17%**	91.67%	8.42%	152	267	2.94%	2.92%	36.26
10	80.02%	86.79%	**92.18%**	8.78%	164	**228**	3.04%	2.98%	43.38
12	79.86%	86.82%	90.62%	**8.32%**	170	232	3.17%	3.09%	50.71
14	78.54%	86.76%	86.27%	8.46%	198	254	3.28%	3.14%	56.54
20	76.73%	84.47%	86.13%	9.33%	254	390	3.35%	3.43%	70.15

5.1.4.1 Elección del intervalo de ejecución de la detección

La elección de un intervalo cada cual se ejecuta la detección y que, por ende, se actualizan los *tracks* se presenta **fundamental para evaluar el sistema en términos de eficiencia y precisión**. Como bien se puede observar en la Tabla 12, se evalúan diversidad de intervalos de distinto tamaño para el uso de la detección. Aquel que tiene un **intervalo de 0**, se corresponde con el uso de la detección en todos los fotogramas, llevando a cabo el ***tracking* por medio de la detección** exclusivamente. Esta opción es la que garantiza **un menor número de falsos positivos y falsos negativos**. Esto es debido a la constante comprobación de las identidades de las clases de los objetos que se aparecen en escena, pudiendo detectar su aparición desde del primer fotograma en que lo hacen. Del mismo modo, si dichos objetos desaparecen, se tiene constancia desde un primer momento. Contrariamente, si se tiene un mayor intervalo de fotogramas entre ejecuciones del detector, se considerarán de forma más tardía nuevos objetivos en la escena que seguir, y se mantendrá el seguimiento de otros que ya no están.

Aquellas ejecuciones con **intervalos en la detección entre 8 y 12** alcanzan los **mejores resultados en cuanto a *tracks* mayoritariamente seguidos (MT) y perdidos (ML)**. Esto se puede deber a las bondades del algoritmo de *tracking*, pues se ejecuta durante más tiempo que en las ejecuciones en las que la detección se realiza en intervalos inferiores. Esto resulta en que se puede efectuar una estimación durante más tiempo de la ubicación de los *tracks* en situaciones de oclusión parcial o total, por lo que se minimiza el efecto de una detección fallida. Así, por ejemplo, ante la situación de dos personas que se cruzan y en la que queda una parcial o totalmente ocluida, el detector considera a una única persona o incluso a ninguna. Por su parte, el *tracking* mantiene el foco sobre cada uno de los sujetos al encontrarse y considerando la estimación de su ubicación, permite manejar la oclusión. Por la misma razón estas ejecuciones también consiguen un **menor número en la fragmentación de los *tracks* (FM)**, permitiendo no interrumpir el seguimiento de una persona por una situación anómala, en la que esta no haya sido detectada por un corto periodo de tiempo.

Como el tracking de instancias se ejecuta para todos los fotogramas que llegan al sistema, los resultados en la **métrica de cambio en la identidad van a ser similares siempre que la detección se ejecute en intervalos cortos o medios**. Esto se debe a que el error acumulado de no actualizar el *tracking* puede llevar a que se dejen de seguir a sujetos que nuevamente detectados puedan ser considerados con nuevas identidades.

Así, en general, las **ejecuciones con intervalos de tamaño de 4 a 8 consiguen la mejor exactitud y precisión en el seguimiento de múltiples objetos**. Sin embargo, hemos de considerar el intercambio de precisión y velocidad anteriormente comentado. Aquellas **ejecuciones con un intervalo igual o mayor a 8** consiguen que su velocidad de ejecución **garantice el requisito del tiempo real** en vídeos de alta calidad (más de 30 fotogramas por segundo). Así, se pueden considerar las ejecuciones con un intervalo en la ejecución de la detección de valor 8, 10, 12 o incluso 14, de acuerdo con las características del sistema *hardware* en que este software quiera ser ejecutado. En el caso del sistema en el que se han realizado las pruebas, un intervalo de 8 fotogramas es suficiente para garantizar el procesamiento en tiempo real (funcionamiento a 36.26 FPS en las prueba). Para un sistema *hardware* con recursos más limitados, este intervalo será necesariamente mayor.

5.2 Comportamiento del sistema ante problemas potenciales

En este apartado de los resultados mostraremos la respuesta del sistema ante situaciones comúnmente consideradas como complejas para problemas de videovigilancia. Por ello, consideramos grabaciones de situaciones en la que personas quedan ocluidas, bien por la influencia de otras personas, o por otro obstáculo en escena. Del mismo modo, se evalúa el comportamiento del sistema ante la aparición de otros objetos en escena distintos a personas.

5.2.1 Oclusiones

Cuando el sistema se encuentra ante oclusiones parciales o totales de un sujeto humano puede actuar de distinta forma de acuerdo con la naturaleza de la oclusión que se de.

5.2.1.1 Tipos de oclusiones

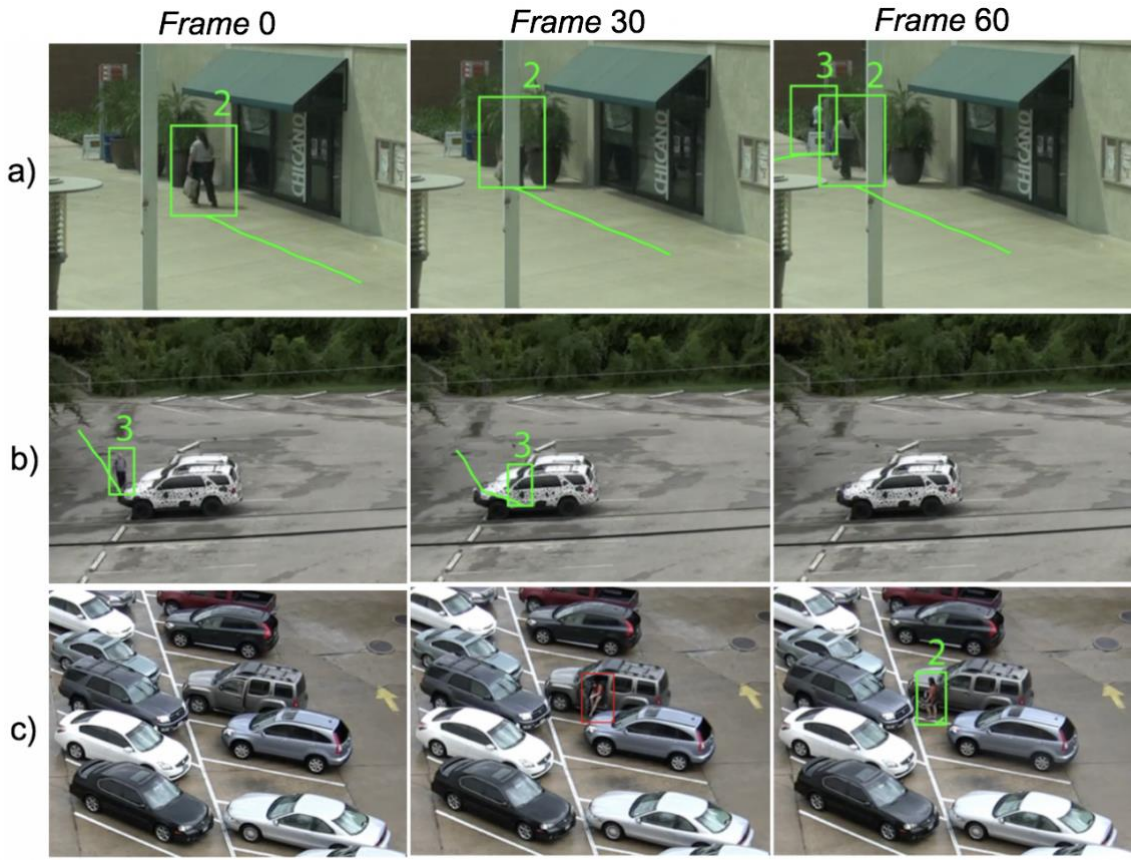

Ilustración 24 Comportamiento del sistema ante tipos de oclusiones

En la Ilustración 24 se muestran las tres situaciones distintas en las que se puede encontrar un sujeto humano ocluido y como el sistema reacciona a ellas:

a) ***Track* temporalmente ocluido que reaparece**: En primer lugar, se muestra el caso de aquel *track* que, habiendo estado rastreado, la visión de este resulta obstaculizada durante un corto periodo de tiempo. En este caso, el sistema es capaz de estimar su ubicación por medio del filtro de Kalman (véase apartado correspondiente a Actualización de apariencia, ubicación y estado), utilizando la información de su trayectoria y velocidad. Una vez estando nuevamente expuesto a la cámara, el sistema re-identifica al sujeto por medio de la información de su apariencia o por su ubicación,

b) *Track* ocluido que desaparece: Si bien en una oclusión el sujeto monitorizado pudiera aparecer nuevamente, también puede darse el caso que este desaparezca de la escena definitivamente. Este caso el sistema, al igual que en la situación anterior, intentará primeramente estimar la ubicación del sujeto mientras, a fin de que pueda ser re-identificado en el corto plazo. Sin embargo, si después de un cierto número de fotogramas (establecido en este sistema en 30 fotogramas), no se detecta al sujeto en cuestión, se pausará su seguimiento.

c) Sujeto humano ocluido que aparece: Esta situación considera a aquel sujeto humano que ha permanecido ocluido durante un periodo largo de tiempo. Esta situación es similar al caso en que una persona entrara en escena por primera vez. Así, una vez expuesto a la cámara, el sistema detecta la presencia de un objeto en movimiento, que permanecerá en estado de detección durante un cierto de número de fotogramas (establecido en este sistema en 15 fotogramas). Una vez pasado ese tiempo, si sigue detectándose su presencia, se re-identifica al sujeto en caso de que ya haya visto anteriormente. En caso de que sea su primera aparición, se crea un nuevo *track* y se le dota de identidad.

5.2.1.2 Fallos en el manejo de oclusiones

Si bien en el apartado anterior se definen los tipos de oclusiones existentes y como el sistema se encarga de manejarlas, hay ocasiones en las que por falta de información o por la dificultad de la escena, no se pueden hacer frente a ellas de forma satisfactoria. Sin embargo, es posible minimizar sus efectos. Por ello,

a continuación, en la Ilustración 25 se muestran ejemplos en los que el sistema falla al manejar las oclusiones y como tras ello, actúa para su rearme.

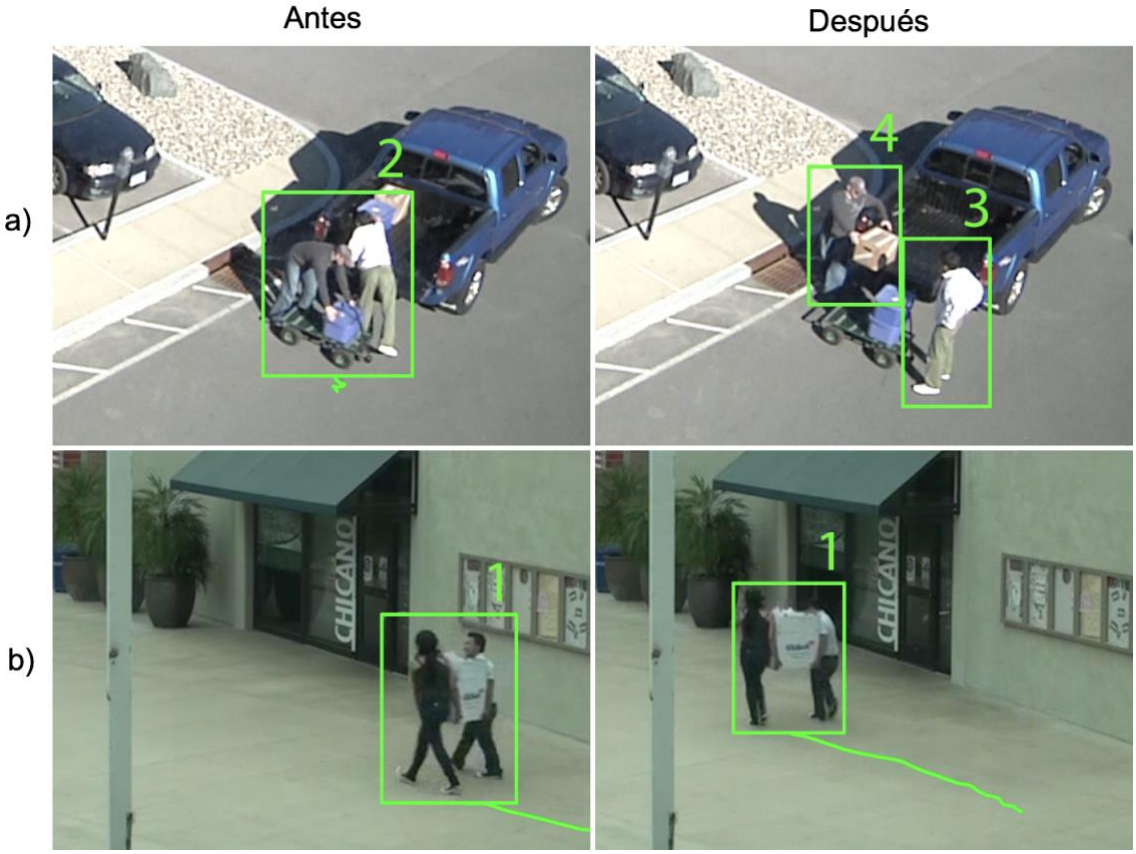

Ilustración 25 Fallos del sistema al manejar oclusiones

Así, a continuación, se valoran cada uno de estos fallos y la reacción posterior del sistema:

a) **Dos o más personas que se encuentran inicialmente juntas**: Esta situación se da cuando al aparecer en escena, dos o más personas lo hacen de manera conjunta, solapándose los unos a los otros. El detector considera la región en la que se encuentran las estas personas como un único individuo al que empieza a seguir. Cuando el grupo se disperse o se separen lo suficiente, el detector será capaz de identificar a cada una de las personas, asociándolo a su *track* correspondiente.

b) **Dos o más personas que se encuentra juntas siempre**: Este escenario es análogo al anterior. La diferencia reside en que, al no separarse en ningún momento durante la grabación, no se puede especificar un *track* a cada uno de los individuos.

5.2.2 Presencia de otros objetos en escena

Del mismo modo, también resulta vital evaluar el comportamiento de este sistema ante apariciones de diversos objetos distintos de personas que puedan derivar en falsos positivos. Un sistema de videovigilancia ha de ser robusto ante detecciones que puedan suponer en falsas alarmas.

5.2.2.1 Tipos de objetos encontrados

En la Ilustración 26 se muestran situaciones en las que el sistema actúa correctamente, situaciones en las que falla y en las que se existen mecanismos de contingencia para minimizar la influencia de fallos causados por la presencia de objetos en movimiento distintos a personas.

Ilustración 26 Comportamiento ante objetos distintos a personas

Por lo tanto, los objetos distintos a personas que se han encontrado y la reacción del sistema ante su aparición, son los siguientes:

a) **Personas con accesorios o portando objetos**: De primeras, el detector de personas es capaz de reconocer a sujetos humanos que porten objetos tales a carritos, carretillas o maletas. Pues ha sido entrenado con un

conjunto de datos suficientemente diverso en este sentido. Del mismo modo a la hora de considerar la apariencia de estos sujetos al llevar a cabo la re-identificación, el modelo de aprendizaje automático utilizado se encuentra entrenado con imágenes de personas con diversa apariencia, vestimenta y con distintitos objetos que puedan portar.

- **Objetos distintos a personas en movimiento**: Ante diversidad de elementos en movimiento, el sistema es capaz de discernir cuáles de ellos se corresponden a sujetos humanos y cuáles son objetos distintos a personas que se encuentran en movimiento.

5.2.2.2 Fallos en la clasificación de personas

Ante la aparición de objetos en movimiento en escena que puedan resultar potencialmente personas, el clasificador ha de encargarse de discernir entre aquellas que lo son y el resto de los objetos que no resultan de interés. Sin embargo, hay situaciones en las que el clasificador comete fallos, resultando en falsos positivos y falsos negativos, como los mostrados en la Ilustración 27.

Ilustración 27 Fallos en la clasificación

Así, las situaciones que se pueden dar ante un fallo en la clasificación y las reacciones del sistema ante las mismas son las siguientes:

a) **Falsos positivos**: Debido a la interacción de los objetos en movimiento con el medio o con el resto de los objetos de la escena, hay situaciones en las que son clasificados como personas. Sin embargo, esto suele ser bastante atípico y la solución se plantea desde el módulo del *tracker* de instancias. Así, solo aquellos elementos móviles potencialmente humanos que se detecten de forma continuada durante un cierto número de fotogramas (en nuestro caso 15 fotogramas) tendrán la consideración de personas.

b) **Falsos negativos**: Debido a la existencia de accesorios u objetos que portan ciertas personas, en ocasiones es difícil clasificarlas. Esto es debido principalmente a la inexistencia de conjuntos de datos orientados a la detección de personas que se encuentren en tales situaciones, como, por ejemplo, para la detección de operarios de construcción. Sin embargo, si aun resultando difícil la clasificación como personas, esta se puede llevar a cabo en un cierto número de ocasiones durante un periodo de tiempo (en nuestro caso 30 fotogramas), pueden considerarse dichas detecciones como personas.

5.2.3 Fallos en transformaciones homográficas

Se ha establecido unos supuestos a la hora de llevar a cabo el diseño del calculador homográfico que en ocasiones ciertas escenas podrían no cumplir. Es por ello por lo que el funcionamiento del calculador para la obtención de las coordenadas de las personas detectadas podría verse afectado. En la Ilustración 28 se ilustra el problema existente, que se corresponde con la **inexactitud de las transformaciones de aquellas detecciones que se encuentran por encima del plano del suelo** (coordenada Z > 0). Esto se debe a que la homografía dentro de nuestro sistema está ideada para hallar las coordenadas de aquellas detecciones que se encuentran apoyadas en el plano del suelo. Así, por ejemplo, en la Ilustración 28 para la detección de personas realizada (D_1 en la ilustración), se calculan sus coordenadas y, por tanto, la proyección sobre el plano del suelo (D_1'). Sin embargo, para una detección que se encuentra por

encima del plano del suelo (D_2), se obtiene una proyección sobre el plano del suelo (D_2') que no se corresponde con la real ($D'_{2\ (z=0)}$).

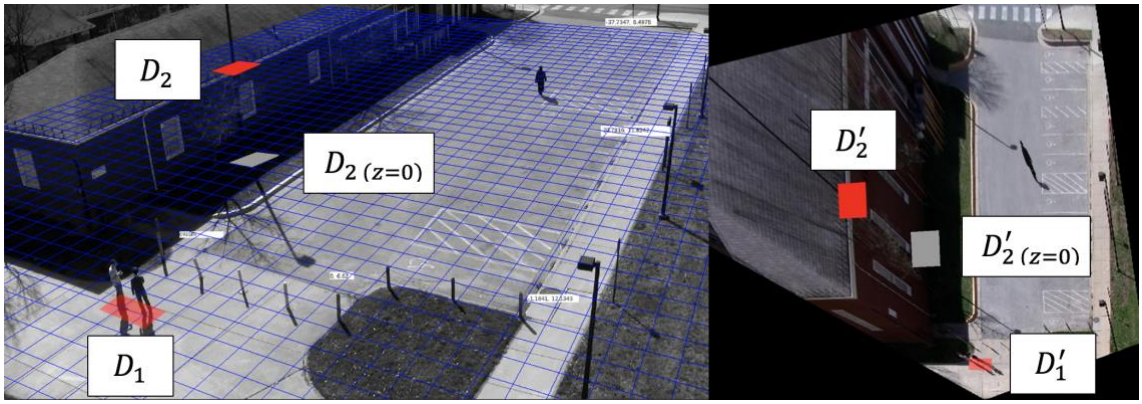

Ilustración 28 Transformación homográfica de detecciones con Z > 0

Por este problema el sistema actual de videovigilancia resulta válido exclusivamente en situaciones en las que la grabación se realiza sobre objetos que respetan el plano dominante de la homografía (en nuestro caso, el suelo). Así, el cálculo de las coordenadas de aquellas detecciones que se encuentran por encima del nivel del suelo no será en ningún caso correcto.

5.3 Apariencia final del sistema

Una vez implementados todos los módulos que se han descrito en el apartado de Metodología, se obtiene una visualización similar a la mostrada en la Ilustración 29.

Ilustración 29 Ejemplo de visualización del sistema

Esta visualización se compone de dos tomas principales:

a) **Vista de la cámara**: Esta toma muestra la imagen tal cual la capta la cámara. Sobre ella se imprimen las detecciones realizadas en cada momento (en rojo), mientras que en verde se muestran los *tracks* con su identificador en la parte superior y el camino que han seguido en los últimos fotogramas.

b) **Vista cenital**: Esta toma consta de una vista del plano cenital de la escena. Sobre ella se proyectan los *tracks* con su identificación en verde, así como el camino que estos han seguido durante los últimos fotogramas.

6 Conclusiones y futuras mejoras

En este trabajo se ha llevado a cabo el estudio, diseño e implementación de una solución integral a un problema de videovigilancia, principalmente aplicable sobre instalaciones críticas, debido a su nivel de automatización y a las funcionalidades que ofrece. A continuación, se definen los objetivos inicialmente establecidos y el estado en el que se encuentran una vez finalizado el proyecto.

Tal y como se estableció, en primer lugar, se ha llevado a cabo un *estudio del estado del arte de las técnicas existentes para la vigilancia de infraestructuras críticas, haciendo hincapié en aquellas que requieren del uso de vídeo*. Tras este estudio se han puesto de manifiesto los problemas socioeconómicos derivados de la falta de seguridad dentro de una instalación de estas características. Además, se han descubierto las deficiencias de la videovigilancia tradicional, que utiliza operadores humanos y la necesidad de la automatización de estas tareas.

En segundo lugar, se quiso *realizar un estudio del estado del arte de las técnicas existentes para detección y seguimiento de personas y valorar su aplicabilidad en un problema de videovigilancia de estas características*. De este estudio bibliográfico se obtuvieron las soluciones más prometedoras a estos dos problemas. Adicionalmente, se identificaron otros aspectos que también tienen cabida dentro de nuestro sistema y que inicialmente no fueron valorados, como la re-identificación de personas. Así, a partir de este estudio se ha confirmado la aplicabilidad de las distintas soluciones existentes a estos problemas en nuestro caso de uso.

Inicialmente, se planteó *el análisis del funcionamiento de cada una de estas técnicas de acuerdo con las métricas que arrojan y el coste computacional que suponen*. Por ello, se han llevado a cabo pruebas de concepto de las distintas soluciones valoradas. Así, se han realizado comparativas en cuanto a su funcionamiento y eficiencia, escogiendo aquellas técnicas más alineadas a los requisitos establecidos para nuestro sistema y para el cometido específico de la solución dentro de este.

El objetivo principal del proyecto ha sido *el diseño y desarrollo de una solución software íntegra de videovigilancia de reducido coste computacional, que*

funcione en tiempo real. Por ello, se ha diseñado una estructura modular para nuestro sistema, definiendo el cometido e interacción de cada uno de los componentes. Seguidamente, se ha llevado a cabo la implementación de cada uno de los módulos del sistema y se han integrado de acuerdo con el diseño realizado. La estructura resultante garantiza la independencia de cada uno de los módulos entre sí. A fin de mostrar la potencia de nuestra solución, esta ha sido probada sobre el conjunto de videovigilancia VIRAT (Oh et al., 2011), con grabaciones de distintas instalaciones e infraestructuras. Con ello, se han obtenido métricas de eficiencia y calidad utilizadas en el estado del arte para establecer un marco de referencia con el cual poder comparar esta, con las distintas soluciones que se propongan en un futuro. Los resultados dejan entrever que la solución es una buena aproximación y confirman su funcionamiento en tiempo real. Sin embargo, de cara a una comparación con otras soluciones, **no se han encontrado otras que abordando el mismo problema concretamente hagan uso del mismo conjunto de datos**. Por ello, **es difícil llevar a cabo una comparación** de estas métricas **con las de otras soluciones**.

El otro objetivo establecido en cuando a eficiencia es la *valoración de la futura integración del sistema implementado en un sistema embebido*. De acuerdo con los criterios de eficiencia previamente definidos, esta solución consigue obtener unos resultados que garantizan por un amplio margen su **funcionamiento en tiempo real**. En general, se han obtenido resultados para distintas configuraciones del sistema. Cada una de estas configuraciones garantizan un nivel de eficiencia diferente, pudiéndose hacer uso de una u otra de acuerdo con las necesidades de velocidad o de precisión existentes. Los resultados de aquellas configuraciones más rápidas parecen indicar la posibilidad de la **integración de este proyecto dentro de un sistema embebido** (Por ejemplo, NVIDIA® Jetson™).

El último objetivo que se estableció inicialmente fue, *de acuerdo con las métricas de calidad de cada uno de los módulos implementados del sistema, valorar la posible aplicación de este sobre una instalación crítica real*. En general, no se ha llegado a una conclusión en cuanto a la aplicabilidad del sistema implementado en una instalación crítica real. Debido a la **inexistencia de conjuntos de datos**

con las mismas características del domino del problema de nuestro proyecto, **no se puede valorar la implantación de este sistema en una situación real**. Sin embargo, el autor de este proyecto es muy positivo de cara a valorar su uso en instalaciones y localizaciones que ya han sido consideradas en los conjuntos de datos utilizados. Así, se han realizado pruebas sobre vídeos de instalaciones y localizaciones como: aparcamientos, construcciones, calles poco saturadas de viandantes, etc.

En cuanto a futuras mejoras, debe considerarse un continuo estudio de las soluciones que puedan surgir, que mejoren el estado del arte en los aspectos de rendimiento y eficiencia considerados en este trabajo. Así, al tratarse de un sistema modular, puede llevarse a cabo una actualización de cualquiera de sus módulos sin afectar al resto del sistema. También se debería garantizar una mayor capacidad de detección y seguimiento en escenas más complejas que las consideradas en este proyecto. Por ello, se propone la compatibilidad del sistema con un entorno de múltiples cámaras, tanto independientes como solapadas. Esto permitiría llevar a cabo una mejor gestión de las oclusiones, detecciones y re-identificaciones, al disponer de más información de la escena y de los sujetos humanos que en ella aparecen.

7 Referencias bibliográficas

Ahmed, E., Jones, M., & Marks, T. K. (2015). An improved deep learning architecture for person re-identification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3908–3916). IEEE. https://doi.org/10.1109/CVPR.2015.7299016

Aldasouqi, I., & Hassan, M. (2010). Human face detection system using HSV. In *International conference on Circuits, Systems, Electronics, Control and Signal Processing - Proceedings*.

Babenko, B., Ming-Hsuan Yang, & Belongie, S. (2009). Visual tracking with online Multiple Instance Learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 983–990). IEEE. https://doi.org/10.1109/CVPRW.2009.5206737

Baumann, U., Huang, Y. Y., Glaser, C., Herman, M., Banzhaf, H., & Zollner, J. M. (2018). Classifying Road Intersections using Transfer-Learning on a Deep Neural Network. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, *2018-Novem*, 683–690. https://doi.org/10.1109/ITSC.2018.8569916

Benenson, R., Mathias, M., Timofte, R., & Van Gool, L. (2012). Pedestrian detection at 100 frames per second. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 2903–2910). https://doi.org/10.1109/CVPR.2012.6248017

Berclaz, J., Fleuret, F., Türetken, E., & Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2011.21

Bernardin, K., & Stiefelhagen, R. (2008). Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, *2008*, 1–10. https://doi.org/10.1155/2008/246309

Bertozzi, M., Broggi, A., Rose, M. Del, Felisa, M., Rakotomamonjy, A., & Suard, F. (2007). A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. In *2007 IEEE Intelligent Transportation Systems Conference* (pp. 143–148). IEEE. https://doi.org/10.1109/ITSC.2007.4357692

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and

realtime tracking. In *Proceedings - International Conference on Image Processing, ICIP*. https://doi.org/10.1109/ICIP.2016.7533003

Bilal, M., & Hanif, M. S. (2019). High Performance Real-Time Pedestrian Detection Using Light Weight Features and Fast Cascaded Kernel SVM Classification. *Journal of Signal Processing Systems*, *91*(2), 117–129. https://doi.org/10.1007/s11265-018-1374-7

Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2010.5539960

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). Training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*.

Bradski, G. (2000). The OpenCV Library. *Dr Dobbs Journal of Software Tools*. https://doi.org/10.1111/0023-8333.50.s1.10

Brown, G., Carlyle, M., Salmerón, J., & Wood, K. (2006). Defending critical infrastructure. *Interfaces*, *36*(6), 530–544. https://doi.org/10.1287/inte.1060.0252

Brunetti, A., Buongiorno, D., Trotta, G. F., & Bevilacqua, V. (2018). Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*. https://doi.org/10.1016/j.neucom.2018.01.092

Brutzer, S., Hoferlin, B., & Heidemann, G. (2011). Evaluation of background subtraction techniques for video surveillance. In *CVPR 2011* (pp. 1937–1944). IEEE. https://doi.org/10.1109/CVPR.2011.5995508

Cheng, D., Gong, Y., Zhou, S., Wang, J., & Zheng, N. (2016). Person re-identification by multi-channel parts-based CNN with improved triplet loss function. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2016.149

Chiu, S.-Y., Chiu, C.-C., & Xu, S. (2018). A Background Subtraction Algorithm in Complex Environments Based on Category Entropy Analysis. *Applied Sciences*, *8*(6), 885. https://doi.org/10.3390/app8060885

Chollet, F. (2015). Keras: The Python Deep Learning library. *Keras.Io*.

Comaniciu, D., & Ramesh, V. (2000). Mean shift and optimal prediction for

efficient object tracking. In *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)* (pp. 70–73). IEEE. https://doi.org/10.1109/ICIP.2000.899297

Dadi, H. S., Pillutla, G. K. M., & Makkena, M. L. (2018). Face Recognition and Human Tracking Using GMM, HOG and SVM in Surveillance Videos. *Annals of Data Science*, *5*(2), 157–179. https://doi.org/10.1007/s40745-017-0123-2

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005* (Vol. I, pp. 886–893). IEEE. https://doi.org/10.1109/CVPR.2005.177

Davis, J. V., Kulis, B., Jain, P., Sra, S., & Dhillon, I. S. (2007). Information-theoretic metric learning. *ACM International Conference Proceeding Series*, *227*, 209–216. https://doi.org/10.1145/1273496.1273523

Fang, W., Chen, J., Lu, T., & Hu, R. (2018). Feature Synthesization for Real-Time Pedestrian Detection in Urban Environment. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 102–112). https://doi.org/10.1007/978-3-030-00767-6_10

Farenzena, M., Bazzani, L., Perina, A., Murino, V., & Cristani, M. (2010). Person re-identification by symmetry-driven accumulation of local features. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2360–2367. https://doi.org/10.1109/CVPR.2010.5539926

Godbehere, A. B., Matsukawa, A., & Goldberg, K. (2012). Visual tracking of human visitors under variable-lighting conditions for a responsive audio art installation. In *Proceedings of the American Control Conference*.

Grabner, H., Grabner, M., & Bischof, H. (2006). Real-time tracking via on-line boosting. In *BMVC 2006 - Proceedings of the British Machine Vision Conference 2006*.

Gray, D., & Tao, H. (2008). Viewpoint invariant pedestrian recognition with an ensemble of localized features. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-540-88682-2-21

Guillaumin, M., Verbeek, J., & Schmid, C. (2009). Is that you? Metric learning approaches for face identification. *Proceedings of the IEEE International Conference on Computer Vision*, (Iccv), 498–505. https://doi.org/10.1109/ICCV.2009.5459197

Guo, L., Xu, D., & Qiang, Z. (2016). Background Subtraction Using Local SVD Binary Pattern. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 1159–1167). IEEE. https://doi.org/10.1109/CVPRW.2016.148

Gupta, R., Agarwal, R., & Goyal, S. (2014). A Review of Cyber Security Techniques for Critical Infrastructure Protection. *International Journal of Computer Science & Engineering Technology (IJCTSET)*.

Haering, N., Venetianer, P. L., & Lipton, A. (2008). The evolution of video surveillance: An overview. *Machine Vision and Applications*. https://doi.org/10.1007/s00138-008-0152-0

Hao, T., Wang, Q., Wu, D., & Sun, J. S. (2018). Multiple person tracking based on slow feature analysis. *Multimedia Tools and Applications*, *77*(3), 3623–3637. https://doi.org/10.1007/s11042-017-5218-4

Hartley, R., & Zisserman, A. (2000). Multiple View Geometry in Computer Vision. *Cambridge University Press*.

Harvey Adam. LaPlace, J. (2019). MegaPixels: Origins, Ethics, and Privacy Implications of Publicly Available Face Recognition Image Datasets. Retrieved from https://megapixels.cc/

Hembury, G. A., Borovkov, V. V., Lintuluoto, J. M., & Inoue, Y. (2003). Deep Residual Learning for Image Recognition Kaiming. *Chemistry Letters*, *32*(5), 428–429. https://doi.org/10.1246/cl.2003.428

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2012). Exploiting the Circulant Structure of Tracking-by-Detection with Kernels. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 702–715). https://doi.org/10.1007/978-3-642-33765-9_50

Hollemans, M. (2018). MobileNet version 2.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. Retrieved from http://arxiv.org/abs/1704.04861

Hue, C., Le Cadre, J. P., & Pérez, P. (2002). Tracking multiple objects with particle filtering. *IEEE Transactions on Aerospace and Electronic Systems*. https://doi.org/10.1109/TAES.2002.1039400

Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, & Li Fei-Fei. (2009). ImageNet: A large-scale hierarchical image database. https://doi.org/10.1109/cvprw.2009.5206848

Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/

Ju, J., Ku, B., Kim, D., Song, T., Han, D. K., & Ko, H. (2015). Online multi-person tracking for intelligent video surveillance systems. *2015 IEEE International Conference on Consumer Electronics, ICCE 2015*, 345–346. https://doi.org/10.1109/ICCE.2015.7066438

KaewTraKulPong, P., & Bowden, R. (2002). An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection. In *Video-Based Surveillance Systems* (pp. 135–144). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4615-0913-4_11

Kalal, Z., Mikolajczyk, K., & Matas, J. (2010). Forward-backward error: Automatic detection of tracking failures. In *Proceedings - International Conference on Pattern Recognition*. https://doi.org/10.1109/ICPR.2010.675

Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(7), 1409–1422. https://doi.org/10.1109/TPAMI.2011.239

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*. https://doi.org/10.1115/1.3662552

Kim, K., Chalidabhongse, T. H., Harwood, D., & Davis, L. (2005). Real-time foreground–background segmentation using codebook model. *Real-Time Imaging*, *11*(3), 172–185. https://doi.org/10.1016/j.rti.2004.12.004

Kostinger, M., Hirzer, M., Wohlhart, P., Roth, P. M., & Bischof, H. (2012). Large scale metric learning from equivalence constraints. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Ldml), 2288–2295. https://doi.org/10.1109/CVPR.2012.6247939

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification

with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97. https://doi.org/10.1002/nav.3800020109

Labbe, R. R. (2018). Kalman and Bayesian Filters in Python. *GitHub Repository*.

Li, W., Zhu, X., & Gong, S. (2018). Harmonious Attention Network for Person Re-identification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2285–2294). IEEE. https://doi.org/10.1109/CVPR.2018.00243

Li, Z., Chang, S., Liang, F., Huang, T. S., Cao, L., & Smith, J. R. (2013). Learning Locally-Adaptive Decision Functions for Person Verification. In *2013 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3610–3617). IEEE. https://doi.org/10.1109/CVPR.2013.463

Liao, S., Hu, Y., Zhu, X., & Li, S. Z. (2015). Person re-identification by Local Maximal Occurrence representation and metric learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07-12-June*, 2197–2206. https://doi.org/10.1109/CVPR.2015.7298832

Liao, S., & Li, S. Z. (2015). Efficient PSD constrained asymmetric metric learning for person re-identification. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 Inter*, 3685–3693. https://doi.org/10.1109/ICCV.2015.420

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*. https://doi.org/10.1023/B:VISI.0000029664.99615.94

Lukežič, A., Vojíř, T., Čehovin Zajc, L., Matas, J., & Kristan, M. (2018). Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *International Journal of Computer Vision*, *126*(7), 671–688. https://doi.org/10.1007/s11263-017-1061-3

Mandellos, N. A., Keramitsoglou, I., & Kiranoudis, C. T. (2011). A background subtraction algorithm for detecting and tracking vehicles. *Expert Systems with Applications*, *38*(3), 1619–1631.

https://doi.org/10.1016/j.eswa.2010.07.083

Mateus, A., Ribeiro, D., Miraldo, P., & Nascimento, J. C. (2019). Efficient and robust Pedestrian Detection using Deep Learning for Human-Aware Navigation. *Robotics and Autonomous Systems*, *113*, 23–37. https://doi.org/10.1016/j.robot.2018.12.007

Matsukawa, T., Okabe, T., Suzuki, E., & Sato, Y. (2016). Hierarchical Gaussian Descriptor for Person Re-identification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 1363–1372. https://doi.org/10.1109/CVPR.2016.152

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*.

Mignon, A., & Jurie, F. (2012). PCCA: A new approach for distance learning from sparse pairwise constraints. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2666–2672. https://doi.org/10.1109/CVPR.2012.6247987

Ni, T., Ding, Z., Chen, F., & Wang, H. (2018). Relative Distance Metric Leaning Based on Clustering Centralization and Projection Vectors Learning for Person Re-Identification. *IEEE Access*, *6*(Cccv), 11405–11411. https://doi.org/10.1109/ACCESS.2018.2795020

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.-C., Lee, J. T., … Desai, M. (2011). A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011* (pp. 3153–3160). IEEE. https://doi.org/10.1109/CVPR.2011.5995586

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*. https://doi.org/10.1016/0031-3203(95)00067-4

Oliphant, T. (2006). NumPy: A guide to NumPy. Retrieved from http://www.numpy.org/

Pedagadi, S., Orwell, J., Velastin, S., & Boghossian, B. (2013). Local fisher discriminant analysis for pedestrian re-identification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3318–3325. https://doi.org/10.1109/CVPR.2013.426

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal*

of Machine Learning Research.

Piccardi, M. (2004). Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)* (Vol. 4, pp. 3099–3104). IEEE. https://doi.org/10.1109/ICSMC.2004.1400815

Python. (2017). What is Python? Executive Summary | Python.org. *Python Software Foundation*.

Quinton, B. J., & Se, J. (2018). Bio-inspired heterogeneous architecture for real-time pedestrian detection applications, 535–548. https://doi.org/10.1007/s11554-016-0581-3

Rai, M., Asim Husain, A., Maity, T., & Kumar Yadav, R. (2019). Advance Intelligent Video Surveillance System (AIVSS): A Future Aspect. In *Intelligent Video Surveillance*. IntechOpen. https://doi.org/10.5772/intechopen.76444

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2016.2577031

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

Sforza, F., & Lippi, V. (2013). Support vector machine classification on a biased training set: Multi-jet background rejection at hadron colliders. *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. https://doi.org/10.1016/j.nima.2013.04.046

Smeulders, A. W. M., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., & Shah, M. (2014). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2013.230

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2*, 246–252.

Sun, L., Jiang, Z., Song, H., Lu, Q., & Men, A. (2018). Semi-Coupled Dictionary Learning with Relaxation Label Space Transformation for Video-Based Person Re-Identification. *IEEE Access*, *6*, 12587–12597. https://doi.org/10.1109/ACCESS.2018.2803789

Tesfaye, Y. T., Zemene, E., Prati, A., Pelillo, M., & Shah, M. (2019). Multi-target Tracking in Multiple Non-overlapping Cameras Using Fast-Constrained Dominant Sets. *International Journal of Computer Vision*, 1–15. https://doi.org/10.1007/s11263-019-01180-6

Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., & Tubaro, S. (2016). Deep Convolutional Neural Networks for pedestrian detection. *Signal Processing: Image Communication*, *47*, 482–489. https://doi.org/10.1016/j.image.2016.05.007

Troscianko, T., Holmes, A., Stillman, J., Mirmehdi, M., Wright, D., & Wilson, A. (2004). What happens next? The predictability of natural behaviour viewed through CCTV cameras. *Perception*. https://doi.org/10.1068/p3402

Turchini, F., Seidenari, L., Uricchio, T., & Del Bimbo, A. (2018). Deep Learning Based Surveillance System for Open Critical Areas. *Inventions*, *3*(4), 69. https://doi.org/10.3390/inventions3040069

Varior, R. R., Haloi, M., & Wang, G. (2016). Gated siamese convolutional neural network architecture for human re-identification. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9912 LNCS*, 791–808. https://doi.org/10.1007/978-3-319-46484-8_48

Viola, P., Jones, M. J., & Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*. https://doi.org/10.1007/s11263-005-6644-8

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. D. (2016). *A survey of transfer learning*. *Journal of Big Data* (Vol. 3). Springer International Publishing. https://doi.org/10.1186/s40537-016-0043-6

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on*

Image Processing (ICIP) (Vol. 2017-Septe, pp. 3645–3649). IEEE. https://doi.org/10.1109/ICIP.2017.8296962

Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., & Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI International Joint Conference on Artificial Intelligence*.

Yusta, J. M., Correa, G. J., & Lacal-Arántegui, R. (2011). Methodologies and applications for critical infrastructure protection: State-of-the-art. *Energy Policy*, *39*(10), 6100–6119. https://doi.org/10.1016/j.enpol.2011.07.010

Zhanfeng, Y., Zhou, S.-K., & Chellappa, R. (2004). Robust two-camera tracking using homography. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing* (Vol. 3, pp. iii-1–4). IEEE. https://doi.org/10.1109/ICASSP.2004.1326466

Zhang, C., Huang, Y., Wang, Z., Jiang, H., & Yan, D. (2018). Cross-camera multi-person tracking by leveraging fast graph mining algorithm. *Journal of Visual Communication and Image Representation*, *55*, 711–719. https://doi.org/10.1016/j.jvcir.2018.08.006

Zhang, J., Yuan, Y., & Wang, Q. (2019). Night Person Re-Identification and a Benchmark. *IEEE Access*, *7*, 95496–95504. https://doi.org/10.1109/ACCESS.2019.2929854

Zhang, S., Benenson, R., Omran, M., Hosang, J., & Schiele, B. (2018). Towards Reaching Human Performance in Pedestrian Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 973–986. https://doi.org/10.1109/TPAMI.2017.2700460

Zhao, Haiyu, Tian, M., Sun, S., Shao, J., Yan, J., Yi, S., … Tang, X. (2017). Spindle Net: Person Re-identification with Human Body Region Guided Feature Decomposition and Fusion. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 2017-Janua, pp. 907–915). IEEE. https://doi.org/10.1109/CVPR.2017.103

Zhao, Hui-huang, & Liu, H. (2019). Multiple classifiers fusion and CNN feature extraction for handwritten digits recognition. *Granular Computing*, *0*(0), 0. https://doi.org/10.1007/s41066-019-00158-6

Zivkovic, Z., & Van Der Heijden, F. (2006). Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition*

Letters, *27*(7), 773–780. https://doi.org/10.1016/j.patrec.2005.11.005