UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Detección de personas en presencia de grupos

Máster en Investigación e Innovación en Tecnologías de la Información y la Comunicación (I^2-TIC)

Autor: Merino Martínez, Sergio

Tutor: García Martín, Álvaro

Ponente: Martínez Sánchez, José María

Septiembre, 2015

Agradecimientos

Al término de esta etapa de mi vida, quiero expresar un profundo agradecimiento a quienes con su ayuda, paciencia y apoyo me alentaron a lograr uno de mis grandes anhelos, formarme como ingeniero en informática.

A mi tutor Álvaro, sabiendo que jamás encontraré la forma de agradecer su constante apoyo y paciencia, y solo quiero que entiendan que mis esfuerzos y logros han sido también los suyos.

A todos los profesores que he tenido a lo largo de la carrera y el máster, gracias a su cariño, guía y apoyo, siempre tendrán mi eterna gratitud por toda la responsabilidad e invaluable ayuda que siempre me han proporcionado.

A Miguel Ángel Hernando, mi jefe durante el breve periodo que pasé en BBVA, por confiar en mí desde el primer momento y darme la oportunidad de demostrar lo que puedo llegar a hacer.

A todos y cada uno de mis compañeros de la Universidad de Valencia y de la Universidad Autónoma de Madrid por acompañarme cada día en clase y por hacer que todos estos años se me hayan hecho demasiado cortos.

A mis amigos de siempre, que en el estrés de los exámenes, han conseguido distraerme y hacerme pasar muy buenos momentos saliendo a cenar, viendo series, películas y sobretodo haciéndome sonreír.

A mi familia, en especial a mi abuela y a mis padres, que desde pequeño han sabido guiarme por el buen camino a pesar de las adversidades y que me han dado todo su amor, compresión y apoyo sin pedir nada a cambio.

Y por último, pero no por eso menos importante, a mi novia Eva que me ha acompañado durante el camino, que me ha soportado en los buenos y en los malos momentos y que con su ayuda y cariño he podido poner fin a una de las etapas más importantes y felices de mi vida. Y por eso ella sabe que tiene y tendrá siempre mi eterno amor y agradecimiento.

Resumen

La visión por computador es una rama de la informática, donde se trata de dotar a las máquinas de ese estimable sentido que es la vista. Esto unido a la inteligencia artificial, donde se quiere conseguir que un computador tenga consciencia, puede lograr que de manera automática y autónoma un ordenador pueda, por ejemplo, vigilar la seguridad de las personas.

Por otro lado, la vídeo-vigilancia puede procesar diferentes tipos de entradas para generar alertas que un usuario pueda utilizar para evitar un insatisfactorio suceso. No hace falta decir que la vídeo-vigilancia no es nada sin la visión artificial, la cual es la encargada de hacer todo el procesamiento de las entradas que proporcionan las cámaras, y dar esas salidas que son las detecciones deseadas.

Este proyecto consta de, por lo tanto, una unión de ambas. Debe de conseguir, utilizando diferentes herramientas, la transparente unificación de todos estos conceptos. Para ello, se quiere como objetivo final conseguir detectar personas de una manera eficaz cuando estas están en grupos de mayor o menor dimensión.

Concretando, buscamos una mejora en la detección de personas en presencia de grupos. En este tipo de escenarios es aún más compleja la detección, dado que las personas pueden están altamente ocluidas unas con otras.

Este proyecto lo podríamos catalogar como I+D+i. Dado que se investiga todo lo relacionado con la visión artificial y la vídeo-vigilancia. Se desarrolla un software a modo de prototipo que dispone de todas las funcionalidades deseadas. Y también se innova en la implementación propia de diferentes algoritmos.

Por último, es importante recalcar que la principal vocación del presente trabajo es sentar las bases para el desarrollo posterior de diferentes aplicaciones relacionadas con la monitorización automática de escenas tanto desde el punto de vista de la implementación eficiente como desde el punto de vista del diseño de nuevos algoritmos de detección.

Palabras Clave: Detección de personas, clasificación de personas, personas en grupos, DTDP, modelo basado en partes.

Abstract

Computer vision is a field in computer science, which tries to provide machines with that appreciable sense that is sight. This coupled with artificial intelligence, where you want a computer to have consciousness, can make automatically and autonomously that computers, for example, monitor people safety.

Furthermore, video surveillance can process different types of inputs to generate alerts that a user can use to avoid an unsatisfactory event. Needless to say that video surveillance is nothing without the artificial vision, which is responsible for all of the processing of the inputs, which the cameras provide, and give those outputs, which are the desire detections.

This project involves, therefore, a junction of the two issues. It must achieve, using different tools, the transparent unification of all these concepts. Therefore, our final aim is to detect people effectively in crowd scenes.

Specifying, we search an improvement in the people detection in the presence of the groups. In this type of scenario is even more complex detection, since people can are highly occluded with other persons.

This project could be included as a I+D+i type. Since everything related to computer vision and video surveillance is investigated. Specific software is developed as a prototype that has all the desired features. And also innovates in the actual implementation of different algorithms.

Finally, make clear that what we want is to be helpful to different projects that linked to this one, can increase knowledge, effectiveness and efficiency of different investigations and future implementations.

Finally, it is important to stress that the main vocation of the present work is to as much lay the foundations for the later development of different applications related to the automatic monitoring from scenes from the point of view of the efficient implementation as from the point of view of the design of new algorithms of detection.

Keywords: person detection, person classification, crowd, DTDP, part-based model.

Índice

Índice de ilustraciones

Índice de Tablas

1. Introducción

En la actualidad existen diferentes entornos que procesan de manera pseudointeligente la información que captan de diferentes fuentes. El gran avance por parte del hardware, haciendo máquinas cada vez más potentes, permite un rápido desarrollo, a su vez, del software. En el caso de la inteligencia artificial aplicada a la vídeo-vigilancia, este avance tiene un mayor interés que en otras disciplinas, debido a su importancia en la velocidad necesaria para hacer grandes análisis. Procesar información visual de manera óptima, es difícil incluso hoy y en día. Este procesamiento está dividido en diferentes módulos que, aunque se pueden optimizar de manera separada, dependen unos de otros.

Particularizando en la detección de personas, hay un sinfín de publicaciones cada año, diferentes tipos de algoritmos con implementaciones muy diversas y en diferentes lenguajes de programación, etc. Nosotros en este proyecto nos centraremos en la mejora de detección de personas en grupos, siendo este un campo a mejorar sustancialmente en la visión artificial y más en concreto en la video-vigilancia. Por lo tanto, para llevar correctamente esta investigación realizaremos un desglose de objetivos a conseguir para fijarnos unas metas claras, una revisión bibliográfica profunda, el posterior desarrollo y por último, las pruebas necesarias para su evaluación. Teniendo en cuenta todo lo anterior, vamos a introducir nuestro proyecto, desglosando las diferentes etapas que lo componen.

En el capítulo 2 se exponen los diferentes motivos y objetivos que han movido el desarrollo de este proyecto, y el cómo la necesidad de sentirnos seguros promueve el avance de la tecnología en este campo.

En el capítulo 3 evaluamos el estado del arte, desde el punto de vista procedimental, en el que explicamos diferentes métodos existentes en la actualidad para la detección de personas. Lo que se pretende con esta sección es poner al corriente al lector sobre todas las herramientas actuales e introducir a los conceptos que atañen a la elaboración de este proyecto.

En el capítulo 4 hemos realizado la descripción del desarrollo del proyecto. En primer lugar hablamos sobre el algoritmo origen y diferentes modificaciones que ha tenido durante estos años. Y por último hablamos concretamente de nuestro algoritmo, haciendo una comparación exhaustiva sobre las modificaciones realizadas a la implementación base. Estas modificaciones mejoran las detecciones en grupos y se añade un autoajuste de parámetros para generalizar el algoritmo y mejorar en lo posible la precisión.

En el capítulo 5 se describirán los resultados del proyecto y se someterá al programa a pruebas para comprobar que cumple con lo establecido, definiendo cada experimento. En las evaluaciones se comparará tanto con diferentes algoritmos ya implementados, como con diferentes configuraciones de nuestro algoritmo para poder escoger la más eficiente y eficaz (algoritmo con autoajuste de parámetro o sin él, por ejemplo).

En el capítulo 6 se analizara si se han cumplido los objetivos, a partir de las conclusiones y las posibles mejoras futuras sobre el proyecto.

2. Motivación y objetivos

2.1. Motivación

Tenemos la necesidad de sentirnos seguros. Ya lo decía Abraham Maslow en su famosa pirámide, la cual se basa en una jerarquía de las necesidades humanas, explicando que conforme se van satisfaciendo las de la base, el ser humano desarrolla otras más elevadas. La necesidad de sentirnos seguros se encuentra en el segundo lugar más relevante de dicha pirámide, después de las necesidades fisiológicas. Dada la necesidad de seguridad, la cual utilizan muchos gobiernos para invadir nuestra privacidad, para captar y analizar información personal sin nuestro consentimiento, y muchas otras cosas que ni siquiera sabemos. Podemos decir que necesitamos de alguien o algo omnipotente y neutral que juegue un papel crucial en nuestra seguridad y, por lo tanto, también en nuestra vigilancia. Por lo tanto, podríamos decir, incluso, que es una sabia elección dejar que realice estas acciones un computador, una máquina capaz de procesar en milisegundos mucha más información que muchos seres humanos en varios minutos. Y no solo eso, si no que le podemos indicar de una manera más o menos sencilla qué queremos buscar, dónde buscar y cuándo hacerlo.

Si analizamos varias películas o series que han tratado el tema de la vigilancia constante por parte de las máquinas, siempre tendremos el mismo resultado de parte de la gente: el miedo a saber hasta dónde somos capaces de llegar y cuánto somos capaces de dejar que hagan las máquinas por nosotros. En mi opinión, se debería dejar bien claro desde un principio, y programar así a las máquinas que nos vigilen, que un análisis de lo que sucede actualmente, sin tener que meterse en la casa de nadie (si no es con su permiso), puede dar grandes ventajas para actuar en consecuencia a lo procesado. Que analizar e intentar predecir sucesos futuros es algo que puede evitar grandes desgracias, pero que nunca debemos dejar que las maquinas lleguen a la conclusión que una persona es mala, por muy cierto que sea.

“Tenemos que hacer planes para la libertad, y no sólo para la seguridad, por la única razón de que sólo la libertad puede hacer segura la seguridad”, es una frase de Karl Popper que da que pensar, sobre todo si la unimos a esta otra de Benjamin Franklin “cualquier sociedad que renuncie a un poco de libertad para ganar un poco de seguridad, no merece ninguna de las dos cosas”. Analizando un poco lo dicho por estas sabias personas, queda patente lo descrito en los párrafos anteriores. Que necesitamos sentirnos seguros, pero para ello necesitamos sistemas no intrusivos que puedan vigilar nuestras acciones y actuar o avisar en consecuencia, para que quede intacta nuestra libertad como personas.

A estas alturas ya podemos deducir la motivación principal que llevó a la elección de este proyecto. Simplemente quería aportar mi granito de arena a lo que, en un futuro, puede llegar a ser una auténtica revolución tecnológica. Para ello es importante preparar un entorno para la detección de personas por parte de una máquina sin, en principio, la interacción humana. Otra de las razones importantes del desarrollo de este proyecto es mi interés por el análisis inteligente de información de cualquier tipo, y en concreto, de imágenes y video. Este análisis inteligente aplicado a la robótica de servicios, vehículos autónomos, etc. nichos de mercado que se están expandiendo y en el cual empieza a haber una gran necesidad de evolución, puede llegar a representar una mejora sustancial en nuestras vidas. Y yo, en mi futuro profesional, quiero formar parte de este desarrollo.

Centrándonos en la detección de personas, no es necesario decir, que es donde más potencial tiene la visión artificial y con ello un campo en expansión durante varios años. Ya he comentado mi pasión por este tipo de desarrollos y se acrecienta cuando un posible desarrollo puede ayudar a otras personas.

Viendo los trabajos futuros de incontables publicaciones en el ámbito de video-vigilancia, porque no escoger una de los retos más complejos y a la vez de los más útil, como puede ser la detección de personas en presencia de grupos (escenarios complejos). Este tipo de desafío, convierte este trabajo fin de máster en algo más, en algo a lo que dedicar mi tiempo tanto por obligación como por devoción.

Yendo al grano y sin alargarlo demasiado, la motivación del trabajo radica en mejorar la detección de personas en grupos, tras el trabajo ya realizado en el laboratorio de investigación de la UAM, el VPULab, para la detección de parejas y grupos de personas, tratando, por tanto, de mejorar dichos resultados. Para ello, se ha decidido tratar de aprender, en cierta medida, la escena y adaptar el algoritmo a dicha escena de forma lo más dinámica posible. De esta forma, se espera optimizar los resultados de dicho algoritmo para cada escena que nos encontremos, generando un algoritmo lo más general posible. En particular, se plantea un marco de "self-tuning" o autoajuste en el que la selección de varios parámetros del algoritmo se realiza de manera automática, evitando así la selección de forma manual de estos.

No quiero acabar este apartado de motivación sin dejar claro que lo que se pretende, principalmente, es motivar, valga la redundancia, a otras personas y por supuesto, a mí mismo a continuar con el tema de vídeo-vigilancia. La visión artificial y, en concreto, la vídeovigilancia unido a otras especialidades de mayor importancia como pueden ser: la robótica, la medicina, la biología o la psicología pueden conseguir una sociedad mucho más agradable y sencilla, para que vivamos los seres humanos. Porque el motivo principal de la invención del ordenador es hacer cosas tediosas y repetitivas para las personas, pero que las máquinas pueden hacer sin aburrirse y de una manera muy rápida, y si llevamos eso al extremo, no solo podemos dejar que hagan cosas por nosotros, sino que pueden llegar a hacerlas también mejor. Por eso es importante desarrollar esta parte que está en auge y que nos va a proporcionar ciertas ventajas muy favorables con respecto a épocas pasadas.

2.2. Objetivos

El principal objetivo de este TFM es el desarrollo de un algoritmo de detección jerárquica de personas en entornos con alta densidad de éstas, de tal forma que no se centre únicamente en la detección de personas individuales, sino que aproveche la información de detección de múltiples personas para mejorar los resultados obtenidos en este tipo de escenarios. El algoritmo utilizará la información de la fisionomía de una persona, con el objetivo de detectar personas como combinaciones de múltiples personas, que a su vez pueden ser definidas como un todo o escogiendo únicamente algunas de sus partes como cabeza, hombro, tronco, etc. Adicionalmente, se realizara un trabajo exhaustivo para que diferentes parámetros del modelo se autoajusten, mejorando, por tanto, la precisión (menos falsos positivos) y el tiempo computacional, u por tanto haciendo el algoritmo lo más general posible a cualquier tipo de escenario que se enfrente.

Los objetivos generales son, en definitiva, los siguientes:

- Realizar una investigación de la literatura actual y pasada en el tema del presente proyecto.
- Desarrollo de un algoritmo eficaz para detección de personas en entornos con grupos de estas.
- Evaluación del algoritmo implementado y comparación objetiva con diferentes métodos ya realizados.

El propósito de este proyecto, consiste en realizar un programa capaz de detectar personas en escenarios relativamente complejos (donde hay múltiples personas) procesando la información extraída de los fotogramas de un video y con ciertos parámetros autoajustables. Para ello los objetivos parciales que compondrían los objetivos más generales sería:

- Revisión bibliográfica profunda en visión artificial
- Revisión bibliográfica profunda en videovigilancia y algoritmia para este cometido.
- Estudio y aprendizaje de la implementación realizada por la VPULab de la UAM.
- Realización de un modelo autoajustable a la distribución y a las características de las personas con información de imágenes anteriores.
- Evaluación de resultados obtenidos de manera eficaz para comparar con algoritmos de similares características.
- Realización de pruebas con diferentes tipos de vídeos y de parámetros de configuración del algoritmo.

3. Estado del arte

Para el buen entendimiento de este proyecto es necesario analizar el estado del arte actual. Para ello analizaremos diferentes artículos publicados en revistas de renombre, centrándonos en los que en sí mismo son una recopilación de diferentes artículos que versan sobre el mismo tema (*surveys*).

Vamos por lo tanto a analizar 4 "*papers*" de diferentes años pero muy recientes (de 2012 al 2015). Estos artículos son "*Pedestrian Detection: An Evaluation of the State of the Art*" [1], "*Object Class Detection: A Survey*" [2], "*A Survey of Pedestrian Detection in Video*" [3] y "*Ten Years of Pedestrian Detection, What Have We Learned?*" [4].

Como veremos no se verán algoritmos que autoajusten dentro del Estado del Arte, aunque el motivo principal de este proyecto es su desarrollo. Esto es debido a que la detección de personas, por lo general, se suele centrar en actualizar el modelo de forma online, pero en este trabajo el "self-tunig" se ha planteado para adaptar el detector de grupos y no tanto la detección de personas. Es decir, el modelo de persona no cambia en ningún momento, lo que cambia es el tipo de grupo o parejas que se buscan modificando ciertos parámetros del modelo (no el modelo).

3.1. Introducción

La detección de personas no es más que un ejemplo concreto de la detección de objetos, usando un modelo concreto para dicho objeto (la persona) [2].

Como hemos hablado anteriormente la investigación sobre visión artificial centrada en detección y clasificación de peatones ha aumentado durante estos años, esto lo podemos ver en la Ilustración 3.1 donde vemos como ha crecido de manera notable el número de publicaciones en IEEE sobre este tema.

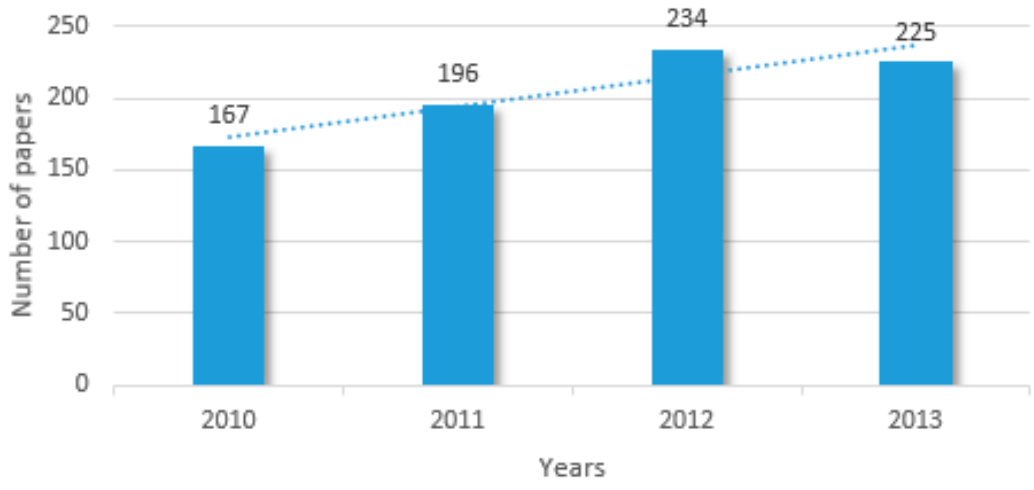

Ilustración 3.1: Número de publicaciones relacionadas con detección de peatones en IEEE [3]

Como podemos ver en la Ilustración 3.2, el avance en estos últimos 10 años ha sido bueno, reduciendo la cantidad de falsos positivos y de falsos negativos cometidos.

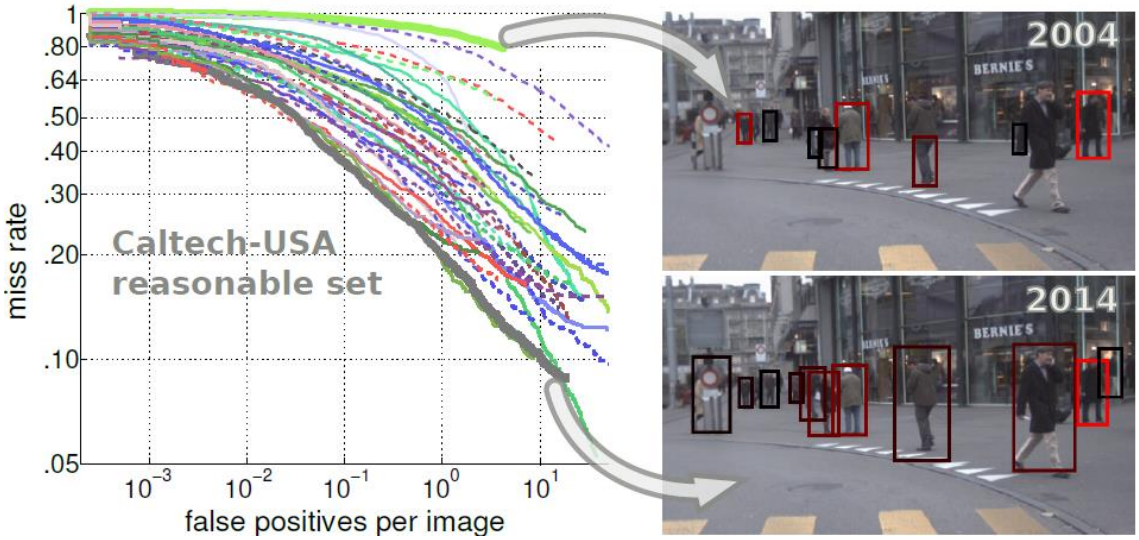

Ilustración 3.2: Mejora de número de falsos positivos de 2004 a 2014 [4]

El proceso de detección de una persona o casi de cualquier objeto, aunque complejo, lo podemos englobar en 3 grandes bloques: entrada, procesado y salida. Una visión más gráfica y más desglosada lo vemos en la Ilustración 3.3.

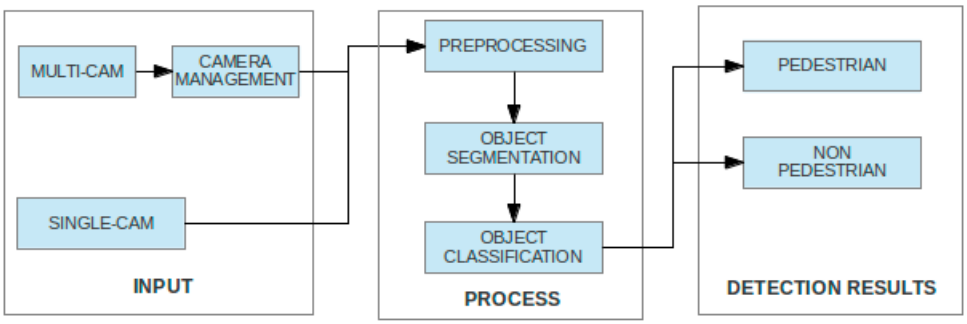

Ilustración 3.3: Proceso de detección de personas [3]

Por lo tanto, analizaremos y estudiaremos la detección de personas desde estos 3 apartados haciendo gran hincapié en los modelos que son usados en el procesado, pero que por sí mismos son un bloque importante.

3.2. Dispositivos de entrada

Aunque existen diferentes dispositivos de entrada, hay uno muy importante en estos últimos años donde se está intentando sacar el máximo aprovechamiento: este es el teléfono móvil (*smartphone*) [5]. Los otros dispositivos usados en menor grado, pero muy importantes son: la cámara laser [6] o múltiple laser [7] como puede ser el Kinect de xBox, la cámara infrarroja [8], la cámara de visión nocturna [9], la cámara Pan-Tilt-Zoom (PTZ) [10], etc. Las más utilizadas son las cámaras de video normales [11] como puede ser una web-cam, una cámara IP o CCTV, dado que son las más baratas y fáciles de encontrar. Algunas investigaciones recientes han tratado de combinar varios tipos de dispositivos de entrada, así como para mejorar la precisión de los resultados de detección: combinación de sensores de escáner láser y cámaras infrarrojas [12], combinación de cámaras IP y cámaras infrarrojas [13], combinación de cámara con sensores (para estimar la distancia de un objeto, por ejemplo) [14]. Otra método a tener en cuenta es la opción de multi-cámara [15], donde aún hay un gran camino por recorrer.

3.3. Modelos

Existen diferentes tipos de modelos con los que trabajar. Podemos trabajar con modelos de movimiento (los cuales no trataremos) o de apariencia. Dentro de lo modelo de apariencia tenemos tres grupos: modelos holísticos (donde el objeto se representa como un todo), modelo basados en partes y modelos mixtos.

3.3.1. Modelo de datos holístico

Los diferentes modelos holísticos difieren entre sí principalmente en la forma utilizada para representarlo y en las características utilizadas. En cuanto a la forma, la más popular es sin duda el rectángulo, donde un gran número de detectores modernos lo utilizan, tales como [16], [17] y [18]. Esta forma rectangular es particularmente eficaz para clases como la de "persona", dado que son innatamente rectangular o comúnmente aparecen en poses canónicas y por lo tanto pueden ser limitadas por cajas rectangulares de manera aceptable. Cabe destacar el trabajo [19] en la detección de peatones, que adoptando un modelo holístico con forma rectangular estableció récords de rendimiento. Sin embargo, debido a la forma de la mayoría de los objetos reales no son rectangulares, usando esta forma, los píxeles del fondo se incluirían también, los cuales producirían ruido. Por lo tanto, una alternativa es utilizar ventanas poligonales para limitar los objetos más estrechamente, como se propone en [20]. Pero las formas poligonales sólo pueden aliviar (pero no solventar) la limitación de las rectangulares, porque en la mayoría de los casos todavía no pueden ajustarse bien a los bordes de los objetos. Además, debido a tener más grados de libertad que un rectángulo, una forma poligonal a menudo aumenta sustancialmente la carga computacional. Para superar este problema, [21] propone utilizar ventanas sin forma, en lugar de las rectangulares o poligonales y explotar los resultados de detección de bordes en la imagen de entrada para aproximarse a la forma de cada objeto destino.

Un modelo holístico, por lo general, se entrena con un conjunto de muestras de entrenamiento que contiene dos grupo: positivos y negativos. Las muestras positivas son descriptores calculados sobre algunos cuadros delimitadores anotados manualmente, los cuales son correctos para un solo objeto de interés. Para asegurar que los descriptores finales tienen la misma longitud, todas las imágenes de entrenamiento primero reajustarán a un tamaño predefinido. Las muestras negativas se calculan a partir imágenes que no contienen el objeto deseado. Clasificadores discriminativos como SVM y variantes se utilizan a menudo para entrenar estos modelos.

3.3.2. Basados en partes

Un modelo típico basado en partes consta de dos componentes: un conjunto de partes y las relaciones geométricas entre ellas (llamada parte topológica). Las partes se refieren, comúnmente, a algunos componentes relativamente rígidos de un objeto, por ejemplo, la cabeza o el antebrazo de una persona. La parte topológica se describe generalmente usando las ubicaciones relativas de las partes y las posibles conexiones entre ellas, por ejemplo, el pie izquierdo está conectado con la pierna izquierda. El primero modelo definido de este modo fue descrito en [22].

En la Ilustración 3.4 vemos las diferentes topologías para un modelo basado en partes, que incluye:

a. La estructura de estrella [23] consiste en una parte central de referencia y un conjunto de partes secundarias conectados a él.

b. La estructura de árbol [24] que restringe la ubicación de cada parte (excepto la principal) a depender sólo de la de su padre.

c. La estructura k-fan [25] que consiste en un conjunto totalmente conectado de partes de referencia ("k" partes) y algunas partes secundarias, cada una de las cuales está conectada a todas las partes de referencia y a nada más.

d. La estructura de constelación completamente conectada [26] que contiene conexiones entre cualquier par de partes.

e. La estructura jerárquica [27] que no es más que una concreción de un grafo acíclico dirigido

f. La estructura de modelos flexibles dispersos (SFM) [28], al igual que en el caso anterior es un ejemplo concreto de un grafo acíclico dirigido. La "k" define el nivel hasta el cual un nodo depende de otro geométricamente.

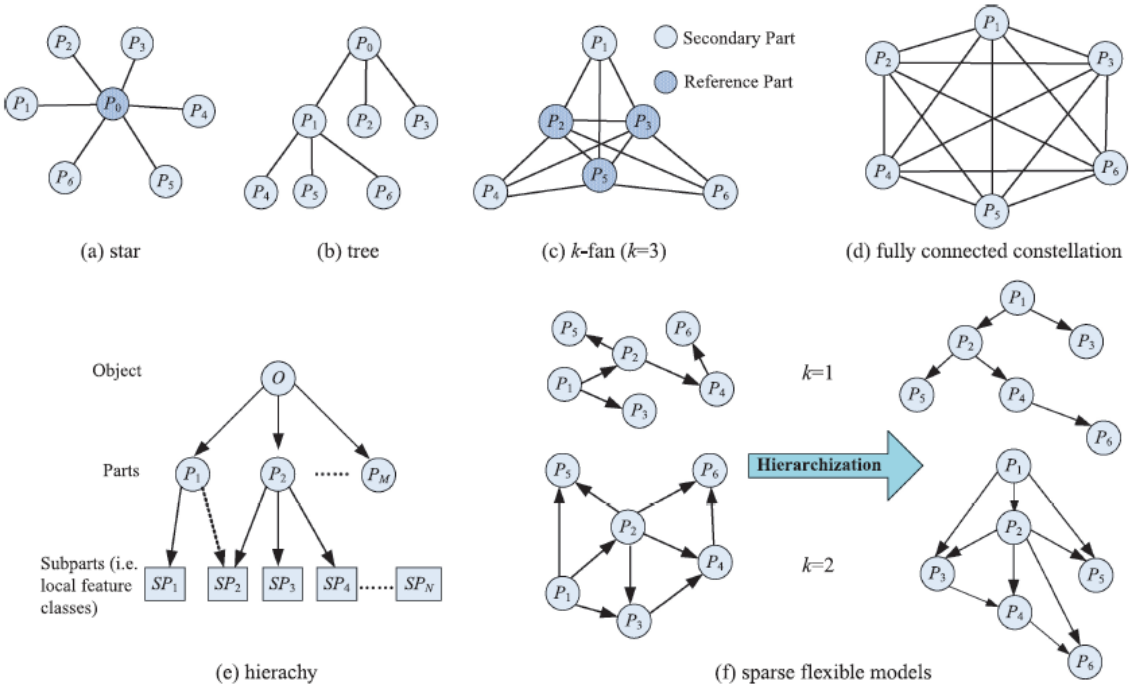

Ilustración 3.4: Representación gráfica de las diferentes topologías de los modelos basados en partes [28]

En general, los modelos complejos son más flexibles, y por lo tanto son capaces de hacer frente a grandes deformaciones de objetos. Sin embargo, tienen más parámetros libres, lo cual hace que el aprendizaje del modelo sea más difícil o incluso intratable. Por lo tanto, se debe llegar a un compromiso entre la complejidad y flexibilidad. Publicaciones recientes, como [29] y [30], han demostrado que las estructuras en árbol y estrella (modelos simples) son bastante eficaces incluso para algunas clases articuladas complejas, como por ejemplo el cuerpo humano.

3.3.3. Modelo mixtos

Un hallazgo importante en la investigación es que los modelos basados en partes pueden adaptarse a las variaciones en la pose, por ejemplo para detectar en escenas deportivas, y también son muy resistentes a las oclusiones si buscas partes determinadas como si fuera la propia persona. Mientras que para categorías menos deformables, como los peatones en escenarios simple, donde suelen estar de pie, el modelo holístico

funciona realmente bien, siendo además menos costoso computacionalmente. En otras palabras, estos dos tipos de modelos son adecuados en diferentes escenarios de aplicación. Un ejemplo destacado es el modelo estructurado en estrella [23] [30], este esquema de combinación es bastante sencillo y rígido:

- Los diferentes nodos de cada parte se colocan al doble de la resolución espacial de la colocación del nodo raíz.
- Cada parte en el modelo mixto se debe conciliar al detectar un objeto.

3.3.4. Conclusiones

Para poder saber qué tipo de modelo usar lo que primeramente hay que saber es la variabilidad que van a tener nuestros datos de entrada con respecto a los modelos almacenados. Como hemos dicho los modelos basados en partes soportan mucho mejor las diferentes poses y las oclusiones que los modelo holísticos, esto se debe a que:

- Las posiciones relativas de las partes de un objeto detectado con modelos basados en parte se pueden utilizar para describir el objeto a representar y ayudar a la estimación.
- Un porcentaje de las partes en un modelo basado en partes es suficiente para indicar que un objeto es válido.

Cabe decir que los modelos holísticos se entrenan de una manera mucho más rápida y más fácil, computacionalmente hablando, pero su uso se limita mayoritariamente a detección en escenarios simples o no muy complejos.

3.4. Procesado de los datos

Una vez tenemos los datos de entrada estos son procesados con diferentes técnicas. Primeramente, se realiza un pre-procesamiento donde ajustamos los datos de entrada a un formato que nos interese. Como paso posterior, algo realmente útil puede ser la detección de la región de interés (ROI), siendo la más simple la substracción del fondo [31], aunque no todos los algoritmos la usan [19], dado que no es del todo eficaz en escenarios dinámicos (aun habiendo sido mejorada [32]). Se suele usar en mayor medida los algoritmos de ventana deslizante, estos son métodos exhaustivos, por lo que suele producir mejores resultados a costa de tiempo computacional.

3.4.1. Técnicas de detección

Podemos distinguir dos tipos de procesamiento hablando en tiempo de computación: en tiempo real [11] y no en tiempo real.

Un breve resumen de las técnicas más usadas lo vemos en la Tabla 3.1, donde cabe decir que lo métodos avanzados que usan histogramas de gradientes orientados (HOG) son los más usados.

De los 26 trabajos que se incluyen en la Tabla 3.1, 17 documentos utilizan el método de clasificación SVM y sus derivados. SVM es una técnica que se puede utilizar para llevar a cabo la clasificación de datos y la predicción. Este método se basa en la teoría del aprendizaje estadístico y los resultados son bastante buenos en comparación con otros métodos. El principio fundamental de esta técnica es encontrar la función separadora (clasificador) que es óptimo para separar los datos en una clase diferente.

Method	Feature	Dataset	Classifier	Results	Year
FPDW [16]	HOG	Caltech	VJ detector	FPPI 37.5%	2010
Gaussian-PSO [33]	HOG	INRIA	Linear SVM	Detection rate 70.3%, 12 fps	2011
Non-background HOG [10]	HOG	INRIA, CAVIAR	Linear SVM	Better and faster	2012
Improved Shape Context [34]	ISC	OSU Infrared Image DB	Hough Voting	Accuracy 90.54%	2012
Blob Motion Statistic [35]	Motion of tracked-blob	PETS	Bayess & SVM classifier	False-Negative 22%	2013
HOG-SVMLight [36]	HOG	Daimler DB	SVM	MR 70%	2013
LBP-HOG [37]	HOG, LBP	Daimler DB	SVM	FPPI 78%	2013
WSPD [38]	HOG	Caltech	SVM	25-480 FPS	2013
DPM [39]	HOG	-	Latent SVM	More accurate	2013
MB-BLP & WCRM [40]	MB-BLP; WCRM	TUD-B; INRIA; Caltech	EFLDA Classifier	Accuracy 90-95%	2013
HOG+B/F [41]	HOG	ETHZ; CVLAB; PETS	Linear SVM	Speed 0.31 s/frame	2013
APD+HLBD [42]	Shape	INRIA; CAVIAR; Munich Airport DB	Multiclass-SVM	Promising	2013
LRMPD [43]	Haar-like, HOG	INRIA	SVM	Accuracy 95%	2013
HOG+ViBe [43]	HOG	INRIA	SVM	Accuracy 92.3%	2013
ABM-HOG [44]	HOG, ABM	INRIA; TUD	SVM	Accuracy 90.2%	2013
Improved Codebook [45]	Shape	Caltech; INRIA	K-means	Accuracy 78.7%	2013
Edgelet-LBP [46]	Edgelet & LBP	TrecVid SED; ETH; CAVIAR; INRIA	AdaBoost	Precision 78-94%	2013
Multifeature Covariance [47]	HOG, FDF	INRIA	LogitBoost	TPR 84.5%-90.9%	2013
Gradient Distribution [48]	Gradient distribution	INRIA	SVM	Accuracy 93.81%	2013
2-stage SVM [49]	HOG	Daimler	Linear SVM	Detection rate 73% of 10-4 FPPW	2013
Scene dependant classifier [50]	Shape, texture	Daimler	Classifier map	Detection rate 80%	2013
CHOG-DOD [51]	Cell-based HOG	INRIA	Linier SVM	21.24 time/frame (PC)	2014
VDPM-MP [52]	Mixture of DPM	Daimler; TUD; Caltech; CVC	Part-based classifier	FPPI 50.4% (Caltech)	2014
Fast Feature Pyramid [53]	Multi-scale HOG	INRIA; Caltech; TUD-B; ETH	AdaBoost	MR(missed-rate) : 40%	2014
Cascaded two layer [54]	Part-based HOG	Daimler DB	Linier SVM	More better than full- body-based	2014
DTM [55]	Part-based HOG	CVC-02	Coarse-to-Fine (CtF) - SVM	FPPI 74%	2014

Tabla 3.1: Métodos de detección de peatones [3]

En 2005 Dalal y Triggs introdujeron un detector usando HOG [19], que en 2008 sirvió como un bloque principal de construcción para el modelo de partes deformable (DPM) por Felzenswalb, McAllester y Ramanan [23]. Este último es parte del algoritmo base de este proyecto, el cual explicaremos en la sección 4.1.

3.4.2. Métodos de clasificación

Lo más importante sin duda es la clasificación de la detección, para ello se puede ir de lo más simple a lo más complejo, últimamente se usan mucho las máquinas de vector de soporte (SVM) [49], y las redes neuronales (NN) [5] [8]. Pero nosotros solo nos centraremos en los modelos latentes y SVM.

Las variables latentes u ocultas se suelen usar en los sistemas modernos para mitigar la carga humana de etiquetado de las muestras, es decir, para reducir la supervisión humana. Las variables latentes son variables no observadas directamente, pero que se deducen de otras variables observadas. Estas se utilizan para describir algunos conceptos intermedios para completar la jerarquía conceptual y explicar mejor algunas observaciones. Un modelo latente a destacar es la LatentSVM propuesto en [23] [30] y que explicaremos con más detalle en la sección 4.1. Los autores definen las ubicaciones de las partes relativas en su modelo estructurado en estrella (sección 3.3.2) para ser variables latentes, que se estiman durante el entrenamiento. Por lo tanto, su modelo propuesto basado en partes puede ser entrenado usando imágenes anotando solamente cuadros delimitadores de la totalidad de los objetos, pero no de cada una de sus partes, es decir, etiquetamos débilmente las muestras de entrenamiento, lo que reducen significativamente la supervisión humana. Estas variables latentes siempre son estimadas por algunos algoritmos iterativos, por ejemplo, descenso de gradiente estocástico [30]. Por lo tanto, el proceso de entrenamiento es costoso computacionalmente. Por otra parte, la convergencia en máximo local depende de la inicialización. Por lo tanto, en cierto sentido, la reducción del trabajo humano es a costa de más cálculos y menor precisión de la estimación.

Yendo a un paso anterior y para entender mejor lo anteriormente explicado, a continuación describiremos como funciona SVM. Las máquinas de soporte vectorial o SVM (Support Vector Machine) es un modelo que representa los puntos de la muestra, separando las clases por un espacio lo más amplio posible. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de su proximidad pueden ser clasificadas a una u otra clase [56]. Un ejemplo simple (2 dimensiones) es el de la Ilustración 3.5 donde podemos ver una separación lineal de dos clases.

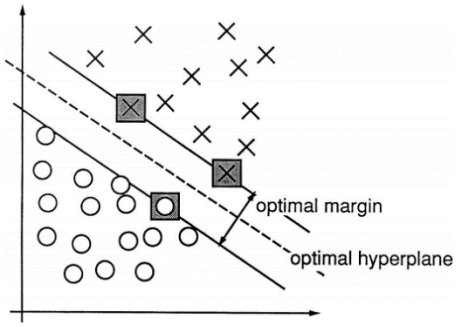

Ilustración 3.5: Ejemplo clasificador SVM para dos clases [57]

Las dos ideas fundamentales para el diseño de un SVM son: la transformación del espacio de entrada en un espacio de alta dimensionalidad (incluso infinita) y la obtención de un hiperplano separador óptimo para ese espacio. La transformación inicial se realiza mediante la adecuada selección de una función kernel. La mayor ventaja de trabajar en un espacio de alta dimensionalidad radica en que las clases que consideremos serán linealmente separables con una alta probabilidad, y por lo tanto, hallar un hiperplano que separe de forma óptima será poco costoso computacionalmente. Además, el hiperplano encontrado vendrá determinado por pocas

observaciones, denominadas vectores de soporte, dado que son las únicas de las que depende la estructura del hiperplano. El mayor problema que tiene la construcción del SVM es la difícil elección de una función kernel adecuada. El mapeo a un espacio de entradas X a un nuevo espacio de características de mayor dimensionalidad matemáticamente se representaría de la siguiente forma:

$$F = \{\varphi(x)|x \in X\}$$

$$donde\ x = \{x1, x2, \cdots, xn\} \rightarrow \varphi(x) = \{\varphi(x)1, \varphi(x)2, \cdots, \varphi(x)n\}$$

Existen diferentes tipos de función kernel, destacan: Polinomial-homogénea, perceptron, función de base radial Gaussiana, sigmoide, etc.

3.4.3. Características usadas

Nos centraremos en HOG y sus variantes. HOG fue propuesto por primera vez en [19]. Esencialmente, es una versión de los descriptores de SIFT. Concretamente, se divide la región de entrada en un conjunto de celdas de 8x8 pixeles espaciadas uniformemente usando una rejilla densa y se usa la superposición de normalización del contraste local para mejorar la precisión. Finalmente, todos los descriptores en bloque se concatenan para formar el descriptor HOG final de la región de entrada.

HOG es una representación muy poderosa para objetos estructurados. En particular, no se ha podido demostrar que ninguna característica supere a HOG para la detección de peatones hasta ahora [1]. Por lo tanto, ha recibido una amplia atención desde de su primer desarrollo, y se han propuesto varias modificaciones. Cabe destacar que, inspirado en el modelo de pirámide espacial con bolsa de características (Bag-of-Features, BoF) [58], [59] propone la representación Pyramid HOG (PHOG), que es esencialmente una pirámide de resolución múltiple de HOGs. Otra modificación notable del HOG es el descriptor de co-ocurrencia de histogramas de Gradientes Orientados (CoHOG) propuesto por [60], que utiliza pares de gradiente orientados discretizados, en lugar de orientaciones discretas, para formar los intervalos del histograma (CoHOG aumenta el poder discriminativo, pero aumenta también el espacio de características a un espacio de dimensionalidad mucho mayor).

3.5. Salida

Y al final de una manera u otra representamos lo obtenido en el paso anterior, es decir, los resultados de la detección. La forma habitual de representación es mediante una lista de blobs (*Binary Large OBJectS*) o cajas que rodean el objeto (*bounding boxes*) que contiene las coordenadas X e Y de la esquina superior izquierda y de la esquina inferior derecha (también en vez de estos últimos se pueden usar el ancho y el alto).

Estos resultados tiene mucha utilidad como por ejemplo en la robótica (o entornos industriales [61]), sistemas de vigilancia (contar personas [62]), análisis de tráfico, sistemas avanzados de asistencia al conductor [5] y muchos otros campos.

3.6. Conjuntos de datos (datasets)

La parte obligatoria para poder obtener esta salida y establecer que técnicas usar es tener varios "*datasets*" una recopilación de estos, en su mayoría gratuitos, la podemos ver en la Tabla 3.2, donde cabe decir que INRIA es de lo más usados.

Datasets	#pedes trian	#neg images	#pos images	Year
INRIA [19]	1.208	1.218	614	2005
ETH [63]	2.388	-	499	2007
TUD-det [64]	400	-	400	2008
TUD- Brussels [65]	1.776	218	1.092	2009
Daimler DB [66]	15.600	6.700	-	2009
Caltech [1]	192.000	61.000	67.000	2009
CVC [67]	2.534	7.650	1.016	2007-2010

Tabla 3.2: Datasets de peatones [3]

INRIA está entre las más antiguas y como tal tiene comparativamente pocas imágenes. Sin embargo, se beneficia de las anotaciones de alta calidad de los peatones en diversos entornos (ciudad, playa, montañas, etc.), por lo que es de las más usadas. ETH y TUD son conjuntos de datos de vídeo de tamaño medio. Daimler no es demasiado usada porque carece de canales de color.

En este proyecto se usará un modelo en el cual se usó INRIA para su entrenamiento, el cual explicaremos en la sección 4.1.2.

3.7. Retos

Aunque hemos mejorado mucho en estos años aún hay muchas cosas que mejorar, como siempre. Corrientes que se están investigando ahora y donde aún queda un gran trayecto podrían ser:

- Uso de "smartcameras", es decir, donde toda la lógica y algoritmia la lleve la propia cámara.
- Uso de los móviles como dispositivos de entrada [5].
- Mejora de la eficiencia para poder trabajar en tiempo real [11].
- Mejora de la precisión en escenas complejas, como puede ser situaciones donde hay mucha gente o grupos numerosos (lugar donde se engloba este proyecto).
- Creación de modelos que incrementando la variabilidad soportada (mejor "recall") no reduzcan la precisión.
- Ampliación y mejora de "datasets" bien estructurados y etiquetados para poder entrenar diferentes modelos y clasificadores.

En definitiva y usando cuatro palabras podemos decir que lo que debemos mejorar es la precisión, el "recall", la eficiencia y el multi-dispositivo.

4. Desarrollo e implementación

4.1. Algoritmo original

4.1.1. Introducción

Tal como hemos adelantado en el estado del arte (sección 3) nuestro algoritmo base parte de [23], es decir, la detección de objetos se basa en mezcla de modelos de partes deformables discriminativamente entrenados (Discriminatively Trained Deformable Part-based, DTDP). Los autores afrontan la detección de objetos en imágenes estáticas, haciendo que sea un gran reto dada la variabilidad que tienen (sobre todo las personas). Para solventar, de alguna manera, toda esta variabilidad se usan los modelos deformables, los cuales están basados en estructuras pictóricas [24] que representan a objetos formados por un conjunto de partes deformables. Dado un objeto, este está dividido en partes y cada una de ellas engloba las propiedades de apariencia locales de su región, mientras que la componente deformable se caracteriza por la conexión entre pares de partes cercanas. En la práctica, esto es difícil de conseguir ya que en las bases de datos más complejas estos modelos están descritos, normalmente, con plantillas rígidas [19]. Como objetivo adicional, intentan realizar modelos, que siendo más ricos, mantienen un alto nivel de rendimiento, por ejemplo modelando los objetos usando gramática visual [68]. Estos modelos gramaticales generalmente están formados por partes deformables con estructuras jerárquicas intrínsecas al objeto, además de tener en cuenta posibles variaciones estructurales.

Mantener el rendimiento usando modelos complejos puede tener muchos problemas. El principal problemas, es que son muy costosos de entrenar ya que en muchas ocasiones se necesita información latente. Tal como explicamos en la sección 3.4.2, las imágenes de entrenamiento sólo disponen de la información de un rectángulo (forma elegida, sección 3.3) alrededor del objeto y no se conoce dónde están sus partes de manera precisa por lo que esta información está oculta durante el entrenamiento. Si las partes estuvieran etiquetadas el entrenamiento sería mucho más simple dado que tendríamos mucha más información, pero como desventaja una persona (o equipo de personas) gastaría mucho tiempo para realizar dicho etiquetado, y hoy día lo que pueda hacer una máquina por nosotros no lo haremos nosotros mismos (a no ser que sea realmente necesario).

La mejora más destacable en cuanto al modelo de [19] es el uso de una estructura basada en partes y definida por un “root” (modelo del objeto entero), un conjunto de filtros de partes y los modelos deformables que les corresponde. En la Ilustración 4.1 vemos el modelo explicado:

a. Modelo root: filtro del cuerpo entero.
b. Modelo de cada una de las partes. A doble resolución que root.
c. Deformación de cada una de las partes.

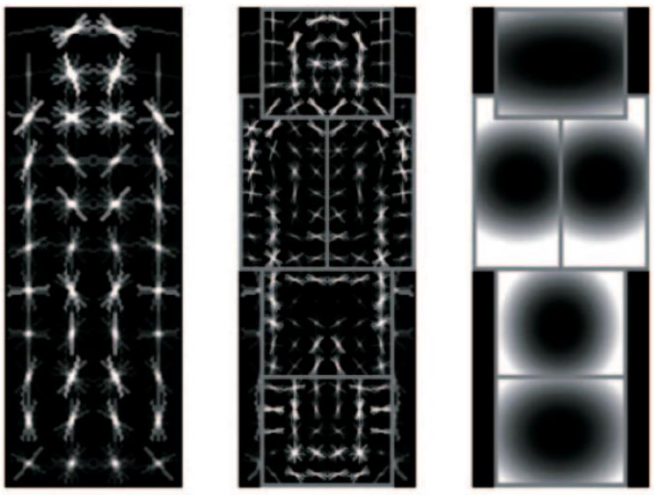

Ilustración 4.1: Modelo de persona con cinco partes y root [30]

Para el entrenamiento del modelo se usa una variante del SVM de instancia múltiple [69], que llamaremos a partir de ahora Latent SVM (LSVM).

Hablando un poco más sobre el modelo, INRIA es el utilizado en el algoritmo original para realizar la detección de personas. Si comparamos la Ilustración 4.1 y la Ilustración 4.2 vemos que el modelo INRIA, posee más partes que el modelo descrito en [30].

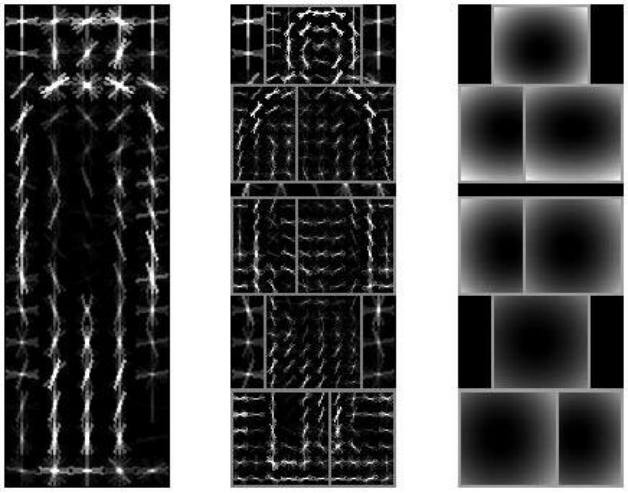

Ilustración 4.2: Modelo de persona con ocho partes y root

En el primer modelo (*paper*) se pueden identificar 5 partes: cabeza, torso izquierdo, torso derecho, parte superior de las piernas y parte inferior de las piernas. Mientras que en INRIA se identifican 8 partes: cabeza, torso superior izquierdo, torso superior derecho, torso inferior izquierdo, torso inferior derecho, parte superior de las piernas, parte inferior izquierda de las piernas y parte inferior derecha de las piernas. INRIA por defecto posee dos modelos: persona visto de manera frontal y dada la vuelta (flip). La principal modificación de nuestro algoritmo se basa en modificar este modelo para detectar parejas. Posteriormente iremos adaptaremos varios parámetros que componen el modelo (no el modelo en sí) a lo largo del tiempo (autoajuste de configuraciones y escala) para generalizar el algoritmo con un mismo modelo.

4.1.2. Modelo

Introducción

El modelo implica filtros lineales que se aplican a los mapas densos de características. Un mapa de características es una matriz cuyas entradas son vectores de características d-dimensionales calculadas a partir de una red densa de ubicaciones en una imagen. Intuitivamente cada vector de características describe una parte de la imagen. En la práctica se utiliza una variación de las características del HOG de [19], aunque es independiente de la elección específica de características.

Un filtro es una plantilla rectangular definida por un conjunto de vectores de peso d-dimensional. La puntuación de un filtro *F* en una posición *(x, y)* en un mapa de características *G* es el producto escalar del filtro y una subventana del mapa de características con la esquina superior izquierda en *(x, y)*:

$$\sum_{x',y'} F[x',y'] \cdot G[x+x', y+y']$$

Debemos definir una puntuación en diferentes posiciones y escalas en una imagen. Esto se hace usando una pirámide de característica, que especifica un mapa de características para un número finito de escalas en un rango fijo. En la práctica se calculan las pirámides de características mediante el cálculo de una pirámide de imagen estándar a través de repetir un suavizado y el submuestreo, y luego calcular un mapa de características de cada nivel de la pirámide de la imagen. De manera visual, en la Ilustración 4.3 vemos dicha construcción.

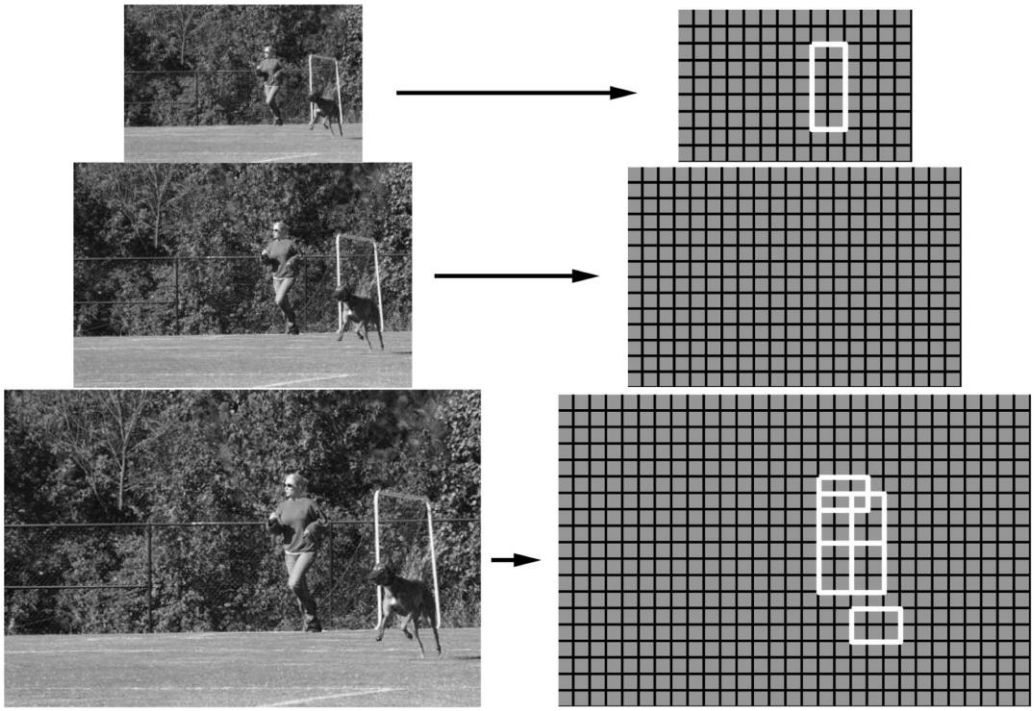

Ilustración 4.3: Pirámide de características con la representación de un modelo de persona. Los filtros de partes están en el nivel de la pirámide que les otorga doble resolución respecto al root [30]

El muestreo por escala en una pirámide de característica está determinada por un parámetro λ, este parámetro define el número de niveles en una octava. Por lo tanto, λ es el número de niveles que tenemos que bajar en la pirámide para llegar a un mapa de características al doble de la resolución de otra. En [30] usan $\lambda = 5$ en el entrenamiento y $\lambda = 10$ en testeo.

Modelo basado en partes deformables

El modelo en estrella se define por un filtro principal (root) que cubre aproximadamente un objeto completo y filtros en alta resolución que cubren las partes más pequeñas del objeto (sección 3.3.2). La Ilustración 4.3 muestra una instanciación de un modelo de este tipo en una pirámide. La ubicación del filtro raíz define una ventana de detección (los píxeles que contribuyen a la parte del mapa de características cubierto por el filtro). Los filtros se colocan λ niveles hacia abajo en la pirámide, por lo que las características de ese nivel se calculan al doble de la resolución que las características en el nivel del filtro principal.

Un modelo para un objeto con n partes se define formalmente con $n+2$ tuplas con forma $(F_0, P_1, \ldots, P_n, b)$ donde F_0 es un filtro raíz, P_i es un modelo para la parte i-ésima y b es un término sesgo (bias, hace que las puntuaciones de múltiples modelos, si existiesen, sean comparables para poder ser combinadas en un modelo mezcla) de valor real. Cada modelo de una parte está definida por una fila de 3 dimensiones con forma (F_i, v_i, d_i) donde F_i es un filtro para la parte i-ésima, v_i es un vector bidimensional que especifica una posición "ancla" para la parte i con relación a la posición del "root" y d_i es un vector de cuatro dimensiones que especifica los coeficientes de una función cuadrática para definir el coste de deformación para cada posible colocación de la parte relativa a la posición de anclaje ("anchor").

Una hipótesis objeto especifica la ubicación de cada filtro en el modelo en una pirámide de características, $z = (p_0, \ldots, p_n)$, donde $p_i = (x_i, y_i, l_i)$ representa la puntuación del píxel (x, y) para la parte i en la escala l. Por lo tanto, se requiere que en el nivel de cada parte el mapa de características en ese nivel se calcule al doble de la resolución del filtro root, $l_i = l_0 - \lambda$ para $i > 0$.

La puntuación de una hipótesis viene dada por las puntuaciones de cada filtro en sus respectivos lugares, menos un coste de deformación que depende de la posición relativa de cada parte con respecto al filtro root, más el término "bias":

$$score(p_0, \ldots, p_n) = \sum_{i=0}^{n} F_i' \cdot \phi(H, p_i) - \sum_{i=1}^{n} d_i \cdot \phi_d(dx_i \cdot dy_i) + b$$

Donde:

$$(dx_i \cdot dy_i) = (x_i \cdot y_i) - (2(x_0 \cdot y_0) + v_i)$$

$$\phi_d(dx \cdot dy) = (dx, dy, dx^2, dy^2)$$

La puntuación de un hipótesis z se puede expresar en términos de un producto escalar, $\beta \cdot \psi(H, z)$, entre un vector de parámetros del modelo β y un vector $\psi(H, z)$:

$$\beta = (F_0', \ldots, F_0', d_1, \ldots, d_n, b)$$

$$\psi(H, z) = \big(\phi(H, p_0), \ldots, \phi(H, p_n)\big), -\phi_d(dx_1 \cdot dy_1), \ldots, -\phi_d(dx_n \cdot dy_n), 1)$$

Esto ilustra una conexión entre nuestros modelos y los clasificadores lineales. Utilizamos esta relación para el aprendizaje de los parámetros del modelo con Latent SVM.

Matching

Para detectar objetos en una imagen se calcula una puntuación global para cada ubicación del filtro "root" de acuerdo con la mejor posible ubicación de las partes:

$$score(p_0) = \max_{p_1,\dots,p_n} score(p_0, \dots, p_n)$$

Las zonas donde el "root" tiene una puntuación alta definen una detección, mientras que las zonas para cada una de las partes con alta puntuación producen una ubicación para el root y define una hipótesis de objeto completo. Al definir una puntuación global para cada ubicación raíz podemos detectar varias instancias de un mismo objeto. Utilizamos la programación dinámica y la transformación de distancias generalizadas [70] [24] (métodos muy eficientes) para calcular las mejores ubicaciones para las partes en función de la ubicación del "root".

La puntuación final para cada píxel y en cada nivel, se obtiene como la suma de la respuesta del filtro "root" a ese nivel más las versiones desplazadas de las respuestas de las partes transformadas y submuestreadas y el término bias:

$$score(x_0, y_0, l_o) = R_{0,l_0}(x_0, y_0) + \sum_{i=1}^{n} D_{i,l_0-\lambda}(2(x_0, y_0) + v_i) + b$$

Siendo:

$$R_{i,l}(x, y) = F_i' \cdot \phi\big(H, (x, y, l)\big)$$

$$D_{i,l}(x, y) = \max_{dx,dy} \Big(R_{i,l}(x + dx, y + dy) - d_i \cdot \phi_d(dx, dy)\Big)$$

$R_{i,l}(x,y)$ es un vector que almacena el i-ésimo filtro en el l-ésimo nivel de la pirámide de características, y $D_{i,l}$ es la máxima contribución de la i-ésima parte a la puntuación de la localización del "root" en la posición (x, y) y nivel l.

Conociendo la posición (x, y, l) del "root" con la mejor puntuación podemos hallar la localización óptima de sus partes. En la Ilustración 4.4 vemos de forma gráfica lo explicado anteriormente, podemos observar que en este caso concreto la cabeza es más discriminativa que la otra parte usada (el hombro derecho).

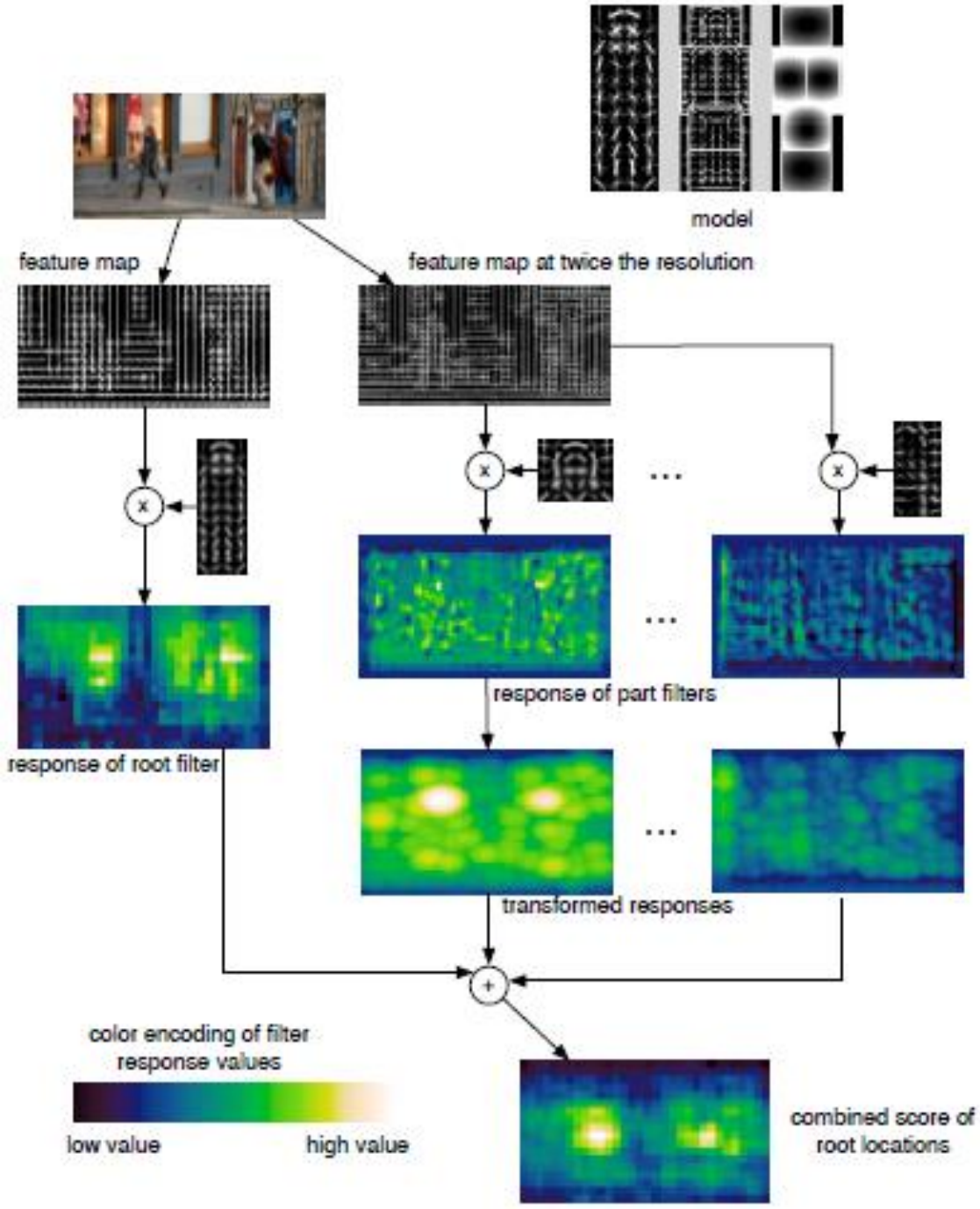

Ilustración 4.4: Proceso de matching en una escala concreta [30]

Mezcla de modelos

Una mezcla de modelos con m componentes se define por una tupla de dimensión m, $M = (M_1, \ldots, M_m)$, donde M_c es el modelo para c-ésima componente.

Una hipótesis de un objeto para una mezcla de modelos especifica una componente de la mezcla, $1 < c < m$, y un lugar para cada filtro de M_c, $z' = (c, p_0, \ldots, p_{n_c})$. Aquí n_c es el número de partes en M_c. La puntuación de esta hipótesis es la puntuación de la hipótesis $z' = (p_0, \ldots, p_{n_c})$ para el c-ésimo componente del modelo.

Como en el caso de un modelo de una sola componente la puntuación de una hipótesis para una mezcla de modelos se expresa por un producto escalar entre un vector con los parámetros del modelo (β) y un vector $\psi(H, z)$. Para una mezcla de modelos el vector β es la concatenación de los vectores de parámetros del modelo para cada componente y $\psi(H, z) = (0, ..., 0, (H, z'), 0, ..., 0)$ con esta construcción:

$$\beta \cdot \psi(H, z) = \beta_c \cdot \psi(H, z')$$

Para detectar objetos en una mezcla de modelos se usa el mismo algoritmo de "matching" descrito en el apartado anterior independientemente para cada componente.

4.1.3. Características HOG

Mapa de características a nivel de pixel

Siendo $\theta(x, y)$ y $r(x, y)$ la orientación y la magnitud (respectivamente) del gradiente de intensidad en un píxel (x, y) en una imagen. Al igual que en [19], se calculan los gradiente utilizando filtros de diferencia finita, $[-1, 0, +1]$ y su transpuesta. Para imágenes en color se utiliza el canal de color con el gradiente de mayor magnitud que definen θ y r en cada píxel.

La orientación del gradiente en cada píxel se discretiza en uno de los valores de p utilizando, ya sea, un contraste sensible (B_1) o insensibles (B_2), usaremos B para referirnos a cualquiera de las dos:

$$B_1(x, y) = round\left(\frac{p\theta(x, y)}{2\pi}\right) mod\ p$$

$$B_2(x, y) = round\left(\frac{p\theta(x, y)}{\pi}\right) mod\ p$$

Se define un mapa de características a nivel de píxel que especifica la magnitud del histograma de gradiente disperso en cada píxel. Por tanto, siendo $b \in \{0, ..., p-1\}$ el rango sobre la orientación de los "bins", el vector de características en (x, y) es:

$$F(x, y)_b = \begin{cases} r(x, y), & b = B(x, y) \\ 0, & otro\ caso \end{cases}$$

Podemos pensar en F como un mapa de borde orientado con canales de orientación p. Para cada píxel seleccionamos un canal por discretizar la orientación del gradiente. La magnitud del gradiente puede ser vista como una medida de la resistencia del borde.

Agregación espacial

Sea F un mapa de características a nivel de píxel para una imagen y siendo $k > 0$ un parámetro que especifica la longitud del lado de un área de imagen cuadrada, podemos definir una red densa de "celdas" rectangulares y características agregadas a nivel de píxel para obtener un mapa de características C basado en celdas, con vectores de características $C(i, j)$ para $0 \leq i \leq \lfloor (w-1)/k \rfloor$ y $0 \leq j \leq \lfloor (h-1)/k \rfloor$. Esta agregación proporciona alguna invariancia a pequeñas deformaciones y reduce el tamaño de un mapa de características.

El enfoque más simple para la agregación de características es mapear cada píxel *(x, y)* en una celda $(\lfloor x/k \rfloor, \lfloor y/k \rfloor)$ y definir el vector de características de una celda como la suma (o promedio) de las características a nivel de pixel en esa celda.

En lugar de la mapear cada píxel a una única celda seguimos [19] y usamos un enfoque de "soft binning", donde cada píxel contribuye a los vectores de características en las cuatro celdas alrededor de ella utilizando la interpolación bilineal.

Reducción de dimensionalidad de características

Hemos recogido una gran cantidad de características HOG 36-dimensionales a diferentes resoluciones de un gran número de imágenes, por lo que realizamos PCA en estos vectores. El análisis de los componentes principales se muestra en la Ilustración 4.5.

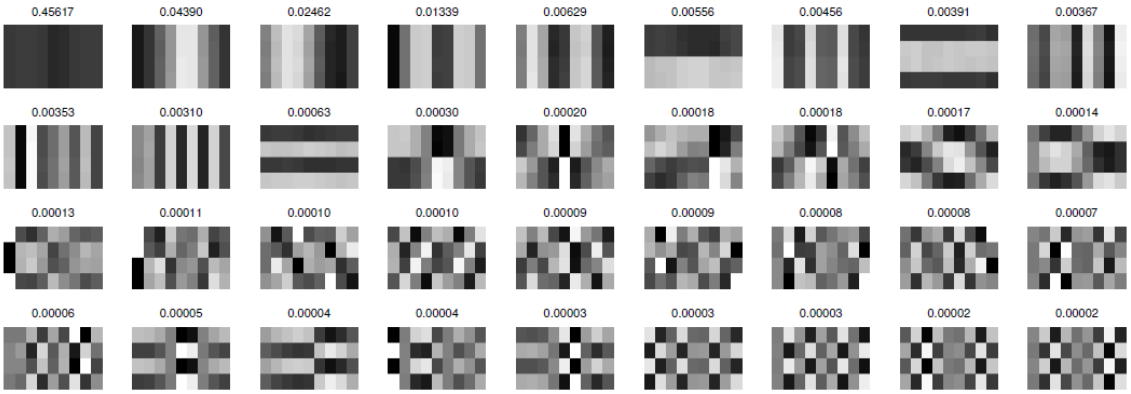

Ilustración 4.5: PCA de características HOG [30]

Podemos ver que cada vector propio se muestra como una matriz de 4x9, de modo que cada fila corresponde a un factor de normalización y cada columna a una orientación. Los valores propios se muestran en la parte superior de los vectores. El subespacio lineal atravesado por los 11 vectores propios capta casi toda la información en un vector de características. El uso de características de menor dimensión conduce a modelos con un menor número de parámetros y acelera los algoritmos de detección y aprendizaje. Sin embargo, esa ganancia la podríamos perder dado que tenemos que realizar un paso de proyección relativamente costoso al calcular las pirámides de características.

Los principales vectores propios de la Ilustración 4.5 tienen una estructura muy especial: son cada constante a lo largo de cada fila o columna de su representación matricial. Así, los principales autovectores se encuentran (aproximadamente) en un subespacio lineal definido por los vectores dispersos que tienen a lo largo de una sola fila o columna de la matriz de su representación.

Siendo $V = \{u_1, \dots, u_9\} \cup \{v_1, \dots, v_4\}$ con:

$$u_k(i,j) = \begin{cases} 1, & j = k \\ 0, & otro\ caso \end{cases}$$

$$v_k(i,j) = \begin{cases} 1, & i = k \\ 0, & otro\ caso \end{cases}$$

Podemos definir una función de 13 dimensiones, tomando el producto escalar de una característica HOG 36-dimensional entre u_k y v_k. La proyección en cada u_k se calcula sumando durante las 4 normalizaciones para una orientación fija. La proyección en cada v_k se calcula sumando las 9 orientaciones para una normalización fija.

Con las características de PCA 11-dimensionales se obtiene el mismo rendimiento que usando las características HOG de 36 dimensiones o las características 13-dimensionales definidas por V. Sin embargo, el cálculo de las características de 13 dimensiones es mucho menos costoso que realizar proyecciones de los principales vectores propios obtenidos a través de PCA dado que u_k y v_k son dispersos. Por otra parte, las características 13-dimensionales tienen una interpretación simple como 9 funciones de orientación y 4 características que reflejan la energía total pendiente en diferentes áreas alrededor de una celda.

4.1.4. Post-procesado

Predicción de cajas delimitadoras

Podemos decir que usando [23] descartamos información potencialmente valiosa adquirida en el uso de un modelo de partes deformable multiescala. Esto se debe a que solo se informa de las cajas delimitadoras derivadas de la localización del filtro raíz. Sin embargo, el uso de los modelos de partes localiza cada filtro de una parte y el del filtro principal. Además, los filtros de cada parte se localizan con mayor precisión espacial que los filtros "root". Está claro, por tanto, que el enfoque original descarta información potencialmente valiosa adquirida en el uso de un modelo de partes deformables multiescala.

Se utiliza la configuración completa de una hipótesis objeto, *z*, para predecir una caja delimitadora para el objeto. En primer lugar se hace una predicción de las cajas de detección a partir del vector de características $g(z)$, de la esquina superior izquierda (x_1, y_1) y de la esquina inferior derecha (x_2, y_2). Para un modelo con n partes, $g(z)$ es un vector de $n + 3$ dimensiones que contiene la anchura del filtro de raíz en píxeles de la imagen (esto proporciona información de escala) y la ubicación de la esquina superior izquierda de cada filtro en la imagen.

Después de entrenar el modelo usamos la salida del detector en cada instancia para aprender cuatro funciones lineales para predecir x_1, y_1, x_2 y y_2 de $g(z)$. Esto se realiza mediante una regresión lineal por mínimos cuadrados, independientemente para cada componente de una mezcla de modelos.

Un ejemplo de la posible mejora que produce este tipo de predicción lo vemos en la Ilustración 4.6 donde podemos observar que la detección (en este caso de un coche) es bastante precisa.

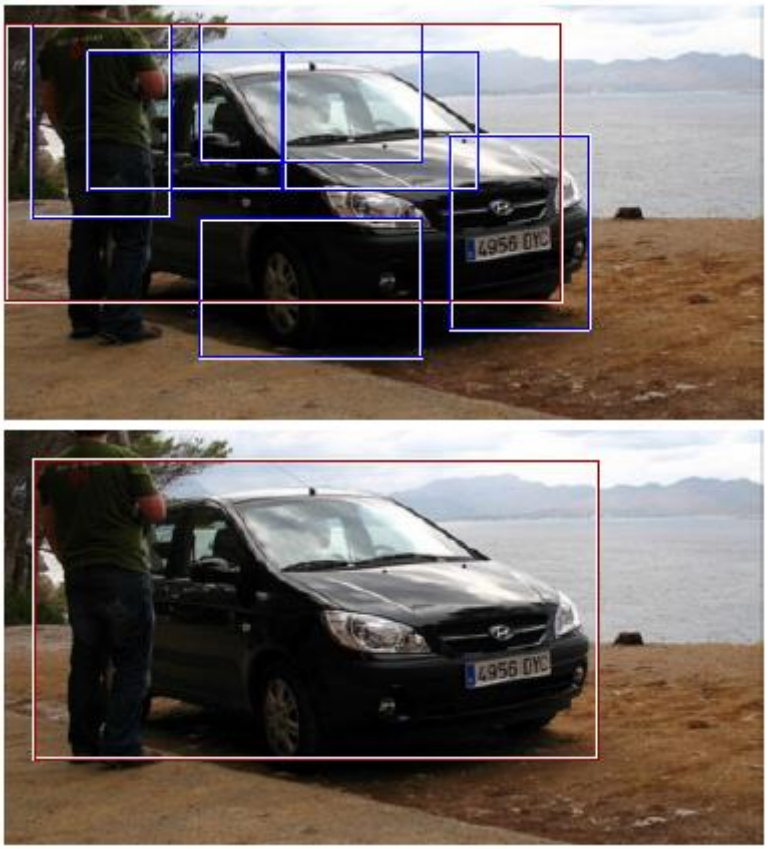

Ilustración 4.6: Prediccion de la caja delimitadora de un coche usando la configuración del objeto [30]

Non-Maximum Suppression

Al detectar objetos en diferentes escalas y orientaciones se producen bastantes detecciones muy cercanas y solapadas para un mismo objeto, para eliminar este exceso de cajas que delimitan un mismo objeto se usa un método de supresión (Non-Maximum Suppresion, NMS). Para ello, ordenamos las detecciones por puntuación y las que se solapen más de un porcentaje definido (50% en la práctica) las eliminamos dejando la de puntuación más alta. Entendemos como solapamiento las cajas que cumplen las dos condiciones mostradas en la Ilustración 4.7. Con este método se eliminan muchísimos falsos positivos, pero puede que con objetos muy ocluidos uno con otro no logremos detectar uno de ellos, dado que su solape supere el porcentaje establecido.

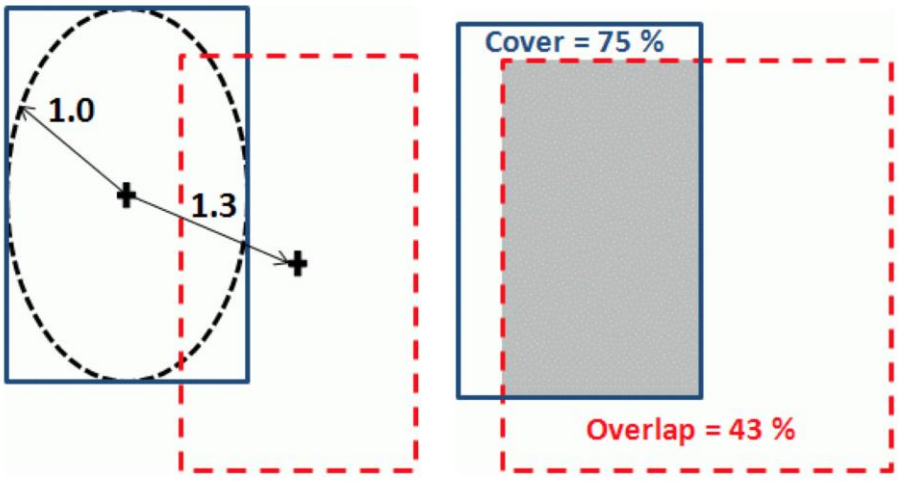

Ilustración 4.7: Solapamiento, cobertura y distancia relativa

4.1.5. Conclusiones y trabajo futuro

En este algoritmo base se describe un sistema de detección de objetos a base de mezclas de modelos de partes deformables multiescala. Se basa principalmente en nuevos métodos de entrenamiento discriminativo de clasificadores que hacen uso de la información latente, apoyando por tanto muchas otras publicaciones de los últimos años. También se basa en gran medida en los métodos eficaces para la búsqueda de modelos deformables en las imágenes. El sistema resultante es a la vez eficaz y preciso, estando entre lo más altos del estado del arte (incluso en conjuntos de datos complejos).

Los modelos son capaces de representar clases de objetos muy variables, pero se podría avanzar hacia modelos mucho más ricos. Se puede considerar jerarquías de parte más profundas (partes con partes) o mezcla de modelos con muchas más componentes. Se debería construir modelos basados en la gramática que representan objetos con estructuras jerárquicas variables.

4.2. Nuestro algoritmo

Explicaremos a continuación los cambios realizados en el algoritmo original para aumentar su eficacia. En adelante, este algoritmo pasará a denominarse "Hierarchical Detector of Groups of Person" (HDGP) ya desarrollado por VPULab (laboratiorio de investigación de la Universidad Autónoma de Madrid) y con la modificación del autoajuste de parámetros se denominará "Self-tuning HDGP". La explicación posterior, es por tanto sobre un algoritmo ya desarrollado que hemos tenido que estudiar y entender detenidamente. Y que, por tanto, tras ver algunas de las deficiencias hemos realizado los cambios de código oportunos y la adición del autoajuste.

4.2.1. Modelo

Jerarquía de grupos

Una de las diferencias que se introducen en el modelo con respecto al algoritmo original es el diseño de una jerarquía de grupos. Esta jerarquía agrupa la información de cada persona que pertenezca a la pareja en el centro geométrico de la persona principal, y así conseguir que las personas que tienen mayor dificultad en ser detectadas mejoren su puntuación gracias a la información de la personas principal de la pareja a la que pertenece.

En dicha jerarquía diferenciaremos dos tipos de personas, la persona principal (Main Person, MP) que será la persona a partir de la cual se detectará la otra persona de la pareja, llamada persona secundaria (Secondary Person, SP).

Sea el "anchor" un vector tridimensional *(x, y, r)* que define la posición *(x, y)* de cada parte con respecto al "root" y el "anchor_shift" un vectores bidimensionales *(Δx, Δy)* que definen la posición relativa del "root" de la SP con respecto al "root" de la MP.

Hemos definido un conjunto de "anchors_shift" de nueve vectores. En el eje horizontal (*x*) se han definido los "anchor_shift" entre -6 y +6 que corresponden al desplazamiento que provoca que la SP, ocluida parcialmente por la MP, deje ver la mitad de su cuerpo. En el eje *y*, los "anchor_shift" han sido definidos, de igual manera, entre -6 y +6, que corresponde al desplazamiento del tamaño de una cabeza. Tanto el paso horizontal como vertical es de 6 "anchors". De esta forma definimos un rango de búsqueda de la SP a partir de la MP que puede verse en la Ilustración 4.8. Es necesaria la inclusión de un desplazamiento de $\Delta x=0$ y $\Delta y=0$ para contemplar a las personas que

no estén formando una pareja, dado que de este modo son tratadas como una pareja formada por la MP consigo misma.

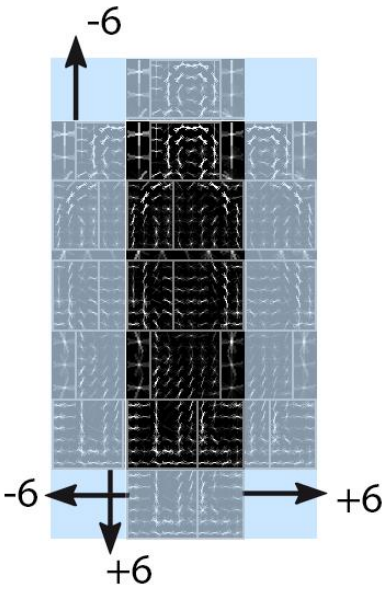

Ilustración 4.8: Zona de búsqueda de la SP definida por los "anchor_shift"

Jerarquía de partes

Creamos una jerarquía de partes para poder decidir que partes pueden definir a una persona por si solas. Esto se debe a que, como de manera predomínate, la SP va estar ocluida por la MP, partes como los pies dan poca información (más bien introducen ruido). Se han creado 5 configuraciones diferentes para una MP, las cuales vemos en la Ilustración 4.9. Aunque de una manera lógica, tiene sentido usar tan solo las configuraciones 12, 13 y 14, en el autoajuste de configuraciones usaremos todas ellas para que de manera automática veamos cuales son mejor descartar, dado que dependerá del tipo de escenario.

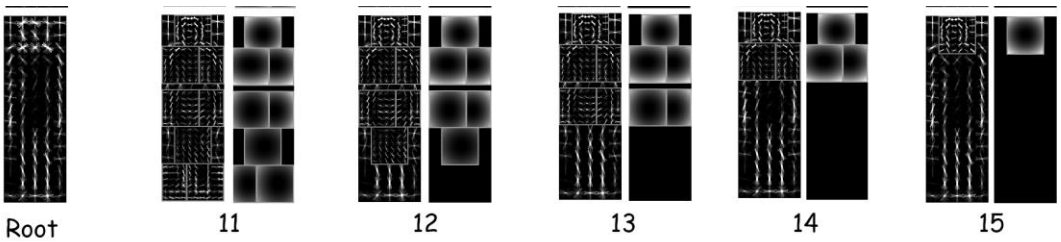

Ilustración 4.9: Configuraciones para la MP

Para la SP existen 3 tipos, que dependerán del “anchor_shift”:

- Sí $\Delta x > 0$: Dado que corresponde a un desplazamiento horizontal hacia la derecha, el SP tendrá una configuración como el MP, pero sólo con las partes que quedan a la derecha del eje longitudinal de la persona.
- Sí $\Delta x < 0$: Dado que corresponde a un desplazamiento horizontal hacia la izquierda, el SP tendrá una configuración como el MP, pero sólo con las partes que quedan a la izquierda del eje longitudinal de la persona.
- Sí $\Delta x = 0$: Dado que no hay movimiento en horizontal, el único movimiento posible es que se vea la parte superior del SP por encima del MP (ver la parte inferior es físicamente imposible), lo que corresponde a un desplazamiento

vertical y, por lo tanto, la configuración del SP constará de tres partes: cabeza, hombro derecho y hombro izquierdo.

Volviendo a la Ilustración 4.8 se puede ver un ejemplo con la configuración 11 para la MP (en negro) y los tres posibles modelos de la SP (en azul, ignorando la parte inferior). Cabe añadir que todas las configuraciones utilizadas usan el filtro "root", tanto para la MP como para la SP.

4.2.2. Detector

Mapas de confianza

El sistema propuesto calcula *M* mapas de confianza para cada escala de la pirámide de características, con $M = 2 \cdot anchor\_shift$. Cada píxel de estos mapas indica la puntuación del algoritmo para un "anchor_shift", componente y escala determinada. Una puntuación alta indica una alta probabilidad de que ese píxel corresponda a una detección de un grupo de personas para el "anchor_shift", componente y escala de dicho mapa.

El algoritmo original recoge todas las puntuaciones de las ocho partes y del "root" en un único píxel en el centro geométrico del modelo para un nivel determinado. Para aplicar las jerarquías diseñadas y poder detectar personas individuales que forman parte de un grupo, el mapa de confianza de la SP es igual que el de la MP desplazado los valores que indican el "anchor_shift", si la MP y la SP tuviesen la misma configuración de partes.

Una vez tenemos todos los mapas de confianza debemos combinarlo para obtener una puntuación única y no $M \cdot escalas$ puntuaciones. Pero dado que tenemos mapas de confianza calculados con diferentes modelos (modelos con diferente configuración de partes), no podemos combinarlos directamente, dado que los rangos de valores de los mapas son diferentes. Para ello, y basándonos en [71], calculamos la confianza total a partir de la fusión de las confianzas ponderadas de varios detectores de diferentes configuraciones de partes del cuerpo. Para ello, calculamos la distribución de densidad de probabilidad que aporta cada parte del modelo, para combinar resultados de detecciones de personas obtenidas con diferentes configuraciones de partes, como vemos en la Ilustración 4.10.

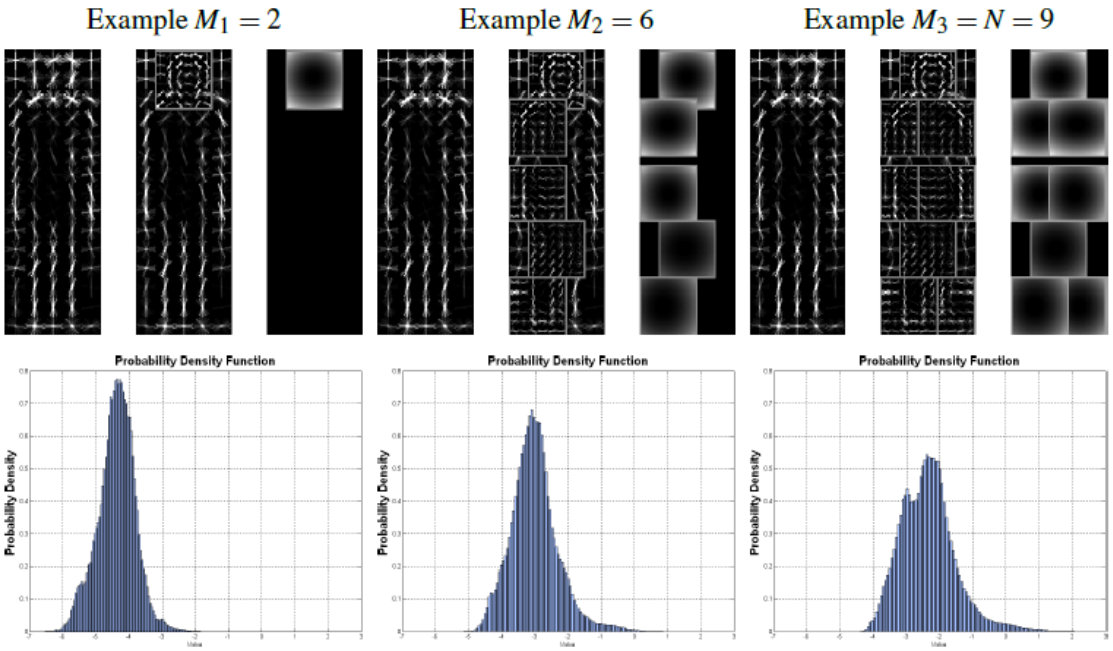

Ilustración 4.10: Ejemplos de configuraciones de modelos de persona y sus distribuciones de densidad de probabilidad [71]

Los mapas de probabilidad tienen el mismo significado que los mapas de confianza, pero convierten la puntuación de confianza en una probabilidad, por lo que tenemos un valor entre 0 y 1 para cada uno de los diferentes modelos, por lo tanto ya podemos realizar la combinación.

Para realizar la combinación, primeramente calculamos la combinación de los mapas de probabilidad con el *anchor_shift = 0*, $S_{probabilidad}(x,y,l,0)$, con cada uno de los mapas generados para cada desplazamiento del "anchor_shift", $S_{probabilidad}(x,y,l,i)$. Para ello, usamos la siguiente formula en un detector de parejas:

$$S_{probabilidad}(x,y,l,i) = \frac{S_{probabilidad}(x,y,l,0) + S_{probabilidad}(x,y,l,i)}{2}$$

Por último, nos quedamos con el máximo de cada una de la probabilidades de entre cada una de las combinaciones anteriores y el mapa con *anchor_shift = 0*.

En la Ilustración 4.11 vemos la aplicación del método explicado para una escena donde no se produce solapamiento. Como vemos, esto se ha realizado para una escala concreta y con los "anchor_shift" (6,0) y (-6,0) para poder realizar su representación de manera sencilla, también definimos los puntos negros como personas que realmente están en la escena y los puntos rojos para los que no (posibles parejas). De esta figura cabe destacar que, aun no produciéndose solapamiento, no se penaliza la puntuación de las personas que realmente existen, aunque si se añaden puntuaciones inexistentes que pueden perjudicar los resultados con umbrales muy bajos.

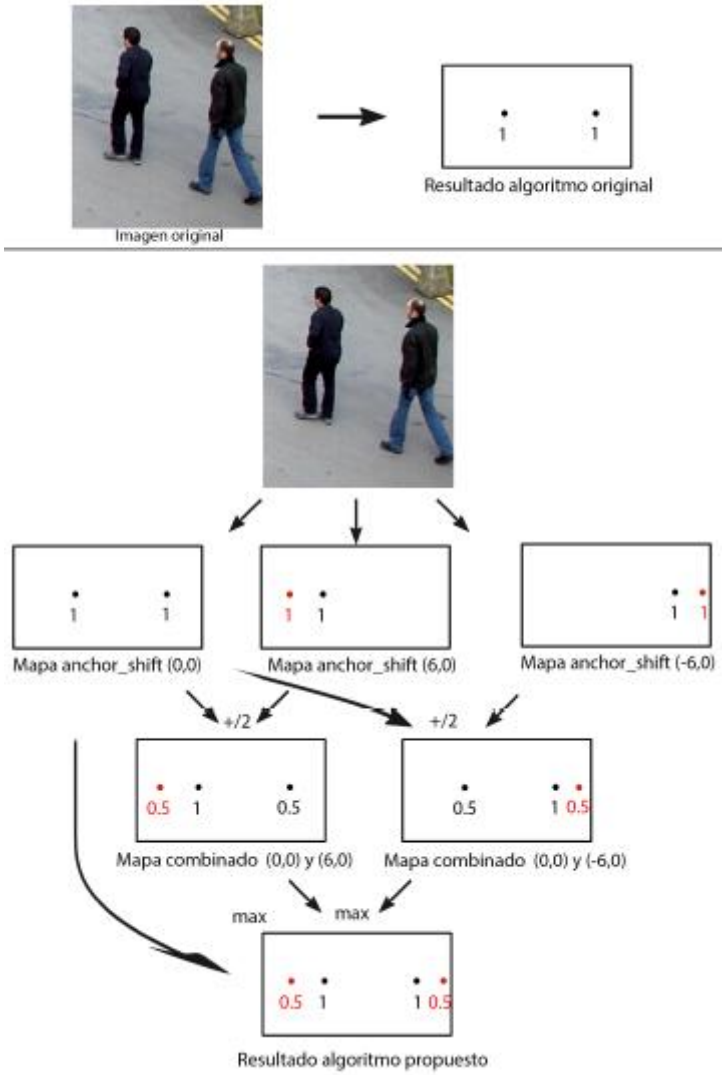

Ilustración 4.11: Esquema de creación y combinación de los mapas de probabilidad para una imagen sin solapamiento

En la Ilustración 4.12 vemos el mismo tipo de planteamiento pero con una escena donde si se produce solapamiento, destacando que la persona ocluida (sobre un 50%) aumenta su puntuación sin penalizar la de su pareja.

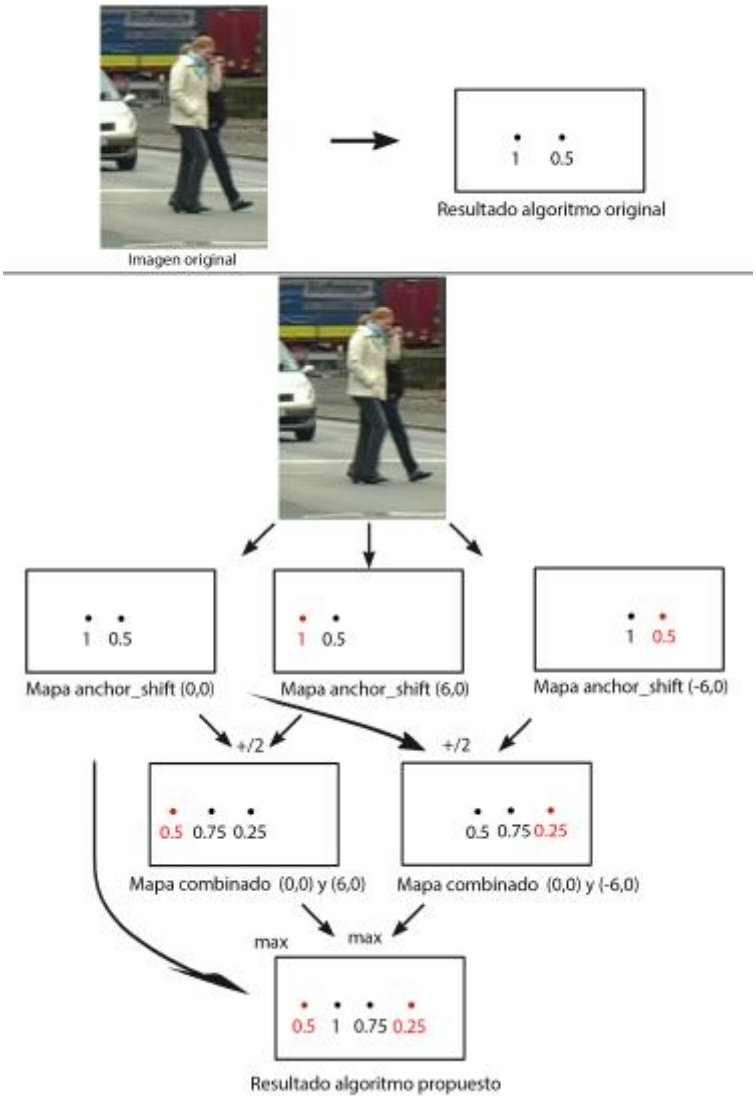

Ilustración 4.12: Esquema de creación y combinación de los mapas de probabilidad para una imagen con solapamiento

Por último, y por completitud, en la Ilustración 4.13 vemos los mapas de probabilidad para esta última escena en la que si se produce oclusión, observamos en el círculo rojo como gracias a la información de pareja su probabilidad es mayor llegando a ser detectada (MP: cuadro verde; SP: cuadro azul).

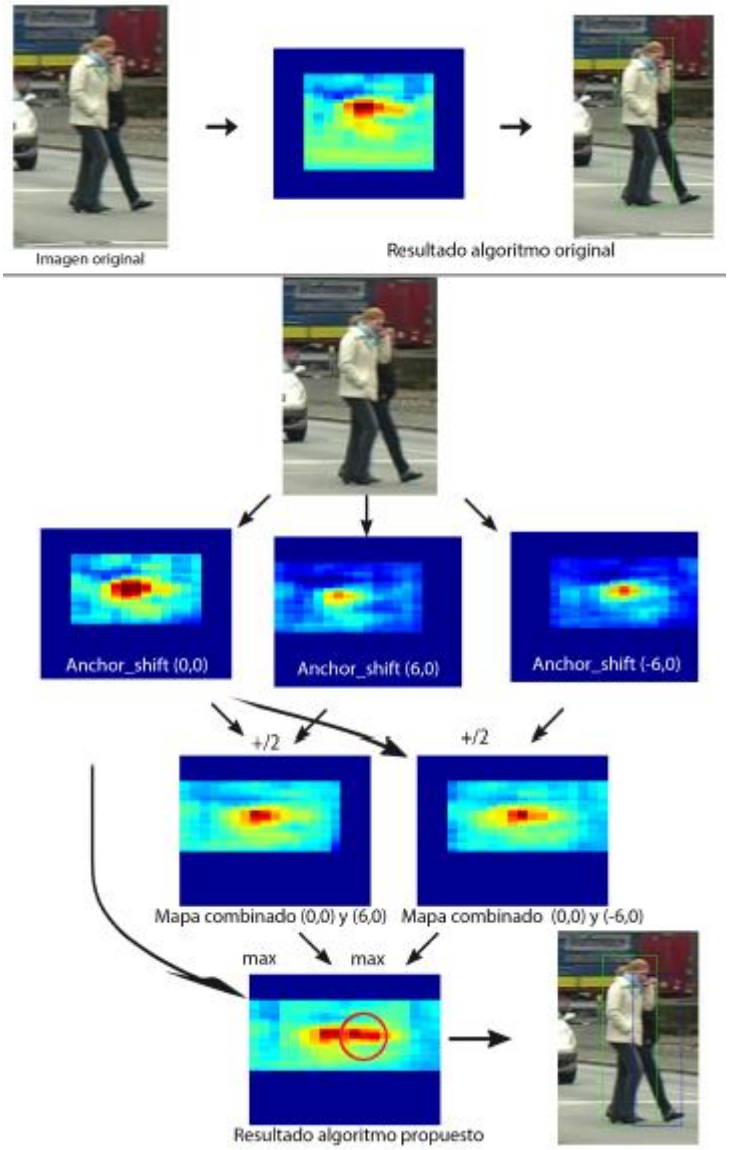

Ilustración 4.13: Representación de creación y combinación de los mapas de probabilidad para una imagen con solapamiento

Caja delimitadora de la pareja

Dado que los mapas, en este algoritmo, lo que indican es la detección de una pareja (el punto está en el centro geométrico de la MP), es necesario deducir donde se encuentran las SP. Para ello, primeramente se usa el método explicado en el algoritmo base (sección 4.1.4) para la predicción de cajas delimitadoras basada en localización de la partes. Después, a partir de esta caja detectada (caja delimitadora del MP), el nivel de la pirámide en la que fue detectada y el "anchor_shift" somos capaces de extrapolar la caja delimitadora de la SP.

En conclusión, los puntos de SP son los mismos que los de la MP sumando un Δx y Δy a los respectivos puntos x_1, x_2, y_1 e y_2. Estos Δs los definimos del siguiente modo:

$$\Delta x = anchor\_shift_x \cdot \frac{sbin \cdot (\frac{pad_x}{w})}{scales(l)}$$

$$\Delta y = anchor\_shift_y \cdot \frac{sbin \cdot (\frac{pad_y}{h})}{scales(l)}$$

Donde:

- *sbin* es el factor de escalado del primer nivel de la pirámide de características con respecto a la imagen original.
- pad_x y pad_y son variables que indican el tamaño del modelo de persona en píxeles en el primer nivel de la pirámide de características (valores de 5 y 15 respectivamente para INRIA)
- *scales(l)* es el factor de escalado de la imagen en el nivel *l* con respecto al tamaño de la imagen en el primer nivel de la pirámide.
- *w* y *h* corresponden al tamaño en "anchors" del modelo de persona (valores de 10 y 30 respectivamente para INRIA).

Non-Maximum Suppression

Dado que, como hemos repetido reiteradas veces, estamos detectado parejas, el NMS explicado en el algoritmo base (sección 4.1.4) no nos valdría. El NMS del algoritmo original elimina las cajas que se solapan más de una 50% (Ilustración 4.7), por lo que descartaría cajas que hemos detectado explícitamente con las modificaciones realizadas. Para poder hacer un filtrado en este algoritmo dividimos el NMS en dos partes, siendo realmente dos NMS por separado:

1. En vez de realizar el NMS con un solape del 50% y para toda la caja, usamos la información de la partes, más concretamente la de la cabeza. Por tanto, hacemos un NMS mucho más estricto (solo un 1%, por ejemplo) pero solo contemplando el solape de la cajas que delimitan la cabeza de cada una de las detecciones.
2. Realizamos un segundo NMS, esta ve si de la caja delimitadora de la persona completa, muy relajado donde permitimos bastante solape (un 90% por ejemplo).

4.2.3. Parámetros autoajustables

Introducción

Esta modificación, no solo debe mejorar los resultados y bajar el tiempo de computación, sino que también debe hacer bastante más general el algoritmo para diferente tipos de conjunto de datos.

Aunque existen diferentes parámetros que podemos autoajustar, los que tienen un mayor peso, tanto en el resultado como en el tiempo de computación, son la escala y las configuraciones que usemos. La idea principal para el autoajuste de ambos parámetros es la misma, basarnos en un determinado número de imágenes anteriores para ver a que tienden los parámetros.

La mayor justificación para realizar el autoajuste de la escala es que al ser grupos de personas, estas estarán cercanas entre sí, y por lo tanto podemos asumir sin mucho error que la variación de escala no será muy grande. Con respecto a la selección de configuraciones, dado que es un parámetro que depende mucho del número de personas, la disposición entre ellas y el punto de vista de la cámara (que será siempre

fija y no móvil), al analizar dichas configuraciones en espacios cortos de tiempo podemos asumir que no varían demasiado.

Cabe mencionar que, en esta primera aproximación, el tiempo de ejecución será idéntico. Es obvio, que si tenemos menos modelos que analizar y combinar el tiempo de computación se reducirá, pero en este caso no se dejará de realizar ninguna operación. Simplemente, se filtrará el resultado final, reduciendo el número de cajas delimitadoras detectadas (que seguramente eran falsos positivos), de este modo podemos recuperar fácilmente configuraciones o escalas descartadas conforme avanzamos en el tiempo.

En el momento final del algoritmo (antes de almacenar las cajas delimitadoras) lo que poseemos es la información de cada una de las detecciones, esta información consta de:

- Información sobre la caja delimitadora (x_1, x_2, ancho y alto).
- Escala en la que se ha encontrado la detección.
- Configuración con la que se ha obtenido la detección.
- Puntuación de la detección.

A continuación veremos el proceso de autoajuste para cada uno de los parámetros elegidos.

Autoajuste de las escalas

Nuestro objetivo es tener un vector de tamaño igual al número de escalas normalizado (que sume 1) con la probabilidad de uso de cada determinada escala.

Primeramente, debemos definir 3 parámetros:

- El valor umbral a partir del cual se tendrá en consideración la detección para su acumulación: Este valor hará un pre-flitrado en las detecciones encontradas para no introducir mucho ruido en nuestras probabilidades. Esto se debe a que muchas detecciones malas pueden superar una detección buena. Este valor se define como indica el autor del algoritmo DTDP para el mostrado de las cajas delimitadoras (-0.3).
- El valor umbral que define el límite de escalas a utilizar: Dado que nuestro vector sumara 1 (100%) debemos escoger un umbral con la información que queremos mantener. Es decir, por ejemplo, si nuestro umbral de 0.7 y el vector de escalas ([1 2 3 4 5]') es [0.2, 0.14, 0.4, 0.1, 0.16]', tan solo cogeríamos las escalas 1, 3 y 5, que es lo mínimo necesario para sobre pasar el umbral cogiendo siempre los de valor más alto primero. En la sección 5.2.3 veremos cómo afecta los cambios de este umbral a los resultados.
- Número de fotogramas anteriores que queremos tener en cuenta (*numFrames*): Este parámetro definirá cuanta historia queremos recordar. El objetivo es por tanto tener una matriz en la que cada columna corresponda a un fotograma y haya tantas columnas como el número elegido, al llegar el fotograma *X+1*, este se pondrá en el lugar del primer insertado (la primera columna en este caso) como en un modelo FIFO (First In Fist Out, el Primero en Entrar es el Primero en Salir). El tamaño en filas de esta matriz será el mismo que el de nuestro vector final, dado que ese vector final no es más que la suma por filas de cada una de las columnas con su posterior normalización (número de escalas en este

caso). De este modo lo que tenemos es una ventana corrediza de ese número de fotogramas. Este número se define a partir del número de fotogramas (duración, *nFrames*) que tiene el vídeo, dado que no empezamos a actuar (filtrar) hasta que la matriz no está llena. La condiciones de longitud usadas son las siguientes:

$$numFrames = \begin{cases} 10 & nFrames < 100 \\ 15 & 100 \leq nFrames < 500 \\ 20 & 500 \leq nFrames < 1000 \\ 25 & nFrames \geq 1000 \end{cases}$$

Para poder conseguir la matriz explicada con anterioridad es necesario, primeramente, convertir la puntuación (previamente filtrada) que tenemos en probabilidad usando la tabla de distribución (la probabilidad nunca será 0, dado que se inicializará con *1 / numeroEscalas*), de igual manera que se realizaba en la detección (sección 4.2.2). Posteriormente y dado que si, por ejemplo, las escalas 3 y 5 tienen una alta probabilidad sería muy restrictivo quitar la escala 4, aunque su probabilidad sea muy baja, por tanto se realiza una media a 3 con sus vecinos. Tras ello solo queda realizar el almacenaje de la manera explicada, la suma para cada "frame" y su filtrado en las detecciones finales. A modo de resumen y para que quede mucho más claro, estos son los pasos a seguir:

1. Filtramos nuestras detecciones para que su puntuación supere un determinado umbral y así no añadir ruido.
2. Convertimos nuestro vector de puntuaciones a un vector de probabilidades.
3. Inicializamos un vector de tamaño *numeroEscala x 1* con valor *1 / numeroEscalas*.
4. Sumamos en el vector anterior las probabilidades de una misma escala en ese "frame".
5. Acumulamos en una matriz la probabilidad de todas las detecciones que han usado una escala determinada. Por lo tanto tenemos una matriz de tamaño *numeroEscalas x numFrames* con la confianza de cada una de las escalas para cada "frame" hasta *numFrames*.
6. Una vez esta rellena, sumamos por filas las columnas de la matriz anterior, obteniendo un vector columna con las suma de las probabilidades. Es decir tendremos un vector de *numeroEscalas x 1* con las suma de las probabilidades para cada una de las escalas en los *numFrames* "frames".
7. Utilizamos la media a 3 para que escalas muy cercanas tengas valores parecidos.
8. Normalizamos el vector dividiendo cada una de la filas entre la suma del vector, por lo que la suma de este vector será 1.
9. Aplicamos el umbral para quedarnos con las escalas que generan detecciones más probables.
10. Filtramos en las detecciones.

De modo más visual, en la Ilustración 4.14 vemos como son los histogramas en diferentes "frames" de un mismo video.

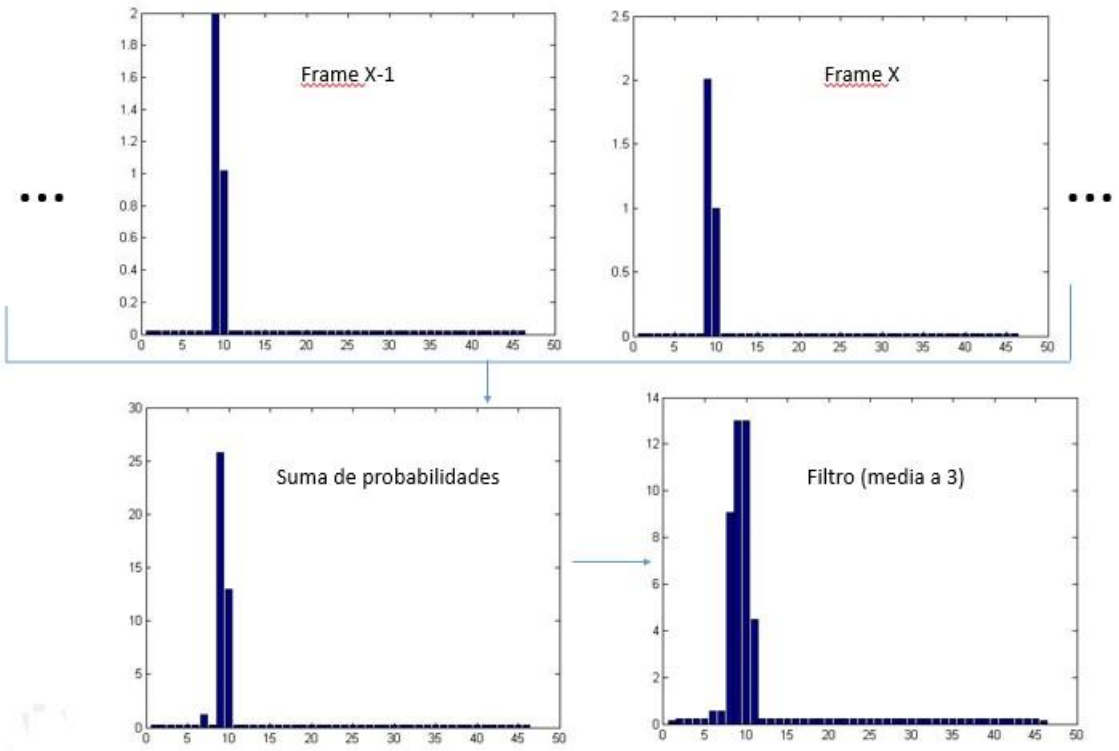

Ilustración 4.14: Ejemplos en el autoajuste de escalas

Cabe destacar una limitación en este autoajuste debido a la formación de las escalas. Esta limitación impide un buen autoajuste si el tamaño de las imágenes del conjunto de entrada varía en el tiempo, dado que la formación de la pirámide de características depende del ancho y alto de la imagen, lugar de donde se obtiene la escala.

Autoajuste de las configuraciones

Nuestro objetivo, al igual que en el caso anterior, es tener un vector de tamaño igual al número de configuraciones usadas (en nuestro caso será siempre 5, dado que usaremos las configuraciones 11, 12, 13, 14 y 15, Ilustración 4.9) normalizado (que sume 1) con la probabilidad de uso de cada determinada configuración.

En este caso solo debemos definir 2 parámetros:

- El valor umbral a partir del cual se tendrá en consideración la detección para su acumulación: Explicado en la sección de más atrás. Esta vez se aplicará un valor algo menos restrictivo (-1). Esto se debe a que solo hay 5 configuraciones y la mayor parte de las veces aportan información útil.
- Número de fotogramas anteriores que queremos tener en cuenta (*numFrames*): Explicado en la sección de más atrás.

Podemos ver que nos ahorramos un valor umbral con respecto al autoajuste de las escalas. Esto se debe, a que en este caso, es probable que el uso de todas las configuraciones (11, 12, 13, 14 y 15) sea la que obtiene los mejores resultados, y añadir este límite no permitiría mantenerlas todas. Este umbral en el autoajuste de las escalas si tiene sentido dado que, a una resolución igual, las escalas usadas deben de ser siempre muy parecidas, habiendo muchas inútiles y que por tanto debemos descartar con este umbral.

Para poder conseguir la matriz explicada con anterioridad es necesario, primeramente, al contrario que en el caso anterior o es necesario convertir la puntuación que tenemos en probabilidad dado que la detección con esa configuración aplicando el filtro ya es lo suficientemente restrictivo. A modo de resumen y para que quede mucho más claro, estos son los pasos a seguir:

1. Filtramos nuestras detecciones para que su puntuación supere un determinado umbral y así no añadir ruido.
2. Acumulamos en una matriz la presencia de detecciones fiables de cada configuración en número determinado de "frames" (hasta *numFrames*).
3. Una vez esta rellena, cogemos el máximo de cada columna para cada fila. Por tanto obtenemos un vector columna de tamaño *nConfigs x 1* con un 1 si la configuración se ha usado en los *numFrames* anteriores y 0 en caso contrario.
4. Nos quedamos con las configuraciones que tiene un 1 en el vector anterior.
5. Filtramos en las detecciones.

De modo más visual, en la Ilustración 4.15 vemos como son los histogramas en diferentes "frames" de un mismo video.

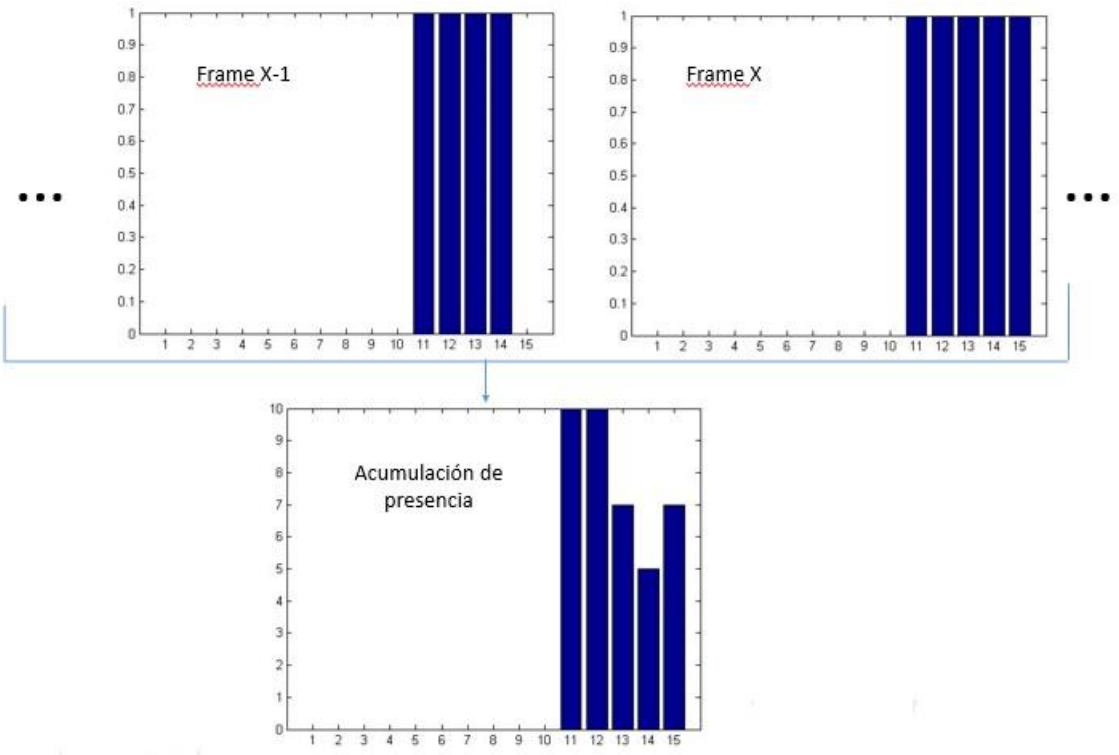

Ilustración 4.15: Ejemplos en el autoajuste de configuraciones

5. Evaluación y pruebas

En primera instancia hablaremos un poco sobre los datasets utilizados, analizando los videos, su complejidad y la utilidad para nuestro proyecto (sección 5.1). En la sección 5.2 explicamos la métrica utilizada y realizamos las comparaciones entre algoritmos. Por último, en la sección 5.3 deducimos las conclusiones más importantes obtenidas en la sección 5.2.

5.1. Conjuntos de datos usados

Usaremos principalmente 3 conjuntos de datos (datasets): PETS, TUD y MPII-2Person. En las siguientes secciones explicaremos en detalle cada una de estas bases de datos y el motivo de su elección.

5.1.1. PETS

Esta base datos ha sido selecciona porque es una de las más extendidas en la actualidad y dispone de secuencias de vídeo en las que hay grandes grupos de personas. Desde que se fundó este "workshop" hasta el día de hoy, cada año, Performance Evaluation of Tracking and Surveillance (PETS) proporciona un nuevo conjunto de datos donde propone diferentes retos a la comunidad investigadora.

Concretamente, para nuestra evaluación hemos seleccionado la base de datos PETS2009 que contiene diferentes escenarios grabados desde múltiples cámaras. Esta base de datos fue especialmente creada para realizar estimaciones de densidad, tracking (seguimiento) y detección de eventos; todo ello para personas en multitudes [72]. Aunque originalmente no se tienen los videos etiquetados con las detecciones, en [73] se crea dicho fichero de etiquetación para una vista determinada de este dataset. Siendo más precisos, entre todas sus secuencias (vistas), hemos elegido la secuencia 2 con sus 3 niveles de complejidad. Esta complejidad es dada por el número de personas simultáneas en la imagen, las oclusiones que ocurren y las acciones que realizan las personas. En la Ilustración 5.1 vemos una imagen ejemplo de cada uno de los niveles de dicha secuencia y en la Tabla 5.1 podemos informarnos sobre datos más técnicos de estos "datasets" seleccionados.

Ilustración 5.1: Frames de ejemplo para PETS2009-S2

	PETS2009-S2L1	PETS2009-S2L2	PETS209-S2L3
Número de fotogramas	795	436	240
Tamaño (pixels)	768 x 576		
Número máximo de personas simultaneas en un frame	8	35	42
Complejidad	Baja	Media	Alta

Tabla 5.1: Información detallada para PETS2009-S2

5.1.2. TUD

Para poder evaluar con una base de datos, de igual importancia que la anterior, pero que no está orientada a multitudes usamos TUD (Technische Universität Darmstadt) [64]. Concretamente, usamos las secuencias de "Crossing" y "Campus", las cuales tiene una complejidad media-baja y el desplazamiento es casi completamente en horizontal. En estas secuencias, aunque si disponemos del etiquetado hecho por los autores, no se incluyen las detecciones de personas ocluidas más de un 50%. Por este motivo, usamos el desarrollado en [73] que sí contempla todas las detecciones. A modo gráfico, en la Ilustración 5.2 vemos un fotograma de ejemplo de las secuencias de este dataset, también tenemos la Tabla 5.2 para poder ver información más detallada.

Ilustración 5.2: Frames de ejemplo para TUD

	TUD-Campus	TUD-Crossing
Número de fotogramas	201	71
Tamaño (pixels)	640 x 480	
Número máximo de personas simultaneas en un frame	11	7
Complejidad	Media-baja	Media-baja

Tabla 5.2: Información detallada para TUD

5.1.3. MPII-2Person

En [74] usan MPII-2Person (Max Planck Institut Informatik) como un dataset de entrenamiento. Los autores usan esta base de datos para entrenar un modelo que aprende a detectar parejas, nosotros lo usaremos para evaluar nuestro algoritmo con los conjuntos de datos de diferentes grados de oclusión. Con la base de datos se incluye varios ficheros donde están etiquetadas todas las detecciones de los videos, separados por los niveles de oclusión que tenemos.

El desplazamiento de las personas es en horizontal y siempre hay dos personas en todos los fotogramas de la muestra. El tamaño de los "frames" puede ir variando, dado que el dataset está formado por 4 escenarios diferentes con resolución (tamaño) de imagen distintos (incluso en un mismo escenario puede haberse capturado la imagen con tamaños diferentes). En la Ilustración 5.3 vemos una imagen de cada conjunto de datos donde se ve la oclusión entre las parejas de personas. De forma adicional en la Tabla 5.3, aparte de información básica, tenemos el porcentaje aproximado de oclusión que se produce en cada nivel del dataset.

Ilustración 5.3: Frames de ejemplo para MII-2Person

	MPII-2Person									
	L1	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**	**L8**	**L9**	**L10**
Número de fotogramas	149	176	78	79	74	71	78	78	50	12
Personas simultaneas en un frame	2									
Porcentaje de oclusión	< 5	5-15	15-25	25-35	35-45	45-55	55-65	65-75	75-85	> 85

Tabla 5.3: Información detallada para MPII-2Person

5.2. Métrica de evaluación

5.2.1. Introducción

Para evaluar los algoritmos usaremos una comparación entre los resultados obtenidos y los resultados de una detección perfecta [75]. Esta detección perfecta, es un archivo donde están etiquetadas de forma manual las cajas (blobs) que envuelven a las personas de cada uno de los "frames". A este documento lo llamamos "ground truth" (GT) y viene dado por los conjunto elegido para hacer las pruebas. El formato del fichero, tanto del GT, como de nuestras detecciones tiene el siguiente formato (sección 3.5):

- Frame al que pertenecen.
- Posición X de la esquina superior izquierda.
- Posición Y de la esquina superior izquierda.
- Posición X de la esquina inferior derecha.
- Posición Y de la esquina inferior derecha.
- Puntuación que indica la probabilidad de que lo detectado sea el "objeto" buscado. El GT puede no tenerla dado que la probabilidad es 1.

Para la comparación entre blobs se usa la Precisión y el Recall. Las fórmulas de las métricas base es la siguiente:

$$Precision = \frac{NumeroVerdaderosPositivos}{NumeroVerdaderosPositivos + NumeroFalsosPositivos}$$

$$Recall = \frac{NumeroVerdaderosPositivos}{NumeroVerdaderosPositivos + NumeroFalsosNegativos}$$

Donde:

- El número de verdaderos positivos viene dado por el número de blobs que coinciden con en nuestros resultado y el GT. Entendemos como coincidir las los pares de blobs que cumples las siguientes restricciones (Ilustración 4.7):
    - El área de cobertura y solapamiento es mayor al 50%.
    - La distancia relativa (distancia en la que el radio del blob del GT se normaliza a 1) entre los centros de los blobs es menor a 0.5.
- El número de falsos positivos indica los blobs que se encuentran en nuestro resultado, pero que no están en el GT.
- El número de falsos negativos indica los blobs que se encuentran en el GT, pero que no están en nuestros resultados.

Por último, mediante la variación del umbral obtenemos los diferentes puntos de la Curva Precisión-Recall (CPR), la cual usamos para obtener una gráfica donde visualmente podemos comparar diferentes algoritmos o configuraciones de un mismo algoritmo. La precisión media (Average Precision, AP) es utiliza para condensar el rendimiento general de un algoritmo en un único valor, el cual corresponde al área bajo la curva (Area Under Curve, AUC) Precision-Recall. Para aproximar correctamente esta métrica utilizaremos la aproximación descrita en [76].

5.2.2. HDGP vs DTDP en videos con diferentes grados de oclusión

En esta prueba veremos cómo mejora (si mejora) el algoritmo modificado (HDGP) frente al algoritmo base en un dataset específico para parejas con oclusión de una persona con otra (MPII-2Person, sección 5.1.3).

Primeramente, hemos evaluado el dataset con la configuración recomendada por el autor de HDGP:

- Detección de la segunda persona en ambos ejes.
- Pasos de 6 en la búsqueda de la SP con respecto a la MP.
- 3 configuraciones (12, 13 y 14).
- Todas las escalas

MPII-2PERSON	DTDP	HDGP	Mejora
MPII-2Person_OcclusionLevel_1	93,2%	89,3%	-4,23%
MPII-2Person_OcclusionLevel_2	83,6%	73,3%	-12,38%
MPII-2Person_OcclusionLevel_3	61,2%	56,3%	-8,13%
MPII-2Person_OcclusionLevel_4	53,4%	48,8%	-8,59%
MPII-2Person_OcclusionLevel_5	50,8%	48,7%	-4,13%
MPII-2Person_OcclusionLevel_6	53,7%	45,6%	-15,12%
MPII-2Person_OcclusionLevel_7	47,0%	41,4%	-11,76%
MPII-2Person_OcclusionLevel_8	49,4%	42,2%	-14,46%
MPII-2Person_OcclusionLevel_9	51,8%	41,0%	-20,91%

MPII-2Person_OcclusionLevel_10	49,3%	44,0%	-10,91%
MEDIA	**59,3%**	**53,0%**	**-11,06%**

Tabla 5.4: Resultados DTDP vs HDGP con la configuración por defecto en MPII-2Person

En la Tabla 5.4 vemos que los resultados son negativos, por tanto procedemos a ver el porqué de dichos resultados. Esto se debe a que los parámetros (sección 4.2) de HDGP están elegidos para escenarios con una alta densidad de personas (grupos) y no para parejas aisladas, como es este caso. Para solucionar esto hay que realizar diferentes ajustes a los parámetros del algoritmo, las modificaciones son las siguientes:

- Detección de la segunda persona solo en el eje X dado que en este conjunto de datos el movimiento solo es en horizontal.
- Pasos de 1 en la búsqueda de la SP con respecto a la MP.
- La configuración usada es solo la 11, dado que el desplazamiento es solamente en horizontal, por lo que no habrá oclusiones de los pies, piernas, etc, al menos, completamente. Esto se debe a que el dataset sólo incluye parejas aisladas, nunca grupos, por lo tanto la oclusión siempre será 1 a 1, no habiendo más posibles oclusiones.
- Eliminamos la primera octava de escala, que hace referencia a personas muy pequeñas, que en este conjunto de datos no se da nunca.

Todo esto, justifica aún más, el autoajuste que proponemos (sección 4.2.3) haciendo el algoritmo HDGP lo más general posible. Por ejemplo, y simplemente usando el autoajuste de configuraciones (el de escalas tiene las limitaciones del cambio de tamaños de la imagen, diferentes tipos de escenario en un mismo conjunto de datos y "frames" de mismos escenarios no consecutivos por la clasificación por grado de oclusión) y si el tamaño del dataset fuera mucho mayor (como sería en casos más reales) el algoritmo sería capaz de adaptarse al escenario de forma automática, de tal forma que el sistema iría eliminando aquellas configuraciones menos útiles y terminaría usando, en este caso concreto, sólo la configuración 11. Por tanto, aunque el algoritmo no está diseñado para este escenario concreto con esta mejora, el autoajuste, se podrían mejorar los resultados tanto del algoritmo HDGP por defecto como del DTDP. En este trabajo nos centramos en la escala y las configuraciones tan solo, pero se podrían ir autoconfigurando de forma similar otros parámetros, como los modificados para este caso (paso y desplazamiento).

En el Gráfico A y B de la Ilustración 5.4 vemos como tanto en el nivel de oclusión 1 como en el 10 el algoritmo mejora los resultados, sobre todo cuando hablamos de "recall". La precisión se ve algo perjudicada por los exhaustividad del algoritmo que a la vez que introduce verdaderas detecciones que antes no se contemplaban, también añade falsos positivos que no estaban. Para ver con más claridad los resultados, tenemos la Tabla 5.5 donde está el AUC de ambos algoritmos para cada uno de los niveles de oclusión de la base de datos. A modo adicional, y para justificar aún más, si cabe, el autoajuste, en la Tabla 5.6 vemos el HDGP con la configuración inicial y el con la configuración establecida manualmente, que se podría conseguir (en un diferente conjunto de datos) con nuestro algoritmo, como veremos en la sección siguiente.

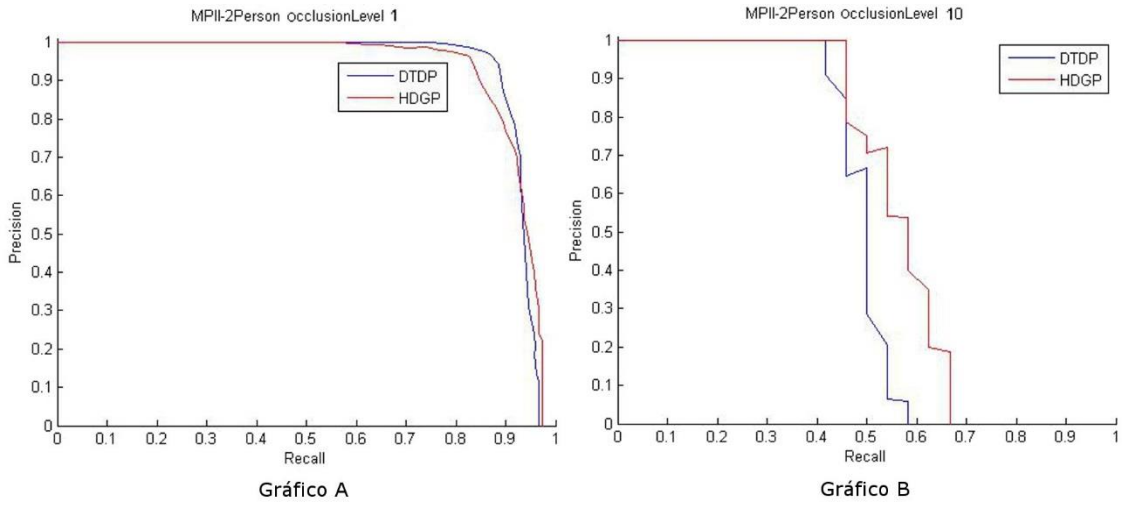

Ilustración 5.4: Gráficos DTDP vs HDGP con configuración personalizada en MPII-2Persons nivel 1 y 10 de oclusión

MPII-2PERSON	DTDP	HDGP	Mejora
MPII-2Person_OcclusionLevel_1	93,2%	92,7%	-0,57%
MPII-2Person_OcclusionLevel_2	83,6%	81,4%	-2,67%
MPII-2Person_OcclusionLevel_3	61,2%	65,5%	6,88%
MPII-2Person_OcclusionLevel_4	53,4%	59,4%	11,29%
MPII-2Person_OcclusionLevel_5	50,8%	55,4%	9,06%
MPII-2Person_OcclusionLevel_6	53,7%	55,3%	2,91%
MPII-2Person_OcclusionLevel_7	47,0%	50,4%	7,36%
MPII-2Person_OcclusionLevel_8	49,4%	51,9%	5,18%
MPII-2Person_OcclusionLevel_9	51,8%	49,3%	-4,93%
MPII-2Person_OcclusionLevel_10	49,3%	56,6%	14,78%
MEDIA	**59,3%**	**61,8%**	**4,93%**

Tabla 5.5: Resultados DTDP vs HDGP con configuración personalizada en MPII-2Person

MPII-2PERSON	HDGP def	HDGP mod	Mejora
MPII-2Person_OcclusionLevel_1	89,3%	92,7%	3,81%
MPII-2Person_OcclusionLevel_2	73,3%	81,4%	11,05%
MPII-2Person_OcclusionLevel_3	56,3%	65,5%	16,34%
MPII-2Person_OcclusionLevel_4	48,8%	59,4%	21,72%
MPII-2Person_OcclusionLevel_5	48,7%	55,4%	13,76%
MPII-2Person_OcclusionLevel_6	45,6%	55,3%	21,27%
MPII-2Person_OcclusionLevel_7	41,4%	50,4%	21,74%
MPII-2Person_OcclusionLevel_8	42,2%	51,9%	22,99%
MPII-2Person_OcclusionLevel_9	41,0%	49,3%	20,24%
MPII-2Person_OcclusionLevel_10	44,0%	56,6%	28,64%
MEDIA	**53,0%**	**61,8%**	**16,60%**

Tabla 5.6: Resultados HDGP con la configuración por defecto vs HDGP con configuración personalizada en MPII-2Person

5.2.3. Self-tuning HDGP vs HDGP vs DTDP

Autoajuste de la escala

En esta prueba veremos los resultados del autoajuste de escalas en los "datasets" de PETS (sección 5.1.1) y TUD (sección 5.1.2).

Una prueba importante de esta autoconfiguración, es la elección de umbral adecuado, por ello veremos cómo afecta usar 3 umbrales diferentes (0.7, 0.8 y 0.9) a los

resultados que obtenemos. Cabe mencionar que se han usado solo las configuraciones 12, 13 y 14. Esto se debe a que es perfectamente extrapolable con el uso de todas las configuraciones y de esta manera el algoritmo funciona más rápido, clave fundamental para realizar las pruebas.

Para el caso particular de "campus" (TUD), podemos observar en la Ilustración 5.5 vemos la gráfica resultante con umbral de 0.7, en la Ilustración 5.6 la gráfica con un umbral de 0.8 y en la Ilustración 5.7 la gráfica con un umbral de 0.9. Vemos que un umbral más bajo la precisión aumenta, pero se reduce el recall. Esto se debe a que eliminamos escalas que no son nada probables (aumento de la precisión), pero en alguna ocasión son necesarias (disminución del recall). Cuanto más aumentamos el umbral más permisivos somos, acercándonos por tanto más a la curva del HDGP por defecto.

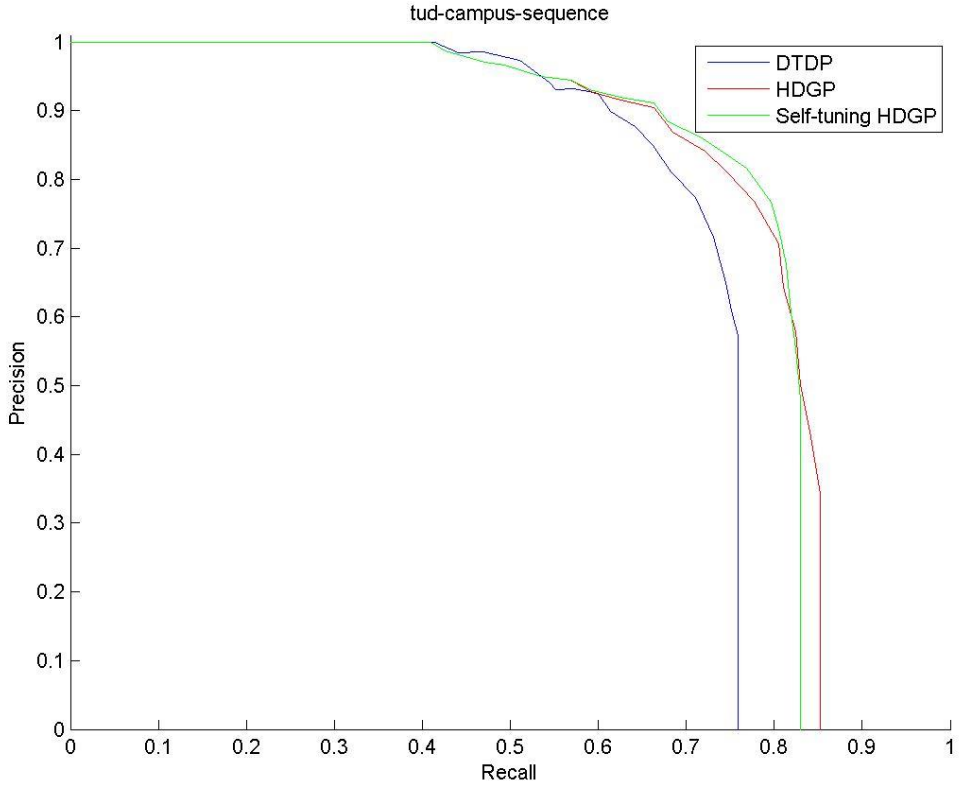

Ilustración 5.5: Gráfica de DTDP vs HDGP vs autoajuste de escalas con umbral 0.7 en la secuencia "campus" de TUD

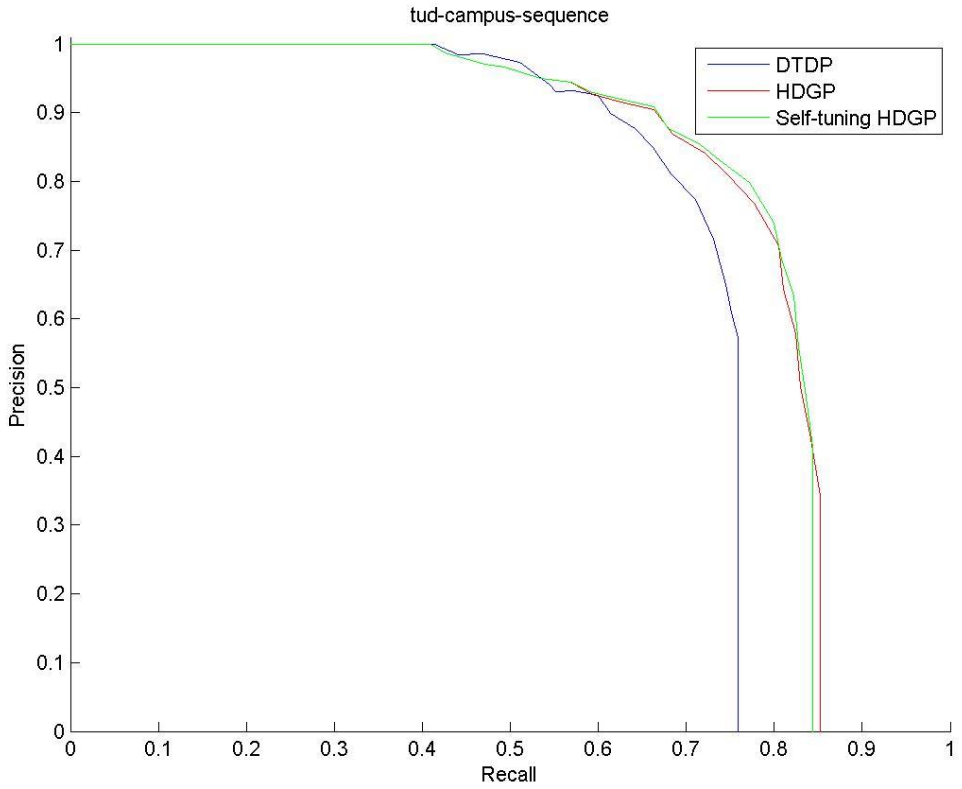

Ilustración 5.6: Gráfica de DTDP vs HDGP vs autoajuste de escalas con umbral 0.8 en la secuencia "campus" de TUD

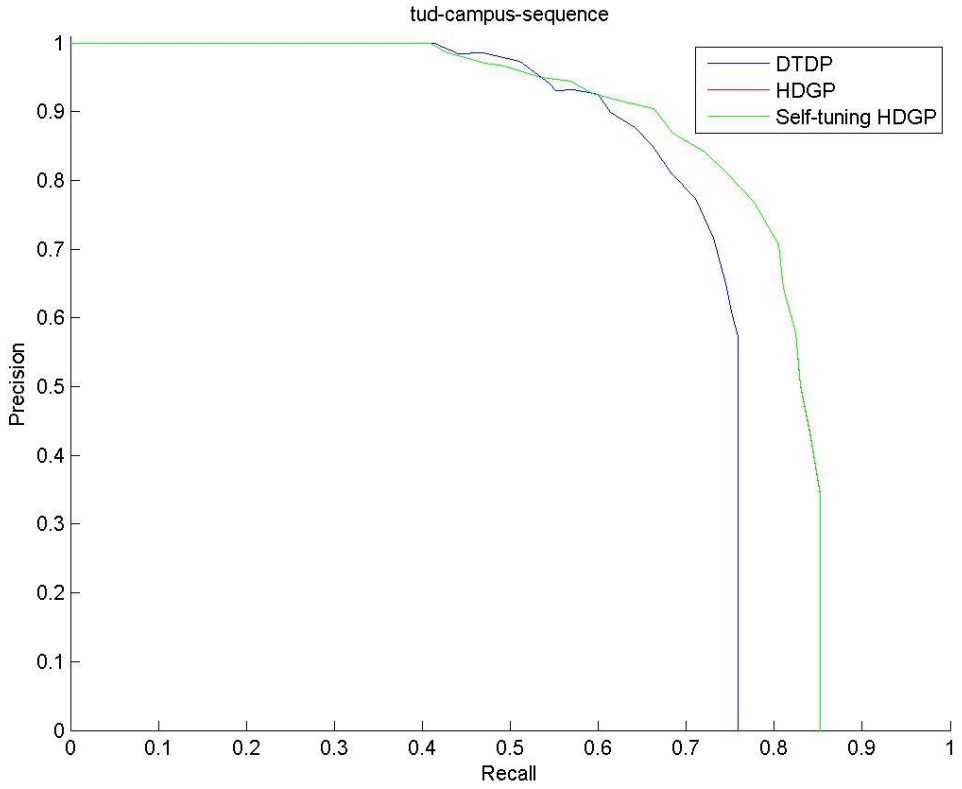

Ilustración 5.7: Gráfica de DTDP vs HDGP vs autoajuste de escalas con umbral 0.9 en la secuencia "campus" de TUD

En la Tabla 5.7, Tabla 5.8 y Tabla 5.9 vemos todos los resultados obtenidos para los diferentes conjuntos de datos y los 3 umbrales seleccionados.

Videos	DTDP	HDGP	Self-tuning HDGP	Mejora DTDP	Mejora HDGP
tud-campus-sequence	72,1%	79,1%	78,6%	9,03%	-0,71%
tud-crossing-sequence	85,4%	85,5%	82,6%	-3,24%	-3,39%
PETS2009-S2L3	24,4%	73,8%	69,9%	186,85%	-5,23%
PETS2009-S2L2	30,7%	80,9%	80,3%	161,34%	-0,72%
PETS2009-S2L1	55,9%	94,9%	94,9%	69,95%	0,05%
MEDIA	53,7%	82,8%	81,3%	84,78%	-2,00%

Tabla 5.7: Resultados de DTDP vs HDGP vs autoajuste de escalas con umbral 0.7

Videos	DTDP	HDGP	Self-tuning HDGP	Mejora DTDP	Mejora HDGP
tud-campus-sequence	72,1%	79,1%	79,1%	9,78%	-0,02%
tud-crossing-sequence	85,4%	85,5%	84,4%	-1,16%	-1,31%
PETS2009-S2L3	24,4%	73,8%	70,5%	189,36%	-4,40%
PETS2009-S2L2	30,7%	80,9%	80,4%	161,53%	-0,65%
PETS2009-S2L1	55,9%	94,9%	95,0%	69,95%	0,05%
MEDIA	53,7%	82,8%	81,9%	85,89%	-1,27%

Tabla 5.8: Resultados de DTDP vs HDGP vs autoajuste de escalas con umbral 0.8

Videos	DTDP	HDGP	Self-tuning HDGP	Mejora DTDP	Mejora HDGP
tud-campus-sequence	72,1%	79,1%	79,1%	9,81%	0,00%
tud-crossing-sequence	85,4%	85,5%	85,5%	0,12%	-0,03%
PETS2009-S2L3	24,4%	73,8%	70,9%	190,95%	-3,88%
PETS2009-S2L2	30,7%	80,9%	80,4%	161,37%	-0,71%
PETS2009-S2L1	55,9%	94,9%	94,9%	69,92%	0,03%
MEDIA	53,7%	82,8%	82,2%	86,43%	-0,92%

Tabla 5.9: Resultados de DTDP vs HDGP vs autoajuste de escalas con umbral 0.9

Como podemos ver en la Ilustración 5.8: Probabilidad de que una escala contenga una detección verdaderaIlustración 5.8 con un umbral de 0.9, se siguen eliminando muchísimas escalas, por lo que el algoritmo funcionaria mucho más rápido, aunque no se mejoren los resultados. Podemos ver el resultado final del autoajuste, una vez se ha sumado las probabilidades del *numFrames* seleccionados y se ha aplicado la media a 3 para que escalas cercanas tenga valores parecidos. Se por tanto que tras la normalización de la escala 6 a la 11 se escogerán seguro.

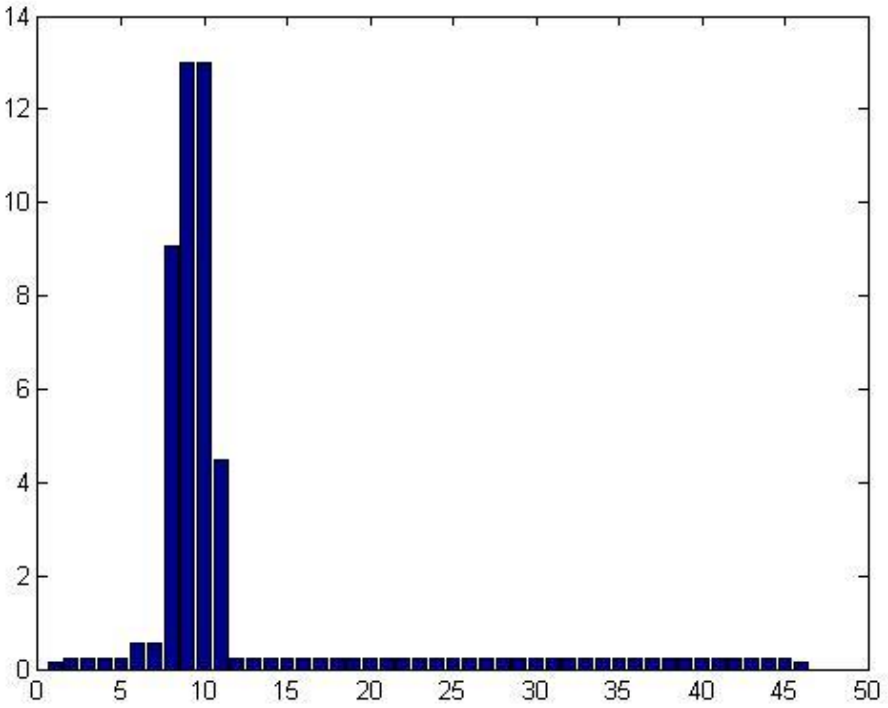

Ilustración 5.8: Probabilidad de que una escala contenga una detección verdadera

Autoajuste de uso de configuraciones

En esta prueba veremos los resultados del autoajuste de configuraciones en los "datasets" de PETS (sección 5.1.1) y TUD (sección 5.1.2).

En las ilustraciones siguientes podemos observar como usamos dos variantes de HDGP. En una tenemos todas las configuraciones, es decir: 11, 12, 13, 14 y 15 (HDGP T). Y en la otra tenemos tan solo las configuraciones 12, 13 y 14 (HDGP 3), las cuales son las que establece el autor como por defecto.

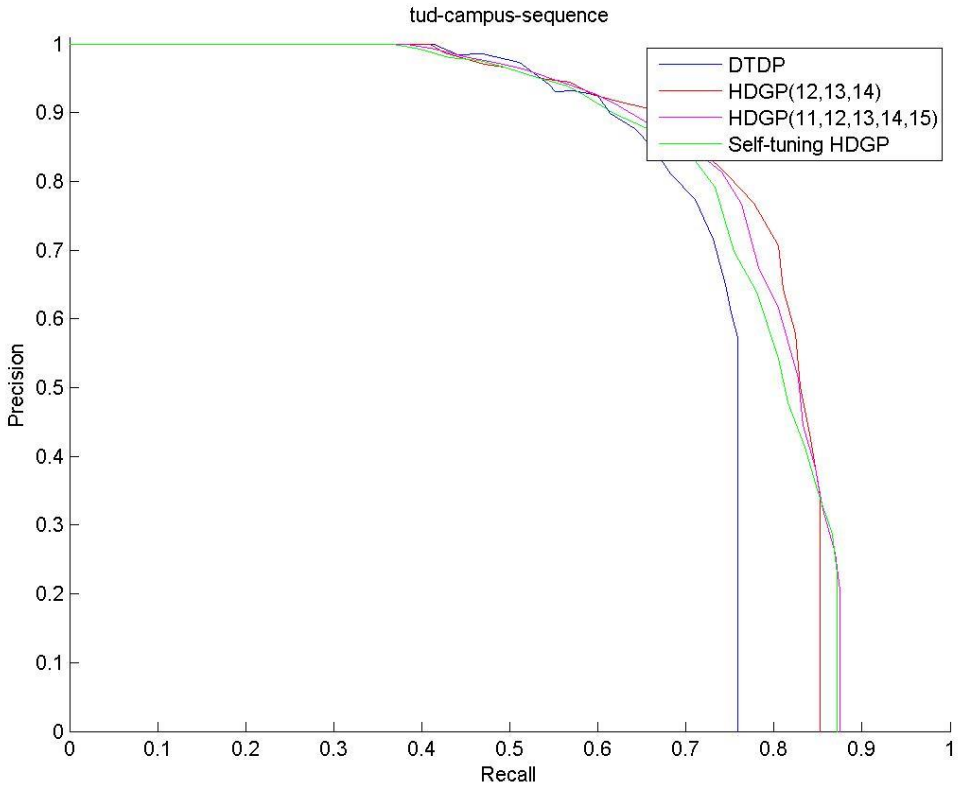

Ilustración 5.9: Gráfica de DTDP vs dos variantes de HDGP vs autoajuste de configuraciones en la secuencia "campus" de TUD

Podemos ver en la Ilustración 5.9 donde se ilustra la curva Precisión-Recall para la secuencia de TUD-campus como el Recall aumenta de manera notable con respecto a HDGP en su versión de 3 configuraciones. Esto se debe a que el uso de solo 3 configuraciones limita las detecciones que se pueden obtener. A pesar de esa mejora, la precisión se reduce, por lo que su valor de AUC no es mayor, tal y como podemos ver en la Tabla 5.10.

Videos	DTDP	HDGP 3	HDGP T	Self-tuning HDGP	Mejora DTDP	Mejora HDGP 3	Mejora HDGP T
tud-campus-sequence	72,1%	79,1%	79,2%	78,1%	8,45%	-1,23%	-1,30%
tud-crossing-sequence	85,4%	85,5%	84,8%	72,8%	-14,70%	-14,83%	-14,12%
PETS2009-S2L3	24,4%	73,8%	71,8%	71,1%	191,72%	-3,62%	-0,97%
PETS2009-S2L2	30,7%	80,9%	79,4%	79,3%	157,93%	-2,02%	-0,11%
PETS2009-S2L1	55,9%	94,9%	94,2%	94,0%	68,25%	-0,95%	-0,21%
MEDIA	53,7%	82,8%	81,9%	79,1%	82,33%	-4,53%	-3,34%

Tabla 5.10: Resultados de autoajuste de configuraciones vs DTDP y dos variantes de HDGP

Si observamos la Tabla 5.10 más detenidamente, veremos como en todos los casos mejoramos sustancialmente el valor original de DTDP (sección 4.1), salvo en la secuencia "crossing" de TUD. Esto puede ser a debido a que las configuraciones que se usan en cada momento cambian de manera muy repentina, no pudiendo nunca ajustarnos y fallando de manera abrumadora como vemos en la Ilustración 5.10.

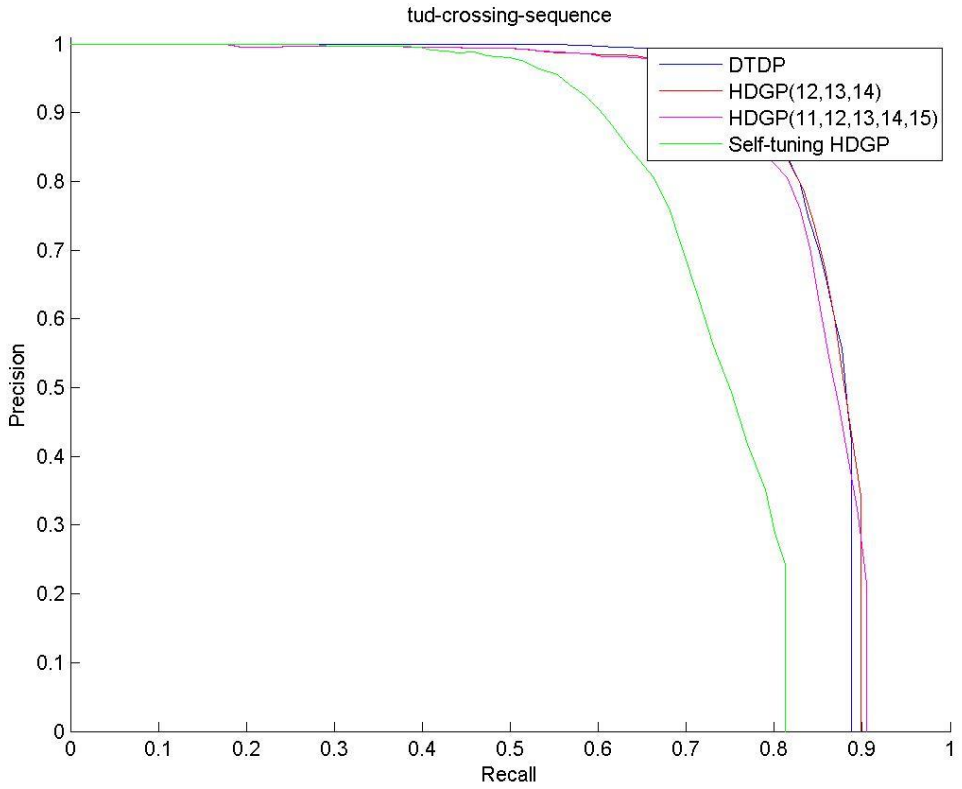

Ilustración 5.10: Gráfica de DTDP vs dos variantes de HDGP vs autoajuste de configuraciones en la secuencia "crossing" de TUD

5.2.4. Self-tuning HDGP vs algoritmos del estado del arte

En esta prueba veremos los resultados de la combinación del autoajuste de ambos parámetros explicados con respecto a ciertos algoritmos del Estado del Arte, en los "datasets" TUD (sección 5.1.2), dada su vinculación a no estar destinados a presencia de grupos.

Del Estado del Arte analizaremos "Implicit Shape Model" (ISM) [75] con el uso del modelo 2 y "Aggregated Channel Features" (ACF) [53] en la variante de Caltech e INRIA (modelos de entrenamiento). Además de DTDP y HDGP.

Primeramente, el umbral escogido para el autoajuste de la escala es de 0.9, dado que es el que mejores resultados aporta. Después, para la combinación de los autoajustes hemos escogido una opción poco restrictiva, es decir, con que uno de los dos parámetros acepte esa detección la detección se quedará. Esta opción, y tal como se ve en la Ilustración 5.11, permite que el "recall" empeore muy poco, y mejore algo la precisión, si es posible.

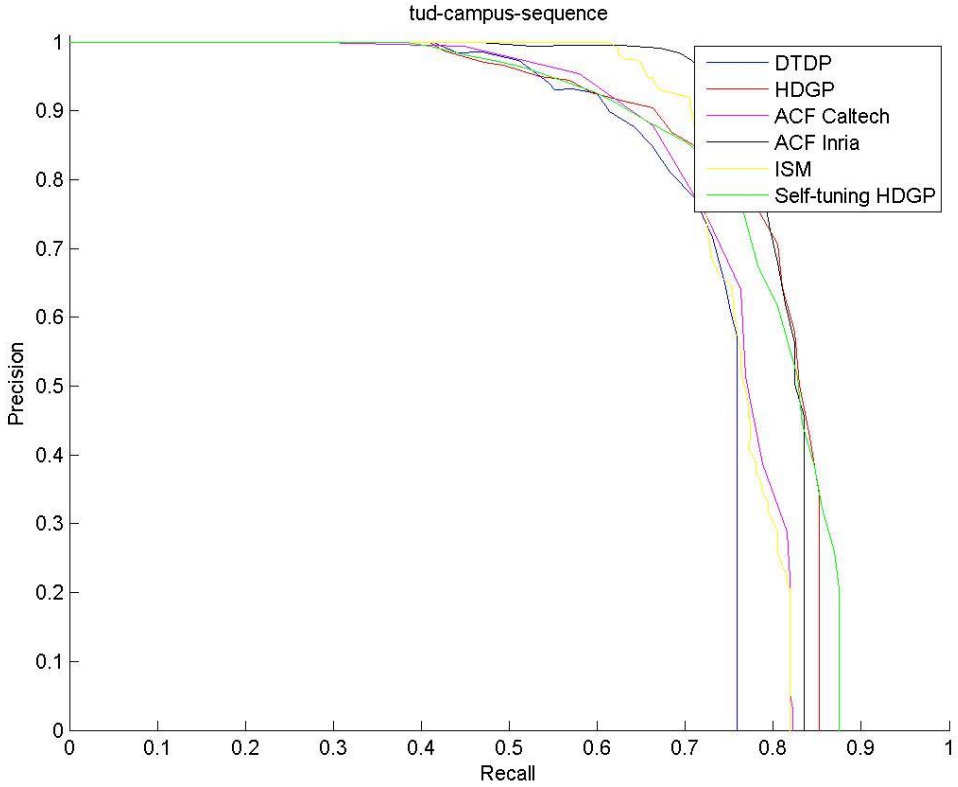

Ilustración 5.11: Gráfica de Self-tuning HDGP vs Estado del Arte en la secuencia "campus" de TUD

En la Tabla 5.11 vemos los resultados de los diferentes algoritmos para las secuencias elegidas, con la respectiva mejora de nuestro algoritmo.

Videos	tud-campus-sequence	tud-crossing-sequence	MEDIA
DTDP	72,1%	85,4%	78,7%
HDGP	79,1%	85,5%	82,3%
ACF Caltech	75,1%	83,4%	79,3%
ACF Inria	80,9%	88,0%	84,4%
ISM	76,1%	84,3%	80,2%
Self-tuning HDGP	79,2%	84,8%	82,0%
Mejora DTDP	9,88%	-0,67%	4,61%
Mejora HDGP	0,07%	-0,82%	-0,38%
Mejora ACF Calt.	5,40%	1,64%	3,52%
Mejora ACF Inria	-2,10%	-3,69%	-2,89%
Mejora ISM	3,97%	0,60%	2,29%

Tabla 5.11: Resultados de Self-tuning HDGP vs Estado del Arte

5.3. Conclusiones

Podemos concluir, por tanto, que gracias a la detección de parejas, mejoramos el "recall" de una manera algo notable, y gracias a la autoajustabilidad de los parámetros podemos mejoramos la precisión, además de generalizar el algoritmo.

Analizando un poco el contenido de las tablas en global, observamos que en escasos sitios mejoramos, pero nos quedamos muy cerca de resultados casi óptimos para la secuencia. Esto permite no empeorar lo resultados y evitar carga computacional.

De lo que no cabe ninguna duda es, que al menos con el autoajuste de las escalas, se consigue igualar los resultados usando muchas menos escalas de las que se usan en los algoritmos del principio (DTDP y HDGP). Esto por tanto, conlleva una menor carga computacional, mejorando los tiempos de respuesta.

Cabe mencionar que HDGP se realizó pensando en "datasets" como TUD y PETS, por lo que está muy optimizado para este tipo de secuencias. Deberíamos de probar en diferentes conjuntos de datos para poder sacar unas conclusiones más reales.

6. Conclusiones y trabajo futuro

6.1. Conclusiones

Después de las pruebas pertinentes podemos ver que los resultados satisfacen adecuadamente los objetivos marcados en la sección 2.2. Por lo tanto, debemos remarcar las tareas realizadas con éxito, que se enumeran a continuación:

- Se ha realizado un exhaustivo estudio del arte, reflejado claramente en la sección 3.
- Hemos conseguido realizar un algoritmo que mejora (mejora del "recall") las detecciones en presencia de grupo y no empeora cuando no hay dicha presencia.
- Mediante la autoajustabilidad del algoritmo se generaliza este a cualquier tipo de conjunto de datos y mejoramos la precisión de los resultados.
- Aunque las pruebas no demuestran unas mejoras notables, si evitamos el cálculo de muchas escalas y de algunas configuraciones, evitando así carga computacional innecesaria en muchos de los fotogramas.

6.2. Trabajo futuro

Hay muchos aspectos susceptibles de mejora en este software, dado que es tan solo un proyecto de investigación sobre un método concreto. Sin contar el trabajo futuro mucho más general que se comenta en la sección 3.7, algunas de las mejoras que podrían ser interesantes llevar a cabo en un futuro son:

- Mejorar la eficiencia y eficacia del detector de personas, estableciendo más parámetros que se autoajusten dependiendo de lo que se esté analizando.
- En este momento se analiza cada fotograma por separado, esto hace que el comportamiento no sea del todo real. Sería interesante hacer una unión con los fotogramas que ya han pasado para poder analizar posibles cambios de unos con otros. Esto también mejoraría el rendimiento del programa al no necesitar analizar de nuevo toda la imagen, si no solo los cambios que hayan sucedido.
- Realizar tracking para poder identificar personas y librarnos de muchos falsos positivos.
- Guardar la configuración y otros parámetros de interés en una base de datos, para poder llevar un mejor control de éstas y utilizarlas en diferentes tipos de análisis.
- Extender el algoritmo a una detección de tríos, cuartetos, etc. buscando cuando haya presencia de grupos el que más convenga.
- Evaluación en un conjunto mayor de "datasets".
- Implementación realista que evite realmente el cálculo de las configuraciones y escalas descartadas tras el autoajuste.
- Realizar el autoajuste de forma local y no solo a toda la imagen por igual.

7. Bibliografía

[1] P. Dollár, C. Wojek, B. Schiele y P. Perona, «Pedestrian Detection: An Evaluation of the State of the Art,» *IEEE Transactions On Pattern Analysis And Machine Intelligence* , vol. 34, nº 4, pp. 743-761, 2012.

[2] X. Zhang, Y. Yee-Hong, Z. Han, H. Wang y C. Gao, «Object Class Detection: A Survey,» *ACM Computing Surveys,* vol. 46, nº 1, pp. 10-53, 2013.

[3] A. Solichin, A. Harjoko y A. E. Putra, «A Survey of Pedestrian Detection in Video,» *International Journal of Advanced Computer Science and Applications,* vol. 5, nº 10, pp. 41-47, 2014.

[4] R. Benenson, M. Omran, J. Hosang y B. Schiele, «Ten Years of Pedestrian Detection, What Have We Learned?,» *Computer Vision - ECCV 2014 Workshops,* vol. 8926, pp. 613-627, 2015.

[5] B. Shin, J. H. Lee, H. Lee, E. Kim, J. Kim, S. Lee, Y.-s. Cho, S. Park y T. Lee, «Indoor 3D pedestrian tracking algorithm based on PDR using smarthphone,» *Control, Automation and Systems,* pp. 1442-1445, 2012.

[6] B. Wu, J. Liang, Q. Ye, Z. Han y J. Jiao, «Fast Pedestrian Detection with Laser and Image Data Fusion,» *International Conference on Image and Graphics,* pp. 605-608, 2011.

[7] D. Meissner, S. Reuter y K. Dietmayer, «Real-time detection and tracking of pedestrians at intersections using a network of laserscanners,» *Intelligent Vehicles Symposium,* pp. 630-635, 2012.

[8] V. Neagoe, A. Ciotec y A. Barar, «A Concurrent Neural Network Approach to Pedestrian Detection in Thermal Imagery,» *COMM,* pp. 133-136, 2012.

[9] X. Liu y K. Fujimura, «Pedestrian Detection Using Stereo Night Vision,» *IEEE Transactions of Vehicle Technology,* vol. 53, nº 6, pp. 1657-1665, 2004.

[10] Y. Xie, M. Pei, G. Yu, X. Song y Y. Jia, «Tracking pedestrians with incremental learned intensity and contour templates for PTZ camera visual surveillance,» *ICME,* pp. 1-6, 2011.

[11] R. Benenson, M. Mathias, R. Timofte y L. Van Gool, «Pedestrian detection at 100 frames per second,» *IEEE on Computer Vision and Pattern Recognition,* pp. 2903-2910, 2012.

[12] D. Weimer, S. Kohler, C. Hellert, K. Doll, U. Brunsmann y R. Krzikalla, «GPU architecture for stationary multisensor pedestrian detection at smart intersections,» *IEEE Intelligent Vehicles Symposium,* pp. 89-94, 2011.

[13] G. Liu y Y. Sun, «An In-Vehicle System for Pedestrian Detection,» *11th International Symposium on Distributed Computing and Applications to Business,*

Engineering & Science, pp. 328-331, 2012.

[14] L. Oliveira y U. Nunes, «Pedestrian detection based on LIDAR-driven sliding window and relational parts-based detection,» *IEEE on Intelligent Vehicles Symposium*, pp. 328-333, 2013.

[15] Z. Jin y B. Bhanu, «Integrating crowd simulation for pedestrian tracking in a multi-camera system,» *Distributed Smart Cameras*, pp. 1-6, 2012.

[16] P. Dollár, S. Belongie y P. Perona, «The Fastest Pedestrian Detector in the West,» *British Machine Vision Conference*, pp. 1-11, 2010.

[17] A. Vedaldi, V. Gulshan, M. Varma y A. Zisserman, «Multiple kernels for object detection,» *International Conference on Computer Vision*, pp. 606-613, 2009.

[18] S. Walk, N. Majer, K. Schindler y B. Schiele, «New features and insights for pedestrian detection.,» *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1030-1037, 2010.

[19] N. Dalal y B. Triggs, «Histograms of Oriented Gradients for Human Detection,» *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886-893, 2005.

[20] T. Yeh, J. J. Lee y T. Darrell, «Fast concurrent object localization and recognition,» *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 280-287, 2009.

[21] Z. Zhang, Y. Cao, D. Salvi, K. Oliver, J. Waggoner y S. Wang, «Free-shape subwindow search for,» *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1086-1093, 2010.

[22] M. A. Fischler y R. A. Elschlager, «The representation and matching of pictorial structures,» *IEEE Transactions on Computers*, vol. 22, nº 1, pp. 67-92, 1973.

[23] P. Felzenszwalb, D. McAllester y D. Ramanan, «A discriminatively trained, multiscale, deformable part model,» *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8, 2008.

[24] P. F. Felzenszwalb y D. P. Huttenlocher, «Pictorial structures for object recognition,» *International Journal of Computer Vision* , vol. 61, nº 1, pp. 55-79 , 2005.

[25] D. Crandall, P. Felzenszwalb y D. Huttenlocher, «Spatial priors for part-based recognition using,» *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 10-17, 2005.

[26] FergusR., P. Perona y A. Zisserman, «Weakly supervised scale-invariant learning of models for visual recognition,» *International Journal of Computer Vision*, vol. 71, nº 3, pp. 273-303, 2007.

[27] G. Bouchard y B. Triggs, «Hierarchical part-based visual object categorization,» *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 710-715, 2005.

[28] G. Carneiro y D. Lowe, «Sparse flexible models of local features,» *Computer Vision ECCV,* vol. 3953 , pp. 29-43, 2006.

[29] M. Andriluka, S. Roth y B. Schiele, «Pictorial structures revisited: People detection and articulated,» *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on,* pp. 1014-1021, 2009.

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester y D. Ramanan, «Object detection with discriminatively trained part-based models,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, nº 9, pp. 1627-1645, 2010.

[31] J. Wu, S. Yang y L. Zhang, «Pedestrian detection based on improved HOG feature and robust adaptive boosting algorithm,» *Image and Signal Processing (CISP),* pp. 1535-1539, 2011.

[32] K. Ahmad, Z. Saad, N. Abdullah, Z. Hussain y M. Mohd Noor, «Moving Vehicle Segmentation in a Dynamic Background using Self-adaptive Kalman Background Method,» *IEEE Colloquium on Signal Processing and its Applications* , pp. 439-442, 2011.

[33] S.-T. An, J.-J. Kim, J. W. Lee y J.-J. Lee, «Fast human detection using Gaussian Particle Swarm Optimization,» *Digital Ecosystems and Technologies,* pp. 143-146, 2011.

[34] L. Chen, W. Li, Z. Xu y L. Tang, «Pedestrian Detection Based on ISC in Infrared Images,» *Networking and Distributed Computing,* pp. 166-169, 2012.

[35] P. Borges, «Pedestrian Detection Based on Blob Motion Statistics,» *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 23, nº 2, pp. 224-235, 2013.

[36] M. Bui, V. Fremont, D. Boukerroui y P. Letort, «People Detection in Heavy Machines Applications,» *IEEE Conference on Cybernetics and Intelligent Systems,* pp. 18-23, 2013.

[37] C. Cosma, R. Brehar y S. Nedevschi, «Pedestrians detection using a cascade of LBP and HOG classifiers,» *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP),* pp. 69-75, 2013.

[38] F. De Smedt, K. Van Beeck, T. Tuytelaars y T. Goedeme, «Pedestrian Detection at Warp Speed: Exceeding 500 Detections per Second,» *IEEE Conference on Computer Vision and Pattern Recognition Workshops,* pp. 622-628, 2013.

[39] C. Guo, J. Meguro, Y. Kojima y T. Naito, «Detection of pedestrians in road context for intelligent vehicles and advanced driver assistance systems,» *IEEE Conference on Intelligent Transportation Systems,* pp. 1161-1166, 2013.

[40] A. Halidou y X. You, «Fast pedestrian detection using BWLSD for ROI,» *Wireless and Optical Communication Conference*, pp. 610-615, 2013.

[41] Z. Jiang, D. Huynh, W. Moran y S. Challa, «Combining Background Subtraction And Temporal Persistency In Pedestrian Detection From Static Videos,» *IEEE International Conference on Image Processing*, pp. 4141-4145, 2013.

[42] P. Karpagavalli y A. Ramprasad, «Human Detection and Segmentation in the Crowd Environment by Coimbining APD with HLBD approaches,» *Computer Vision, Pattern Recognition, Image Processing and Graphics*, pp. 1-4, 2013.

[43] J. Kim, J. Lee, C. Lee, E. Park, J. Kim, H. Kim, J. Lee y H. Jeong, «Optimal Feature Selection for Pedestrian Detection based on Logistic Regression Analysis,» *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 239-242, 2013.

[44] B. Li, Y. Li, B. Tian, F. Zhu, G. Xiong y K. Wang, «Part-based pedestrian detection using grammar model and ABM-HoG features,» *IEEE International Conference on Vehicular Electronics and Safety*, pp. 78-83, 2013.

[45] X. Li, X. Fang y Q. Lu, «On-road vehicle and pedestrian detection using improved codebook model,» *IEEE International Conference on Vehicular Electronics and Safety*, pp. 1-4, 2013.

[46] Z. Li y Y. Zhao, «Pedestrian detection in single frame by edgelet-LBP part detectors,» *IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 420-425, 2013.

[47] Y. Liu, J. Yao, R. Xie y S. Zhu, «Pedestrian Detection from Still Images Based on Multi-Feature Covariances,» *IEEE International Conference on Information and Automation*, pp. 614-619, 2013.

[48] S. Mehralian y M. Palhang, «Pedestrian detection using principal components analysis of gradient distribution,» *Iranian Conference on Machine Vision and Image Processing*, pp. 58-63, 2013.

[49] K. Min, H. Son, Y. Choe y Y.-G. Kim, «Real-time pedestrian detection based on A hierarchical two-stage Support Vector Machine,» *IEEE on Industrial Electronics and Applications*, pp. 114-119, 2013.

[50] H. Yoshida, D. Suzuo, D. Deguchi, I. Ide, H. Murase, T. Machida y Y. Kojima, «Pedestrian detection by scene dependent classifiers with generative learning,» *IEEE Intelligent Vehicles Symposium*, pp. 1-6, 2013.

[51] Y. Pang, K. Zhang, Y. Yuan y K. Wang, «Distributed Object Detection With Linear SVMs,» *IEEE Transactions on Cybernetics*, vol. 44, nº 11, pp. 2122-2133, 2014.

[52] J. Xu, D. Vazquez, A. Lopez, J. Marin y D. Ponsa, «Learning a Part-Based Pedestrian Detector in a Virtual World,» *IEEE Transactions on Intelligent*

Transportation Systems , vol. 15, nº 5, pp. 2121-2131, 2014.

[53] P. Dollar, R. Appel, S. Belongie y P. Perona, «Fast Feature Pyramids for Object Detection,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, nº 8, pp. 1532-1545, 2014.

[54] A. Ankit, I. R. Ahmad y H. Shin, «A cascade framework for unoccluded and occluded pedestrian detection,» *IEEE Students' Technology Symposium,* pp. 62-67, 2014.

[55] M. Pedersoli, J. Gonzalez, X. Hu y X. Roca, «Toward Real-Time Pedestrian Detection Based on a Deformable Template Model,» *IEEE Transactions on Intelligent Transportation Systems,* vol. 15, nº 1, pp. 355-364, 2014.

[56] J. M. Moguerza y A. Muñoz, «Support Vector Machines with Applications,» *Statist. Sci.,* vol. 21, nº 3, pp. 322-336, 2006.

[57] C. Cortes y V. Vapnik, «Support-Vector Networks,» *Machine Learning,* vol. 20, nº 3, pp. 273-297, 1995.

[58] S. Lazebnik, C. Schmid y J. Ponce, «Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,» *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 2, pp. 2169-2178, 2006.

[59] A. Bosch, Z. Andrew y X. Munoz, «Representing shape with a spatial pyramid kernel,» *ACM international conference on Image and video retrieval,* pp. 401-408, 2007.

[60] T. Watanabe, I. S. y K. Yokoi, «Co-occurrence histograms of oriented gradients for pedestrian,» *Advances in Image and Video Technology,* pp. 13-16, 2009.

[61] P. V. K. Borges, A. Tews y D. Haddon, «Pedestrian detection in industrial environments: Seeing around corners,» *IEEE International Conference on Intelligent Robots and Systems,* pp. 4231-4232, 2012.

[62] D. Yang, H. H. Gonzalez-Banos y L. J. Guibas, «Counting People in Crowds with a Real-Time Network of Simple Image Sensors,» *IEEE International Conference on Computer Vision,* vol. 1, pp. 122-129, 2003.

[63] A. Ess, B. Leibe y L. Van Gool, «Depth and Appearance for Mobile Scene Analysis,» *IEEE Computer Vision,* pp. 1-8, 2007.

[64] M. Andriluka, S. Roth y B. Schiele, «People-tracking-by-detection and people-detection-by-tracking,» *IEEE on Computer Vision and Pattern Recognition,* pp. 1-8, 2008.

[65] C. Wojek, S. Walk y B. Schiele, «Multi-cue onboard pedestrian detection,» *IEEE on Computer Vision and Pattern Recognition,* pp. 794-801, 2009.

[66] M. Enzweiler y D. Gavrila, «Monocular pedestrian detection: survey and experiments,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 2179-2195, 2009.

[67] D. Vazquez, A. Lopez, J. Marin, D. Ponsa y D. Geroimo, «Virtual and Real World Adaptation for Pedestrian Detection,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, nº 4, pp. 797-809, 2014.

[68] S.-C. Zhu y D. Mumford, «A stochastic grammar of images,» *Foundations and Trends® in Computer Graphics and Vision,* vol. 2, nº 4, pp. 259-362 , 2006 .

[69] S. Andrews, I. Tsochantaridis y T. Hofmann, «Support vector machines for,» *Advances in Neural Information Processing Systems,* pp. 561-568, 2002.

[70] P. F. Felzenszwalb y D. P. Huttenlocher, «Distance transforms of sampled functions,» *Theory of computing,* vol. 8, pp. 415-428, 2012.

[71] A. Garcia-Martin, R. Heras y T. Sikora, «A multi-configuration part-based person detector,» *nternational Conference on Signal Processing and Multimedia Applications,* 2014.

[72] A. Alahi, L. Jacques, Y. Boursier y P. Vandergheynst, «Sparsity-driven People Localization Algorithm: Evaluation in Crowded Scenes Environments,» *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,* pp. 1-8, 2009.

[73] A. Milan, S. Roth y K. Schindler, «Continuous energy minimization for multitarget tracking,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, nº 1, pp. 58-72, 2013.

[74] S. Tang, M. Andriluka y B. Schiele, «Detection and Tracking of Occluded People,» *International Journal of Computer Vision ,* vol. 110, nº 1, pp. 58-69 , 2014.

[75] B. Leibe, E. Seemann y B. Schiele, «Pedestrian detection in crowded scenes,» *IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 878-885, 2005.

[76] J. Davis y M. Goadrich, «The relationship between Precision-Recall and ROC curves,» *International Conference on Machine Learning,* pp. 233-240, 2006.