Detección de Personas en secuencias de vídeo en tiempo real

Autor: Javier Oliver Moll
Directores: Alberto Albiol Colomer
Guillermo Peris Fajarnés

En este trabajo se presenta un estudio sobre la viabilidad en el uso de una combinación de métodos para detección de personas en secuencias de vídeo en tiempo real. Se utilizan técnicas de boosting para una primera selección de candidatos en la escena, y máquinas de extracción y clasificación de características que actúan sobre estos candidatos proporcionando la predicción final. Los métodos de boosting proporcionan resultados rápidos aunque con un elevado número de falsas alarmas. Por otro lado, los métodos de extracción y clasificación de características presentan una elevada tasa de acierto a costa de un elevado coste computacional. La combinación de estos dos métodos logra unas prestaciones finales que combinan los aspectos positivos de ambas técnicas, y por lo tanto se establece como un sistema robusto a la vez que abierto a nuevas mejoras.

This document describes a combination of methods for vision-based pedestrian detection in real time. A preliminary selection of candidates in the scene is provided by boosting techniques. Then, a feature extraction and later classification using Vector Support Machines is carried out over these candidates, providing the final prediction. Boosting methods have been demonstrated to yield rapid but inaccurate predictions since they present a high rate of false alarms. On the other hand, feature-based methods present good performances in terms of accuracy, though a high computational cost is needed. Overall, the combination of both methods provides good results since the global system takes advantage of the strengths of each individual method. Moreover, the system is open to further improvements.

Valencia, 3 de Diciembre de 2007

Autor: Javier Oliver Moll, email: javioliver@gmail.com
Director: Alberto Albiol Colomer, email: alalbiol@dcom.upv.es
Director: Guillermo Peris Fajarnés, email: gperis@degi.upv.es

ÍNDICE

I. INTRODUCCIÓN

I.1. MARCO CONTEXTUAL

El presente trabajo está enmarcado en el Proyecto CASBLIP[1] (Cognitive Aid System for Blind and Partially Sight People) del 6º Proyecto Marco, cuyo objetivo consiste en el desarrollo de un prototipo capaz de interpretar y gestionar información procedente del mundo real para asistir a personas con ceguera total o parcial. El sistema reproducirá los elementos de la escena mediante mapas acústicos y mediante imágenes enriquecidas para aquellas personas con ceguera parcial.

El sistema está compuesto por diversos dispositivos y sensores, todos ellos embarcados en el usuario:

- un par de estéreo-cámaras que adquieren imágenes en color y permiten calcular mapas de profundidad de la escena
- un sensor láser CMOS que adquiere información de profundidad muy precisa en un plano acimutal
- un sensor de posicionamiento de la cabeza (HPS) que proporciona información de las coordenadas y ángulos de la cabeza
- un sensor inercial que proporciona información del movimiento y trayectoria del usuario
- un sistema GPS

Todos los dispositivos anteriores captan información del entorno que luego es procesada. En el área de procesado se contemplan dos campos claramente diferenciados, como son el subsistema de procesado de la imagen y el subsistema de sonificación. El presente trabajo queda enmarcado en el primer subsistema y proporcionará el conjunto de datos necesarios para generar los mapas acústicos que servirán de guiado para el usuario. Asimismo, los datos facilitados por el subsistema de procesado de imagen también van a poder ser utilizados para la generación de imágenes enriquecidas.

I.2 MOTIVACIÓN DEL TRABAJO

La generación del mapa acústico de la escena necesita de un análisis cognitivo previo capaz de filtrar aquellos elementos de la escena más relevantes para el usuario. Este proceso de análisis es desarrollado mediante técnicas de procesado de imagen. El procesado de imagen no sólo juega un papel de extracción de información, sino que además realiza labores de fusión de dispositivos, como son la fusión con la información del láser o del HPS, en un mismo paquete de datos.

I.3 OBJETIVOS

Inicialmente, la concepción de esta tarea era la creación de un modelado 3D capaz de integrar todos los dispositivos y sensores de entrada, y proporcionar a partir de este modelo un análisis cognitivo de la

[1] http://www.casblip.upv.es

escena que serviría para ser sonificado. Sin embargo, los primeros meses demostraron la ineficiencia del uso de este modelado 3D para funcionar en tiempo real, por lo que se decidió abordar el problema por separado. Se decide realizar tareas de detección de objetos directamente sobre la imagen 2D así como extraer otro tipo de información de interés para las personas ciegas como es la detección de objetos cercanos no identificados.

En este trabajo se estudia la viabilidad en la detección de personas en tiempo real. Se van a estudiar diferentes métodos y se va a tratar de combinarlos de una forma óptima y complementaria. En concreto, se va a estudiar la viabilidad de combinar técnicas de boosting con técnicas de extracción y clasificación de características. Se requiere que el sistema tenga una elevada tasa de acierto y sea un sistema robusto, lo cual supone un verdadero reto para su realización en tiempo real.

I.4 ORGANIZACIÓN DE LA MEMORIA

En el capítulo II del trabajo definimos el estado del arte en temas de detección de objetos en secuencias de vídeo. Elaboramos un listado de los métodos más interesantes, se analizan sus características y su viabilidad para cumplir con las especificaciones del sistema CASBLIP.

En el capítulo III relatamos la metodología seguida para realizar el trabajo. Se comentan las suposiciones de partida, basadas en trabajos previos, así como las posibilidades de mejora que ofrece cada método con el que vamos a trabajar. Se realiza una descripción de las mejoras introducidas en los métodos de extracción de características. Se realiza una definición de la metodología seguida para comprobar las prestaciones de cada método propuesto por separado, y posteriormente para el estudio del funcionamiento de ambas técnicas conjuntamente.

En el capítulo IV mostramos los resultados del trabajo. En primer lugar evaluaremos las mejoras introducidas en los métodos de extracción de características. En segundo lugar, realizaremos un entrenamiento de una máquina clasificadora de características. Y en último mostraremos los resultados de la combinación del método de un método de boosting con las técnicas mejoradas de extracción y clasificación de características citadas en estas líneas

II. DEFINICIÓN DEL PROBLEMA

II.1 ESTADO DEL ARTE

El problema de la detección de objetos en secuencias de vídeo es un tema de actualidad y muy estudiado por la comunidad científica. La adquisición de imágenes puede hacerse por medio de cámaras convencionales, hasta pasar por estéreo-cámaras, cámaras en el infrarrojo, estéreo-cámaras en el infrarrojo... Todas ellas existen en el mercado, y para cada una de ellas se puede plantear un tipo de detección diferente. Las cámaras en el infrarrojo son especialmente útiles para detección en entornos poco

iluminados y con condiciones meteorológicas adversas. Sin embargo, su elevado coste para su comercialización y su necesidad de recalibración periódica (normalmente cada año), especialmente cuando nos referimos a estéreo-cámaras, hacen difícil su inserción en el mercado. Por otro lado, las cámaras de estéreo-visión en la banda del visible no presentan tales ventajas frente a condiciones de iluminación adversas, pero suponen una ventaja en cuanto a precisión de los mapas de profundidad y a la menor calibración requerida.

En cuanto al procesado de imagen, existen diversas técnicas para el análisis de formas en imágenes. Algunos autores utilizan técnicas basadas en extracción de características y patrones de formas [1], [2], [3], técnicas de Boosting [4], [5], detección de objetos basada en formas mediante métodos Chamfer [6], correlación con patrones humanos probabilísticos [7], máquinas de soporte vectorial (SVM) [8], graph kernels [9], análisis de movimiento [10], análisis de componentes principales [11], clasificadores basados en redes neuronales [12] y redes neuronales convolucionales.

El caso particular de detección de personas supone un verdadero reto en la comunidad científica, ya que las personas pueden encontrarse en diferentes posiciones, con diferentes prendas, diferentes fondos y diferentes condiciones de iluminación. A todo ello, debemos añadir el hecho de que el dispositivo de captura va a introducir un temblor en las imágenes debido a que el sistema va embarcado sobre la persona, además de tener que ser un sistema capaz de funcionar en tiempo real y con elevada tasa de acierto.

El problema se abordará analizando las imágenes planas en color de tamaño 320x240 proporcionadas por el sistema de adquisición de imagen, en este caso un par de estéreo-cámaras. Este dispositivo de captura permite la generación de mapas de profundidad de cada imagen. Los mapas de profundidad (depth maps) proporcionan información de la profundidad de los objetos obtenida mediante el cálculo de la disparidad entre el par de imágenes (la disparidad se entiende como las diferencias que existen entre las dos imágenes en la dirección horizontal).

Este trabajo propone una combinación de métodos para llevar a cabo la detección de peatones. El hecho de que se trate de un sistema capaz de funcionar en tiempo real obliga al mismo a ser rápido y preciso en sus predicciones. Por ello, proponemos una combinación de métodos complementarios y con deficiencias individuales que al ser combinados producen un resultado robusto.

En primer lugar, utilizaremos técnicas de boosting para obtener los primeros candidatos. Sobre estos candidatos, extraeremos sus características que luego serán clasificadas utilizando máquinas de soporte vectorial (SVM). A continuación detallamos las propiedades de cada método.

II.2 TÉCNICAS DE BOOSTING

Las técnicas de boosting consisten en métodos que combinan un conjunto de clasificadores débiles que por sí solos proporcionan resultados poco robustos, pero que al combinarlos mejoran considerablemente sus prestaciones. Estos clasificadores débiles se distribuyen en grupos, y estos grupos se enlazan formando

una cascada, y actuando cada uno sobre las predicciones del anterior, constituyendo de esa forma el clasificador final (Paul Viola y Michael Jones [13]) (Figura 1). Cada uno de estos grupos se denomina una “etapa”, y el clasificador final estará constituido por varias etapas. Las primeras etapas filtran directamente aquellas zonas que son fácilmente identificables como “no personas”, siendo cada etapa sucesiva más restrictiva que la anterior. En función del número de etapas obtendremos unas prestaciones u otras, es decir, a mayor número de etapas filtraremos más las falsas alarmas pero tendremos perderemos también imágenes positivas que no interesaba eliminar.

Para el entrenamiento de cada clasificador nos basaremos en el algoritmo de Adaboost propuesto por Freud y Schapire en 1996 [14]. Dicho algoritmo se basa en la combinación de los clasificadores débiles.

El proceso que da para predecir los candidatos de una imagen queda explicado en la figura (Figura 1).

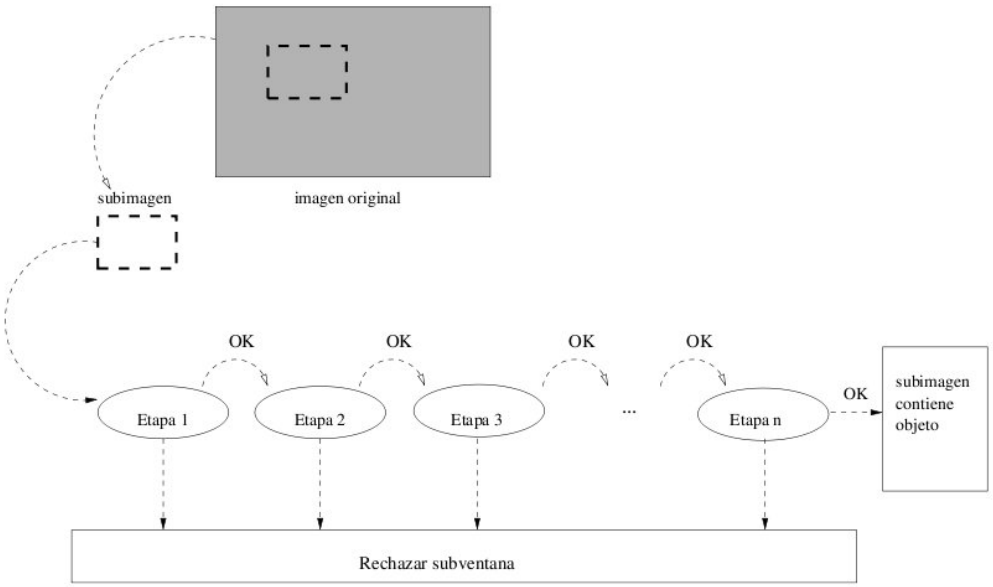

Fig 1: Proceso de obtención de candidatos con técnicas de boosting. Ref [15]

El proceso es como sigue: dada una imagen inicial, escaneamos dicha imagen con subventanas. Estas subventanas son evaluadas por las distintas etapas del clasificador, hasta que una de ellas la rechace. No obstante, la subventana bajo test deberá atravesar todos los clasificadores débiles en el interior de cada etapa. Un buen entrenamiento de los clasificadores es crucial para la máxima efectividad en términos de coste computacional y por tanto temporal.

Los clasificadores con un mayor número de reglas débiles a priori consiguen mejores resultados, pero a costa de un elevado coste computacional. Por ello, y dado el contexto de este trabajo, se hace necesario un estudio de la viabilidad de utilizar un número determinado de etapas para la clasificación. Este trabajo parte de resultados previos en detección de peatones mediante técnicas de boosting (A. Pardo, A. Albiol [15]), donde se establece una relación entre el número de imágenes detectadas con el número de etapas de la clasificación sobre la cual basaremos nuestro estudio.

II.3 TÉCNICAS DE EXTRACCIÓN DE CARACTERÍSTICAS

Como hemos comentado en los párrafos introductorios de este capítulo, la detección de personas en secuencias reales de vídeo resulta ser un reto en la actualidad debido a las tan variadas formas en las que se pueden encontrar las personas.

A parte de técnicas de boosting, existen otros métodos para detección de objetos basados en la extracción de las características de una imagen, que luego serán clasificadas y se obtendrá información de la existencia o no del objeto buscado. Algunos ejemplos de estos métodos son el detector de bordes Canny [16], métodos basados en la computación de la orientación y magnitud de los gradientes de la imágenes, métodos basados en matrices concurrentes [17], métodos basados en la diferencia en la intensidad de los histogramas, otros basados en los histogramas de los gradientes orientados (HOG) [1] [2], métodos basados en el número de unidades de textura (NTU) [18]...

A continuación detallamos brevemente el funcionamiento de cada uno de ellos.

-Detectores de bordes Canny: extraen los bordes de los objetos en las imágenes mediante la selección de aquellas regiones con altas derivadas espaciales. El hecho de tener en cuenta sólo los bordes de los elementos en la imagen reduce significativamente el tamaño de los datos a tratar, y filtra la información no útil de la imagen, conservando las formas, que es lo que proporciona la información relevante.

-Extracción de características en base al cálculo de la magnitud y orientación de los gradientes en la imagen en las dos dimensiones espaciales y para todos los píxeles: presentan obvios problemas de lentitud y elevado coste computacional, aunque cabe destacar sus buenas prestaciones.

-Métodos de matrices co-ocurrentes: se basan en la elaboración de una matriz con las frecuencias relativas P(i,j) con la información de cuales son los dos píxeles vecinos, separados una distancia 'd' y una orientación teta.

-Histogramas de las diferencias en intensidad: consiste en la elaboración de un vector de características con información de las frecuencias relativas de las diferencias en intensidad calculadas entre píxeles vecinos a lo largo de cuatro orientaciones en una imagen en escala de grises.

-Número de unidades de textura (NTU): consiste en la elaboración de un vector de características basado en la extracción de la información de textura de un píxel a partir de sus vecinos.

-Histogramas de los gradientes orientados (HOG): consiste en la división de la imagen en subbloques distribuidos a lo largo y ancho de la misma y con cierto solape entre ellos. Cada bloque se subdivide en subloques (o celdas) y sobre estos últimos se calcula la magnitud y orientación de los gradientes en cada píxel. Sobre cada uno de estos bloques se calcula el histograma de los gradientes orientados promediado por un peso gausiano, y luego se almacena en el vector de características de la imagen.

En este trabajo nos basaremos en los métodos HOG, dado su robustez frente a diferentes condiciones

de iluminación, pequeños cambios en el contorno de la imagen, diferentes fondos y escalas, y dado que este método presenta buenas prestaciones según resultados previos de otros autores [1] [2].

Las bases teóricas de los métodos HOG residen en trabajos previos tales como Histogramas de bordes orientados [Freeman and Roth 1995, Freeman et al], descriptores SIFT[2] [Lowe 1004] y reconocimiento de formas [Belongie et al. 2001], entre otros. Sin embargo, la diferencia añadida que presentan los métodos HOG consiste en que los gradientes no se calculan uniformemente sobre un mallado denso, sino que se divide la imagen en bloques, y a su vez cada bloque en diversos subbloques, y se calcula en cada uno de ellos los gradientes y el histograma.

El procedimiento para el cálculo de los descriptores de una imagen cualesquiera de tamaño 64x128 puede verse en la figura siguiente:

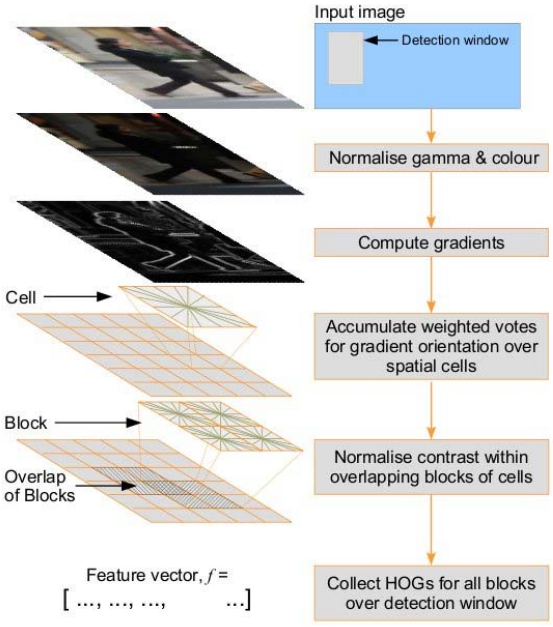

Fig 2. Proceso de extracción de características para una ventana de detección. Ref [2]

Dada una imagen en color, se transforma a escala de grises. A continuación se calculan los gradientes espaciales sobre toda la imagen. Posteriormente, dividimos la imagen en bloques, solapados cierta área. Cada bloque, a su vez, se divide en subloques (o celdas). Para cada celda calculamos los histogramas de los gradientes orientados. Finalmente se aplica una ventana gausiana sobre cada bloque, almacenándose dicha información en el vector de características de la imagen. El proceso se repite para todos los bloques de la imagen.

Definimos **NBP** (*Number of Spatial Bins)* como el tamaño que tiene cada bloque en el que dividimos la imagen medido en número de subbloques. Asimismo, cada subbloque tiene asignado un número de píxeles (Npx), con lo que el tamaño en píxeles de un bloque será: $T_{BLOQUE}\ (px) = NBP \cdot Npx \cdot NBP \cdot Npx$

Definimos **NBO** (*Number of Orientation Bins*) como el número de direcciones angulares que

[2] SIFT: Scale-Invariant Feature Transform

consideraremos para calcular el módulo del gradiente.

Por otro lado, **P_pt** es una matriz en la que hemos definido las coordenadas espaciales de los puntos donde se va a centrar cada bloque (de tamaño NBP*NBP celdas).

El gradiente sobre un píxel se calcula de la forma:

$$(D_X, D_Y) = \left[\frac{I_{X+1,Y} - I_{X-1,Y}}{2}, \frac{I_{X,Y+1} - I_{X,Y-1}}{2} \right] \quad (1)$$

La obtención del descriptor sobre un píxel[3] se obtiene mediante:

$$(módulo\ ,\ fase) = \left[\sqrt{D_X{}^2 + D_Y{}^2}, a\tan\left(\frac{D_X}{D_Y}\right) \right] \quad (2)$$

Con todo lo anterior, definimos el *número de dimensiones* del descriptor como: $N_{DIM} = NBP \cdot NBP \cdot NBO \cdot N_{PTOS}$. Este valor constituirá el número de dimensiones del espacio creado por la máquina clasificadora SVM.

Podemos concluir en que el descriptor HOG es realmente es un histograma en ponderado por una ventana gausiana que contiene la información de módulo y orientación de los gradientes de la imagen agrupados por bloques.

Para la implementación del algoritmo HOG utilizamos el software facilitado por Andrea Vedaldi donde se incluye una versión del algoritmo SIFT programada en c y adaptada para ser utilizada en LINUX y desde MATLAB utilizando las opciones de pre-compilación MEX[4].

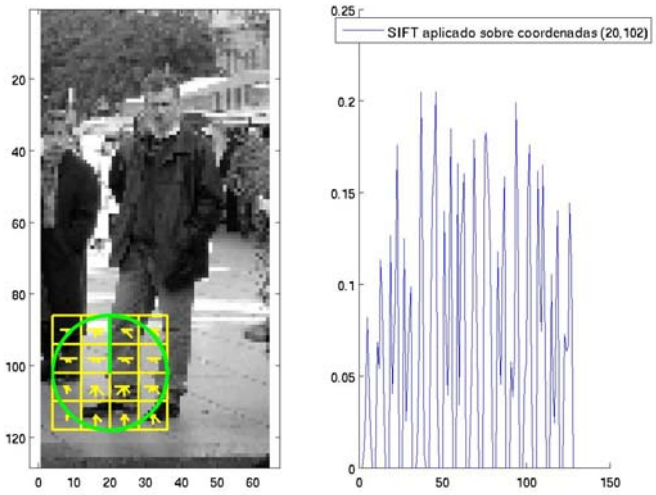

Fig 3. Ejemplo de funcionamiento del algoritmo con NBO=8, NBP=4, solape = ½, Npx = 8, sobre el punto (x,y) = (20,102)

En la figura de la izquierda podemos ver en amarillo un bloque de tamaño 4x4 celdas. En el interior del bloque encontramos un total de 16 celdas, en cuyo interior está representado el histograma de los

[3] S*in normalizar respecto a NBO ni aplicarse la ventana gausiana sobre el bloque total*
[4] http://vision.ucla.edu/~vedaldi/code/sift/sift.html

gradientes acumulados, representado por su módulo y fase en coordenadas polares. Existen 8 diferentes direcciones angulares para cada celda. En color verde, encontramos el efecto del enventanado gausiano sobre el histograma del bloque. En la figura de la derecha podemos ver el histograma del bloque, con 128 componentes. El vector de características final se formará por la concatenación de cada histograma perteneciente a cada bloque de la imagen.

II.4 MÁQUINAS DE SOPORTE VECTORIAL

Para la clasificación de los vectores de características creados mediante métodos HOG, utilizaremos máquinas de clasificación basadas en soporte vectorial, SVM.

Estas máquinas son capaces de clasificar muestras en dos posibles conjuntos. En este caso, los conjuntos son “personas” y “no personas”. Para ello, se necesita un entrenamiento previo de la máquina, facilitándole ejemplos de personas o “positivos” y ejemplos de no-personas o “negativos”. Con todos los ejemplos de entrenamiento, el algoritmo de clasificación SVM elabora una curva M-dimensional que divide ambos conjuntos, obteniendo de esta forma el kernel de la máquina. Las dimensiones del espacio dependen del número de componentes de cada vector a clasificar.

Existen diferentes parámetros a la hora de crear el kernel. Por ejemplo, podemos hablar de curvas trazadas con funciones lineales, polinómicas $k(x,x') = (s \cdot (x \cdot x') + c)^d$ (3), exponenciales $k(x,x') = \exp(-gamma \cdot \|x - x'\|^2)$ (4), de tangente hiperbólica $k(x,x') = \tanh(s \cdot (x \cdot x') + c)$ (5) y definir cada uno de sus parámetros, como el grado del polinomio, el valor de gamma, el valor de sigma “s”,... [20]

Por todo ello, la elaboración del kernel óptimo se hace tediosa. Además, hay que tener en cuenta que son muchas las aplicaciones en las que este tipo de máquinas toman parte, con lo que no hay a priori una inclinación por un método u otro.

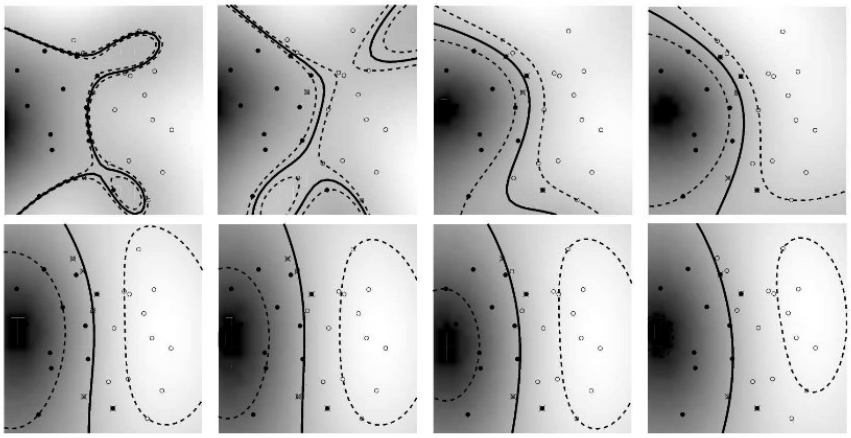

Fig4. Ejemplo de funcionamiento de un kernel gausiano (5) para varios valores del parámetro *gamma*, consistente en la separación de dos conjuntos muestrales $(x_1, y_1), ..., (x_m, y_m) \in \chi \times \{\pm 1\}$ [19]

La Figura 4 muestra un ejemplo de funcionamiento de un kernel gausiano. Los valores de este parámetro van de *gamma*=0.1 (arriba izquierda) a *gamma*=0.8 (abajo derecha). Cuanto más alto sea el

valor de gamma, más puntos permitimos que se encuentren en la zona situada entre ambos conjuntos.

Una vez trazada la línea que divide ambos espacios M-dimensionales, cuando queramos clasificar una nueva muestra no identificada deberemos introducirla en la máquina y representarla como un punto en este espacio. La distancia euclídea de la muestra a la curva trazada en la etapa de entrenamiento de la máquina resultará ser la medida que nos da la predicción sobre la muestra. El signo de esta magnitud nos informa que la muestra pertenece a un conjunto u otro (en este caso, “personas” o “no personas”). Un signo positivo nos indicará que la muestra clasificada pertenece al primer grupo. Análogamente, un signo negativo nos indicará que la muestra pertenece al segundo grupo. Así mismo, el módulo de la misma nos indica la probabilidad de que la muestra esté en un conjunto u otro. Así por ejemplo, valores de 3.2 indicarán que la muestra pertenece al espacio de “personas” con mayor probabilidad que si la predicción hubiera sido de 0.2. En el segundo caso, la máquina no predice con tanta certeza la pertenencia de la muestra a un conjunto u otro.

Una propiedad que hacen a este tipo de máquinas muy atractiva es la capacidad que tienen las mismas para aprender y la alta precisión en la clasificación del conjunto muestral de test entre los dos posibles conjuntos.

En los últimos años, SVM ha sido muy utilizado para diversas aplicaciones y por diversos investigadores en el campo del reconocimiento de formas, lo cual justifica la elección de este método y garantiza su buen funcionamiento.

En este trabajo hemos utilizamos un software comercial de SVM descargado de la url [http://www.cs.cornell.edu/people/tj/svm_light/] en su versión para LINUX, desarrollado por Thorsten Joachims.

En resumen, la combinación de los dos métodos (boosting y extracción y clasificación de características) aprovechará las buenas prestaciones de cada uno y corregirá sus carencias de una forma eficiente, totalmente compatible y bien integrada.

Las técnicas de boosting relacionadas con detección de personas muestran una gran velocidad de procesado y una baja probabilidad de pérdida de candidatos, aunque predice muchas falsas alarmas. Resultados previos [15] demuestran la poca fiabilidad de estos métodos para predicción de personas. Por ello, en este trabajo proponemos la combinación de las técnicas de boosting con la extracción de características y su posterior clasificación utilizando máquinas de soporte vectorial. Esto es, SVM actuará sobre los candidatos propuestos por Adaboost realizando una extracción de características a cada uno de los candidatos propuestos que luego serán clasificadas utilizando máquinas SVM.

Como hemos comentado, este tipo de máquinas clasificadoras produce muy buenos resultados en la predicción, a costa de un elevado coste computacional. En este estudio discutiremos la viabilidad de utilizar esta combinación de métodos para poder hacer frente al problema con elevada exactitud y elevada velocidad de procesado.

III. METODOLOGIA

III.1 CONSIDERACIONES PREVIAS

-*Técnica de Boosting*: Partimos de resultados previos sobre entrenamientos de sistemas de boosting basados en el método de Adaboost. Disponemos de un conjunto de ocho clasificadores finales diferentes, formados por 5, 10, 20, 25, 30, 35, 40 y 45 etapas. Estos conjuntos han sido previamente entrenados con ejemplos positivos (3365 imágenes) y negativos (200 imágenes) obtenidos de una base de datos estándar [15]. El tamaño de estas imágenes para el entrenamiento fue de 16x32.

-*Método de extracción de características (HOG):*

Partimos del algoritmo que calcula el SIFT programado en C y disponible para MATALB. Como vimos en el capítulo anterior, el método HOG parte del algoritmo SIFT, con lo que será necesario hacer leves modificaciones que comentaremos en los párrafos siguientes.

En este trabajo nos basamos en los resultados obtenidos en trabajos previos, en concreto en los presentados por Navneet Dalal [1] [2]. Principalmente, partiremos de la gráfica siguiente para justificar el uso de un tamaño entre bloques en una imagen para la extracción de las características. La gráfica presenta el ratio de pérdidas de personas en función del tamaño de bloque y del tamaño de subbloque o celda. Parece claro que el óptimo se encuentra para un tamaño de bloque 3x3 celdas y un tamaño de subbloque de 6x6 px.

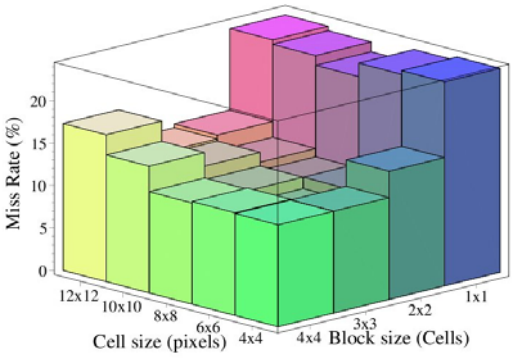

Fig 5. Efecto del tamaño de bloque y del número de píxeles por celda en el Miss Rate, para un solape entre bloques de la mitad de bloque.

Asimismo, encontramos en otros trabajos del mismo autor que los mejores resultados de solape entre bloques se obtienen para un solape alrededor de ½ y ¾. Por ello, son dos opciones que tendremos en cuenta en este estudio.

No obstante, refiriéndonos de nuevo a los tamaños de bloque y subbloque óptimos presentados por Navneet, consideraremos en este trabajo otras combinaciones subóptimas (zona central de la gráfica), ya que el ratio de pérdidas (Miss Rate (%)) varía en algunas zonas del centro tan solo en un 2 o 3% respecto del óptimo propuesto el autor. Por ello, haremos un estudio considerando las diversas combinaciones entre bloques pertenecientes a la zona centra de la gráfica, y analizaremos el detrimento que supone en las prestaciones del sistema final.

-Máquina de Soporte Vectorial (SVM):

Como comentamos en el capítulo anterior, trabajaremos sobre los kernels más comunes: polinómicos y gausianos. Sobre cada uno de ellos, modificaremos los parámetros clave, como es el parámetro 'd' para los polinómicos y el parámetro 'gamma' para los gausianos. Para el resto de parámetros utilizamos los valores por defecto.

-Bases de datos:

Para poder entrenar la máquina clasificadora, necesitamos introducirle ejemplos positivos y negativos para que la máquina aprenda. Tanto los ejemplos positivos como negativos deben proporcionarse en forma de matriz, donde cada fila corresponde con una imagen diferente y cada columna con una característica típica de dicha imagen. Las dos respectivas matrices con ejemplos positivos y negativos son utilizadas para entrenar la máquina.

La selección de ejemplos positivos y negativos se ha hecho incluyendo bases de datos estándares en la detección de personas y muy comúnmente utilizadas. Trabajaremos con las bases de datos de MIT Pedestrian [21] e INRIA [22]. La base de datos MIT Pedestrian representa una base de datos estándar en la detección de personas y contiene un total de 924 imágenes positivas de tamaño 64x128. Dichas personas se encuentran en determinadas posiciones, con diferentes iluminaciones y diferentes fondos. Sin embargo, esta variabilidad es muy limitada ya que las situaciones reales están llenas de una mayor variabilidad tanto en la posición de las personas como en el fondo. Por ello incluimos la base de datos de INRIA, donde originalmente se presenta un conjunto de datos positivos y negativos, organizadas en dos carpetas: test y train. Cada una de estas carpetas contiene a su vez dos más, una con ejemplos positivos y otra con negativos. En esta base de datos encontramos un total de 1192 imágenes de tamaño 64x128 positivas para entrenamiento, 570 imágenes de tamaño 64x128 positivas para test, 1218 imágenes negativas de tamaño 320x240 para entrenamiento y 453 imágenes negativas de tamaño 320x240 para test.

Fig 7. Ejemplos positivos de la base de datos “Pedestrian”

Fig. 8. Ejemplos positivos de la base de datos “INRIA”

Fig. 9. Ejemplos negativos de la base de datos “INRIA”

En total, vamos a juntar todas las imágenes en dos grupos: positivas y negativas. Respecto a los ejemplos negativos, decidimos generar un programa que seleccione 10 subimágenes de tamaño 64x128 a partir de cada una de 320x240. Dichas subimágenes son generadas en posiciones totalmente aleatorias y con tamaños aleatorios, siendo luego normalizados a 64x128. Con esto aumentamos el conjunto de imágenes negativas. Además, decidimos calcular las reflexiones sobre un eje vertical de todas las imágenes positivas y negativas, y añadir estas muestras a la base de datos

En definitiva obtenemos un total de: 4991 imágenes positivas y 16630 imágenes negativas, sin distinción alguna referida a su uso para entrenamiento o test. Más adelante se verá cómo estructurar este conjunto de ejemplos para proceder a la experimentación y veremos como ampliamos el conjunto muestral mediante el aprendizaje de la máquina clasificadora.

Hasta ahora hemos estado hablando de bases de datos con imágenes de tamaño 64x128, que contenían únicamente ejemplos positivos o negativos. Estas imágenes serán utilizadas para entrenar la máquina clasificadora SVM y testear sus propiedades en una primera instancia. Sin embargo, cuando dispongamos del subsistema final formado por la combinación de los métodos Adaboost, HOG y SVM (sección 7.5 y 7.6), necesitaremos otro conjunto de imágenes de testeo. Estas imágenes deben ser secuencias de video reales, que contengan personas y fondo simultáneamente sobre la misma imagen, para poder realizar el análisis de prestaciones del método final. Las imágenes deberán ser de tamaño 320x240, por las propiedades de las estéreo-cámaras utilizadas en el sistema global CASBLIP.

Se decidió utilizar para este estudio una selección de imágenes creada por nosotros y una recopilación de imágenes extraídas de trabajos previos [15], cuyos tamaños fueron normalizados a 320x240 por las especificaciones anteriormente citadas. Estos conjuntos de imágenes contienen personas de diversos tamaños, en movimiento, con fondos variantes, diferentes condiciones de iluminación, ligeras rotaciones de la cámara sobre un eje perpendicular al plano de la imagen, así como muchos otros factores. Por cada frame disponemos información del bounding box de cada persona que aparece en la imagen, dado por su centro, su anchura y altura, todo ello dado en formato texto. Con ello, podemos comparar trabajos previos sobre métodos de Boosting del autor anteriormente citado con el presente trabajo, lo cual nos dará una visión de las mejoras introducidas a la vez que podremos testear las deficiencias del algoritmo y por tanto tomar las medidas oportunas.

Hay que recordar que el sistema definitivo va embarcado sobre la persona, con lo cual es necesario tomar secuencias de vídeo acordes con lo que se espera capturar en el sistema final.

Fig 10. Ejemplo de imágenes propias para testear el subsistema final

Fig 11. Ejemplo de imágenes de otra base de datos [15]

III.2 MEJORAS EN EL ALGORITMO HOG

Por las exigencias del sistema final donde se va a incluir el método resultante de este trabajo, debemos cumplir con las especificaciones sobre el coste temporal de procesado. Por ello, debemos simplificar al máximo todos los métodos y variables que tomen parte en este subsistema.

Para lograr una máxima optimización de los recursos de memoria y velocidad de procesado, proponemos varias modificaciones sobre el algoritmo SIFT.

En primer lugar, debido a la arbitrariedad que existe en la distribución de la luminosidad en las imágenes, hemos considerado que en el cálculo de los gradientes podríamos prescindir de la información acerca del sentido del gradiente. Esto es así porque en detección de peatones lo que nos interesa saber no es el sentido sino su dirección, debido a la arbitrariedad en las prendas que puedan llevar los peatones y en los diferentes objetos e iluminaciones que puedan darse en el fondo de la imagen. Nos interesa conocer la existencia de un cambio de luminosidad y su distribución, pero no en qué dirección.

En segundo lugar, la extracción de características se hace sobre unos determinados puntos prefijados. Lo óptimo sería aplicar algoritmos que extraigan las características más relevantes de la imagen, dejando a un lado aquellas zonas homogéneas y con poca información. Sin embargo, por problemas de elevado coste temporal de estos métodos decidimos fijar *a priori* los puntos en la imagen donde centrar cada bloque.

Haremos un estudio de la posición óptima de estos puntos, partiendo de ideas previas basadas en el trabajo de Navneet y la propia intuición sobre el problema. Nos basaremos en solapes entre bloques de valor ½, ¾, y ¼.

Asimismo, también analizaremos la viabilidad de eliminar los puntos más cercanos al borde de la imagen, ya que a parte de no aportar información relevante, suponen un mayor coste temporal en su procesado.

Para comprobar las mejoras introducidas es necesario utilizar una máquina clasificadora. Como hasta este punto todavía no hemos encontrado el mejor kernel, decidimos realizar este apartado utilizando un kernel paramétrico con d=3, tras realizar unos estudios preliminares con varios kernels y pocas muestras de entrenamiento. Nótese que no es importante trabajar con un kernel subóptimo, ya que todos los estudios para mejorar el algoritmo HOG estarán referidos al mismo kernel, con lo que podrán apreciarse las diferencias.

En la sección de resultados, el efecto de las contribuciones será analizado por separado. En primer lugar, en el apartado IV.1.1 analizaremos el efecto de descartar los puntos cercanos a los bordes de la imagen, estableciendo algunos parámetros a priori para poder realizar el test. En segundo lugar, en el apartado IV.1.2 analizaremos el efecto de solape entre bloques para valores de solape de ¼, ½ y ¾. Finalmente, en el apartado IV.1.3 analizaremos el efecto del tamaño de bloque, basándonos en la Figura 5. Los valores que probaremos serán de NBP = 2 y NBP = 4, para una SIGMA = 8píxeles (SIGMA corresponde con el tamaño de una celda). Por limitaciones impuestas por el algoritmo SIFT sobre el que partimos, solo es posible probar valores pares de NBP.

III.3: ENTRENAMIENTO DE LA MÁQUINA CLASIFICADORA SVM

Para poder entrenar esta máquina se requiere un conjunto de ejemplos positivos y otro negativo. Con ello, como se ha explicado previamente, la máquina sitúa estos puntos en un espacio N-dimensional y traza una curva que separa ambos conjuntos. En función de la precisión con la que la curva separa ambos conjuntos, y de la calidad de las muestras de los mismos (es decir, si las muestras tienen características relevantes), obtendremos unos resultados u otros. Para evaluar las prestaciones de la máquina, necesitamos un conjunto muestral de test, sobre el que calcular unos parámetros de comparación, que se detallan en el punto III.3.2.

III.3.1 Adecuación de la base de datos por escenarios

A partir del conjunto muestral obtenido de las bases de datos INRIA [22] y MIT Pedestrian [21], donde agrupamos los ejemplos positivos por un lado y los negativos por otro, vamos a definir una metodología para proceder al entrenamiento y testeo de los diferentes kernels.

Para ello, decidimos utilizar un 90% de las muestras de cada conjunto para realizar el entrenamiento, y reservar un 10% para el testeo de la máquina. Además, para aumentar la aleatoriedad del conjunto muestral y poder ofrecer unos resultados más generalizados, realizamos diez escenarios, donde en cada uno de ellos seleccionamos un 10% de muestras distinto para testear. Con ello eliminamos dependencias de cualquier tipo que pudiera ocasionar la selección de un único conjunto de test.

Para cada escenario se entrenará la máquina y posteriormente se testeará introduciéndole los ejemplos de test, los cuales conocemos a priori. La máquina clasificará estos ejemplos y predecirá unas etiquetas para cada muestra. De estos resultados se obtendrán unos parámetros de comparación que evaluarán las

prestaciones de la máquina. Finalmente, para cada parámetro se hará la media de cada escenario, obteniéndose al final unos valores promediados y por tanto menos dependientes del conjunto seleccionado para el test.

III.3.2: Parámetros de comparación

Atendiendo a otros trabajos, los parámetros más comúnmente utilizados para analizar las prestaciones de cada kernel son los siguientes: Accuracy, Precision, Recall, FP, FPPW, MR, y se definen como sigue:

Accuracy (*Exactitud en la predicción*): $Accuracy = \frac{FP + FN}{Ntest}$ (6)

Precision (*Relacionado con el grado de falsas predicciones*): $precission\ (\%) = 100 - \left(\frac{FP}{Ntestpos} \right) \cdot 100$ (7)

Recall (*Relacionado con el grado de pérdida de candidatos*): $recall\ (\%) = 100 - MR \cdot 100$ (8)

FP (*Falsas Positivas*). Indica el número de imágenes clasificadas erróneamente como positivas.

FN (*Falsas Negativas*). Indica el número de imágenes no detectadas.

FPPW (*Falsas Positivas por Ventana*). $FPPW = \frac{FP}{Ntestneg}$ (9)

MR (*Miss Rate*). Indica el grado de pérdida de candidatos. $MR = \frac{FN}{Ntestpos}$ (10)

Ntestpos: Número de ejemplos de test positivos

Ntestneg: Número de ejemplos de test negativos

Presentaremos los resultados en forma de tablas, gráficas y resultados visuales donde el análisis subjetivo también juega un papel importante.

III.3.3: Análisis de kernels paramétricos

En este apartado se procede a la realización de diversas máquinas clasificadoras utilizando diferentes kernels paramétricos, $k(x, x') = (s \cdot (x \cdot x') + c)^d$, donde $d \in \{2,3,4,5,6,7,8,9,10\}$ y siguiendo el método propuesto en los puntos anteriores.

III.3.4: Análisis de kernels gausianos

El desarrollo de este apartado se llevará a término de idéntica forma al apartado anterior, pero experimentando con kernels gausianos, $k(x, x') = \exp(-gamma \cdot \|x - x'\|^2)$, donde $g \in \{0.05,0.1,0\ .125,0.15,\ 0.175,0.2,\ 0.25,0.3,0\ .5,0.75,1\ ,2,3,4,5,1\ 0\}$.

Hemos elegido valores comprendidos entre 0 y 1, y luego un orden de magnitud mayor, por tratarse de un kernel de ecuación exponencial.

III.3.5: Aprendizaje del kernel. Realimentación de muestras.

Una vez realizado el estudio de los diferentes kernels, seleccionaremos los que tengan mejores prestaciones. Sin embargo, llegados a este punto la máquina no puede considerarse definitiva todavía. Debemos notar que para que un resultado sea considerado como aceptable debe alcanzar valores de Precisión muy altos. No se concibe un sistema que tenga una alta probabilidad de falsa alarma, ya que ello podría inducir a la toma de decisiones erróneas y posiblemente peligrosas. Por ello, procederemos a testear la máquina con un elevado número de ejemplos negativos, con el fin de recoger aquellas predicciones erróneas o más dudosas para luego realimentarlas en la máquina, reentrenando la misma habiendo aumentado el conjunto de muestras negativas inicial con las seleccionadas en este punto.

Sobre estos kernels realimentados y sobre los kernels sin realimentar, realizaremos los estudios posteriores que tienen en cuenta la fusión de este método con el de Boosting visto en capítulos anteriores.

III.4 COMBINACIÓN DE TÉCNICAS DE BOOSTING CON EXTRACCIÓN Y CLASIFICACIÓN DE CARACTERÍSTICAS

En este estudio partiremos de los resultados previos citados en la sección III.1.1. Sin embargo, hay ciertos parámetros que todavía debemos establecer. En esta sección hacemos un estudio sobre el número de etapas óptimo de Adaboost para obtener la mejor combinación de estos tres métodos. Estableceremos otros parámetros de Adaboost como son el tamaño mínimo de persona buscado o el factor de escalado óptimo que debemos aplicar a la imagen de entrada.

Respecto a SVM, buscaremos el mejor kernel de entre los preseleccionados en los capítulos anteriores.

Para realizar este estudio nos basamos igualmente en las secuencias de video citadas en la sección III.1.4, y analizaremos de nuevo los parámetros de comparación.

Igualmente, el criterio subjetivo juega un papel muy importante en este punto.

La forma de operación de estos tres métodos en conjunto es como sigue:

En primer lugar, establecemos los parámetros de entrada, como son el número de etapas de Adaboost, el tamaño mínimo de persona que buscamos, el factor de agrupación de candidatos cercanos, el factor de escalado de la imagen de entrada, los puntos donde calcular las características, NBO, NBP...

A continuación aplicamos el método de Adaboost sobre la imagen y obtenemos el primer conjunto de candidatos. Sobre estos candidatos (todos ellos de 64x128) calculamos su descriptor (vector de características) para cada uno de ellos, y clasificamos estos vectores con SVM. El resultado de SVM será la predicción final acerca de las personas que hay en la imagen.

A su vez, nos guardamos en un fichero de texto todas las etiquetas generadas por el método de Adaboost y SVM, que nos servirá posteriormente para extraer resultados, comparando con los bounding boxes de asociados a cada frame de la secuencia de vídeo.

IV. RESULTADOS[5]

IV.1 ESTUDIO DE LAS MEJORAS INTRODUCIDAS EN EL ALGORITMO HOG

El objetivo principal de este estudio consiste en crear un sistema robusto capaz de conseguir unas predicciones lo más exactas posibles. Cuando hablamos de exactitud nos referimos principalmente a dos parámetros: a) el número de falsas alarmas que da el sistema y b) el número de personas que el sistema no detecta.

Conocemos en la literatura que Adaboost tiene un gran porcentaje de acierto. Sin embargo, proporciona demasiadas falsas alarmas. Por ello, cuando combinamos Adaboost con las técnicas de extracción y clasificación de características, cada conjunto estará optimizado en una parte. Sabemos que la limitación de Adaboost reside en que no puede mejorar la probabilidad de falsa alarma sin perder muchos candidatos. Sin embargo, puede encontrar con muy buena probabilidad a todos las personas de la escena, en función del número de etapas elegido. Por otro lado, conocemos que las técnicas de extracción y clasificación de características dan muy buenos resultados, a costa de un elevado coste computacional. Por todo ello, decidimos establecer el subsistema final de forma que ambos métodos se complementen. Esto es, elegiremos un número de etapas adecuado en Adaboost para que se detecten el mayor número de personas posibles en la escena, a pesar del elevado número de falsas alarmas. Posteriormente, las técnicas de extracción de características se encargarán de refinar estas predicciones. Habrá que llegar a una solución de compromiso entre ambas técnicas que tenga en cuenta el coste temporal de la predicción, puesto que el sistema está pensado para ser ejecutado en tiempo real.

Por ello, en algunos casos tomaremos decisiones que, lejos de ser las mejors en términos locales, iran en beneficio común para mejorar las prestaciones del subsistema completo.

IV.1.1 Efecto de descartar los puntos cercanos a los bordes de la imagen

Consideraremos para realizar este estudio un solape entre bloques de 0.5, valor elegido intuitivamente a partir de otros resultados anteriores. Notar que este valor no influye en los resultados del estudio, puesto que este parámetro es común en ambos casos. Asimismo, partimos del algoritmo SIFT modificado para que no tenga en cuenta el sentido del gradiente, sino únicamente la dirección.

Los parámetros establecidos inicialmente para llevar a cabo este estudio son:

HOG: NBP = 4, NBO = 4, SIGMA = 8, solape ½.

SVM: kernel polinómico con d=3, th=0.

Se ha realizado la clasificación para cada uno de los 10 escenarios previamente descritos, y

[5] Los resultados obtenidos en los siguientes apartados han sido obtenidos bajo el sistema operativo LINUX y trabajando con un cluster de diversos ordenadores. Las máquinas utilizadas han sido de tipo INTEL T2250@1.73GHz con 1GB de RAM.

posteriormente se han calculado los valores medios de los parámetros mostrados en la Tabla 1 de todos los escenarios.

Los resultados nos muestran cómo al disminuir el número de puntos donde calcular el descriptor HOG disminuimos considerablemente el número de dimensiones del vector de características. Esto repercute directamente en el espacio de memoria ocupado por el vector así como en el tiempo de procesado del vector final, ya que habrá menor número de puntos a clasificar en el espacio Ndim-dimensional.

	Considerando todos los ptos	Sin considerar los ptos de los bordes
Nptos - Ndim	50 - 3200	18 - 2304
Accuracy (%)	99.20	99.01
Precision (%) / Recall (%)	93.56 / 98.75	91.91 / 98.63

Tabla 1. Efecto de no extraer las características en los puntos de los bordes[6] de la imagen

Aunque los parámetros *Precision* y *Recall* disminuyen ligeramente, se puede asumir este descenso en las prestaciones debido al importante decremento del número de dimensiones. En los posteriores apartados veremos como mejorar este valor.

Desde este momento no consideraremos los puntos de los bordes para calcular los descriptores.

IV.1.2 Efecto del solape entre bloques

Los parámetros establecidos inicialmente para llevar a cabo este estudio son:

HOG: NBP = 4, NBO = 4, SIGMA = 8, no consideramos los puntos de los extremos de la imagen

SVM: kernel polinómico con d=3, th=0.

	Solape ¾	Solape ½	Solape ¼
Nptos - Ndim	64 - 4160	18 - 2304	8 - 512
Accuracy (%)	98.80	99.01	99.66
Precision (%) / Recall(%)	90.26 / 98.52	91.91 / 98.63	93.01 / 98.72
Tiempo de clasificación por muestra (s)	0.0219	0.0129	0.0060

Tabla 2. Efecto del solape entre bloques

Los resultados muestran cómo para solapes entre bloques menores se reduce considerablemente el número de características del descriptor. Esto es debido a que al estar más espaciados los bloques, es necesario un menor número de ellos para cubrir toda la imagen. Si analizamos el número de puntos vemos como pasamos de 64 a 8, lo cual supone pasar de calcular los descriptores en 64 puntos a 8.

Por otro lado, vemos que los parámetros de Accuracy, Precision y Recall mejoran considerablemente para solapes menores. En cuanto al tiempo necesario para la clasificación de una muestra, encontramos como tansolo son necesarios 0.006 segundos, mejorando mucho respecto al resto. Esto se hace evidente si analizamos el número de dimensiones de cada vector de características. Por ello, elegimos el solape

[6] *Para cada imagen positiva hay un cierto espacio entre el marco de la imagen y la persona. Este margen puede ser entorno a 10 px. Cuando nos referimos a "no considerar los puntos de los bordes", nos referimos a no extraer las características en estas zonas más extremas.*

óptimo entre bloques como ¼.

IV.1.3 Efecto del tamaño de bloque

Según los resultados publicados por otros autores [2], el tamaño óptimo de bloque es de 3x3. En este estudio, por limitaciones en los métodos sobre los que nos basamos, solo es posible calcular los descriptores sobre bloques de tamaño par. Por ello haremos un estudio para valores de NBP=2 y NBP=4, considerando el tamaño de celda o subbloque igual a 8 píxeles (SIGMA=8).

El resto de parámetros los fijamos a:

HOG: NBP = 4, NBO = 4, SIGMA = 8, no consideramos los puntos de los extremos de la imagen

SVM: kernel polinómico con d=3, th=0.

	NBP=2	NBP=4
Nptos - Ndim	36 - 576	8 - 512
Accuracy (%)	99.20	99.32
Precision (%) / Recall (%)	90.72 / 97.07	93.01 / 98.72
Tiempo de clasificación por muestra (s)	0.0058	0.0052

Tabla 3. Efecto del tamaño de bloque

Claramente obtenemos mejores valores para tamaños de bloque 4x4. Ello puede ser debido a que el número de dimensiones es ligeramente menor para el caso NBP=4. Además, puede deberse también a que la extracción de características sobre la imagen es más robusta en este caso.

IV.2 ESTUDIO DEL MEJOR KERNEL PARA LA MÁQUINA CLASIFICADORA SVM

En la sección anterior hemos comprobado las mejoras realizadas en el algoritmo HOG con una máquina de clasificación elaborada a priori, sin ser el caso óptimo. En este apartado vamos a definir cuál es la mejor máquina clasificadora para nuestro caso. Asumiremos durante este estudio que el umbral de decisión para determinar si una etiqueta es positiva o negativa es igual a 0 (th = 0).

En las tablas siguientes contrastamos los parámetros más significativos[7]. Los valores de Accuracy, Precision y Recall son los usados hasta ahora. El resto de parámetros los detallamos a continuación:

Number of Support Vector: Hace referencia al número de vectores necesitados para generar el kernel del clasificador.

Space in disk (MB): Representa el espacio en MB que ocupa en disco cada kernel.

Runtime per image (s): Representa el tiempo medio que la máquina necesita para clasificar una imagen de test.

[7] *El conjunto de imágenes de entrenamiento es el citado en el capítulo de Metodología. Sin embargo, por motivos que luego justificaremos, hemos aumentado el conjunto muestral de ejemplos negativos (Apéndice 2). Para ello, se ha generado previamente otro conjunto de imágenes negativas (~15000 muestras). Se ha partido de la máquina clasificadora usada en el ejercicio anterior, y se han clasificado estas nuevas muestras. Aquellas muestras mal clasificadas o etiquetadas con valores negativos cercanos a cero han sido incorporadas a la base de datos definitiva*

Pos. labels: (mean, deviation): Haremos un estudio de las etiquetas predichas por las diferentes máquinas. Estudiaremos los valores medios y desviaciones en la predicción sobre el conjunto de test de imágenes positivas.

Neg. labels: (mean, deviation): Igualmente para las muestras de test negativas

POLYNOMIAL KERNELS

	d=2	d=3	d=4	d=5	d=6	d=7	d=8	d=9	d=10
Accuracy(%)	99.57	99.57	99.58	99.57	99.58	99.57	99.56	99.55	99.53
Precision(%) / Recall(%)	99.32/ 93.77	99.61/ 93.57	**99.63/ 93.63**	99.64/ 93.53	**99.78/ 93.53**	**99.96/ 93.26**	100 / 92.92	100 / 92.79	100 / 92.62
Number of Support Vector	3842	4163	4506	4853	5202	5581	5973	6380	6821
Space in disk (MB)	28.6	31.2	33.4	35.9	38.5	41.2	44.6	47.5	50.3
Runtime* per image (s)	0.0123	0.0131	0.0141	0.0152	0.0164	**0.0177**	0.0188	0.0200	0.0217
Pos. labels: (mean, deviation)	2.34, 1.49	(2.14, 1.36)	(1.99, 1.24)	(1.86, 1.16)	(1.76, 1.10)	(1.67, 1.05)	(1.60, 1.02)	(1.54, 0.99)	(1.48, 0.97)
Neg. labels: (mean, deviation)	-4.00, 1.95	(-3.35, 1.29)	(-2.91, 0.94)	(-2.59, 0.74)	(-2.35, 0.62)	(-2.15, 0.54)	(-2.00, 0.50)	(-1.88, 0.47	(-1.77, 0.45)

Tabla 4. Estudio comparativo para un kernel polinómico variando el parámetro d

Como hemos comentado en la introducción a la sección de *Resultados*, nos interesa disponer de un valor de *Precision* elevado. En la tabla anterior se observa como para valores del parámetro 'd' tenemos un valor máximo de *Precision*. Sin embargo, observamos un mayor incremento en el coste temporal al compararlo con valores de 'd' ligeramnente inferiores. Además, para valores elevados de este parámetro obtenemos muy bajas probabilidades de *Recall*, lo cual no es bueno. Sin embargo, para valores de 'd' iguales a 6 o 7, continuamos teniendo valores muy altos de *Precision*, y nos aproximamos ligeramente al óptimo valor de *Recall*. Además, el coste temporal para estos kernels son más bajos.

La Figura 13 representa la distribución de la predicción sobre las muestras positivas y negativas. Son histogramas de las predicciones. En ellas se aprecian distribuciones muy similares entre los kernels. Para valores de predicción cercanos a cero, los histogramas conservan la misma forma, diferiendo para valores más alejados del cero, lo cual no nos interesa. Ello nos indica que todos los kernels tienen comportamientos muy similares en cuanto a los valores de predicción cercanos a cero.

Sin embargo, para valores de 'd' mayores observamos como el histograma se ensancha hacia valores negativos. Esto es porque la curva trazada está menos elaborada y se ajusta peor al conjunto muestral de entrenamiento.

Por todo lo anterior, encontramos el óptimo valor de 'd' para d=7, ya que existe un buen compromiso entre los valores de *Precision*, *Recall* y *Runtime*.

---

como más ejemplos negativos.

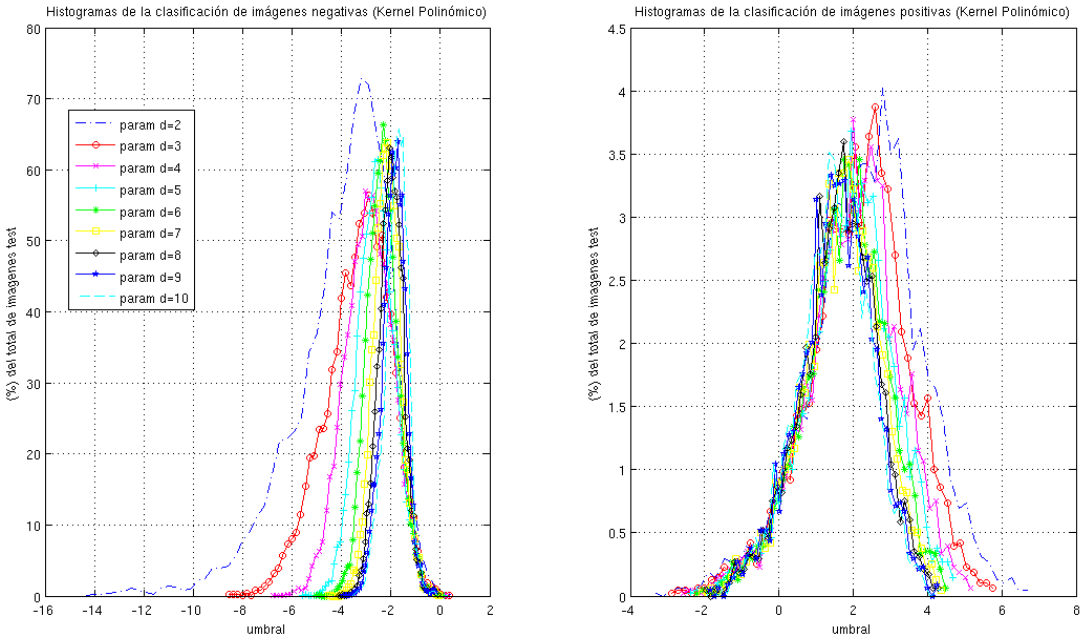

Fig 13: Distribución de la predicción sobre muestras negativas (izquierda) y positivas (derecha).

RADIAL BASIS KERNELS

	g = 0.05	g = 0.1	g = 0.125	g = 0.15	g= 0.175	g = 0.2	g = 0.25
Accuracy(%)	99.57	99.58	99.59	99.60	99.60	99.60	99.60
Precision(%) / Recall(%)	99.25/ 93.90	99.46/ 93.83	99.61/ 93.80	**99.78/ 93.80**	**99.82/ 93.73**	**99.93/ 93.70**	**100 / 93.63**
Number of Support Vector	3967	4214	4337	4457	4589	4717	4978
Space in disk (MB)	29.9	31.3	32	33	34	35	37
Runtime* per image (s)	0.0129	0.0140	0.0153	0.0158	**0.0111**	**0.0114**	0.0169
Pos. labels: (mean, deviation)	(2.25, 1.41)	(2.10, 1.31)	(2.04, 1.27)	(1.99, 1.24)	(1.94, 1.21)	(1.89, 1.18)	(1.81, 1.13)
Neg. labels: (mean, deviation)	(-3.76, 1.74)	(-3.34, 1.33)	(-3.17, 1.18)	(-3.02, 1.05)	(-2.89, 0.95)	(-2.77, 0.87)	(-2.57, 0.73)

	g = 0.3	g = 0.5	g = 0.75	g = 1	g = 2	g=3	g=4
Accuracy(%)	99.59	99.55	99.51	99.43	97.87	95.26	93.81
Precision(%) / Recall(%)	100 / 93.53	100 / 92.96	99.89/ 92.05	99.63/ 90.40	98.99/ 66.80	99.47/ 23.91	100 / 0.43
Number of Support Vector	5233	6371	8079	10563	25622	30817	31965
Space in disk (MB)	39	47	60	79	130	230	235
Runtime* per image (s)	0.0181	0.0216	0.0289	0.0378	0.0838	0.0814	0.0904
Pos. labels: (mean, deviation)	(1.74, 1.08)	(1.51, 0.97)	(1.32, 0.91)	(1.17, 0.89)	(0.48, 0.82)	(-0.25, 0.42)	xxx
Neg. labels: (mean, deviation)	(-2.39, 0.63)	(-1.92, 0.45)	(-1.60, 0.40)	(-1.41, 0.40)	(-1.07, 0.39)	(-0.84, 0.26)	xxx

Tabla 5. Estudio comparativo para un kernel gausiano variando el parámetro gamma ‘g’

El análisis de la gráfica muestra que el mejor compromiso entre los parámetros *Precision* y *Recall* se encuentra para los valores de 'g' cercanos a 0.175 y 0.2. Aunque nuevamente se encuentren mejores valores de *Precision* para valores de 'g' más grandes, el aumento del tiempo de procesado así como el

descenso del *Recall* nos hacen descartar estas posibilidades.

Igualmente, la gráfica que representa los histogramas nos revela resultados como en el caso polinómico: valores de 'g' pequeños producen que la curva no se ajuste tanto a los puntos, por eso se ven valores con etiquetas tan negativas.

Por todo lo anterior, concluimos en que los mejores kernels gausianos son los que corresponden a g=0.175 y g=0.2.

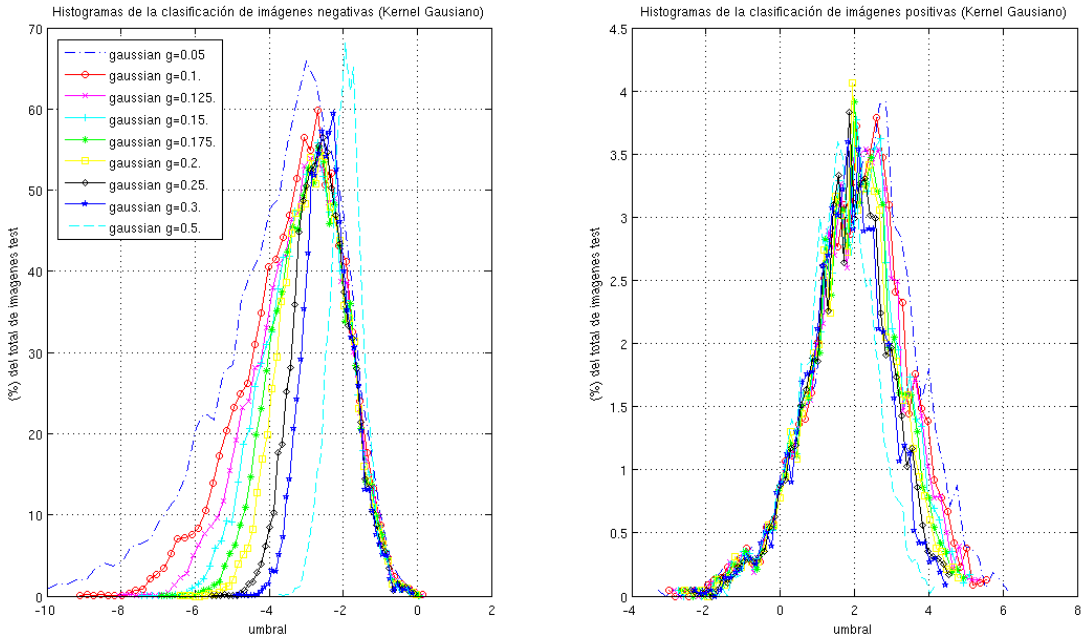

Fig 14: Distribución de la predicción sobre muestras negativas (izquierda) y positivas (derecha).

Comparando los mejores kernels polinómicos con los mejores gausianos, vemos como todos reunen características similares. El parámetro *Recall* es ligeramente superior en el caso de los gausianos preseleccionados. Asimismo, el coste temporal en la clasificación de una muestra es ligeramente mejor para el caso gausiano. Aunque el parámetro *Precision* es más elevado en el caso paramétrico seleccionado, vemos que si hubiéramos seleccionado un kernel gausiano con g=0.25 obtendríamos mejores prestaciones que en el kernel polinómico propuesto, en todos los parámetros. Es por ello por lo que nos inclinamos por la selección de los kernels gausianos con valores de g=0.175 y g=0.2 como los definitivos. Sobre ellos se basarán los apartados siguientes.

IV.4 ESTUDIO DE LA COMBINACIÓN DE ADABOOST CON SVM

En este apartado vamos a ver los resultados que produce la combinación de estos tres métodos en la detección de peatones. Dado que existe un gran número de variables que toman parte en el subsistema final, vamos a realizar suposiciones basadas en los resultados previos de este trabajo y en experiencias de otros trabajos. Así, en cuanto al método de Adaboost vamos a considerar que el factor de escalado de la imagen es de 1 y el factor de agrupación de candidatos es de valor 2. Asimismo, consideraremos el tamaño mínimo de persona buscado de valor: 32x64 (*anchura* x *altura*). (Ver apéndice 1). Realizaremos el estudio

para diferentes números de etapas comprendidos entre 20 y 45, con paso de 5.

En cuanto a los parámetros del algoritmo HOG, trabajaremos con: NBP=4, NBO=4, SIGMA=8, solape entre bloques = ¼, Nptos=8 (número de puntos donde centraremos cada bloque de tamaño NBPxNBP en una imagen 64x128 para calcular el descriptor).

Los parámetros de las máquinas clasificadoras bajo test son: kernel gausiano con g=0.2 y g=0.175. Se han considerado kernels realimentados por su mayor robustez. En el apéndice 2 se detalla el proceso seguido para realizar esta realimentación, así como sus ventajas. Además, se comparan resultados preliminares de los kernels propuestos con y sin realimentación de muestras.

En referencia al conjunto de test, tal y como se comenta en el capítulo de *Metodología* disponemos de dos conjuntos muestrales para realizar el testeo: uno basado en una recopilación de secuencias del PFC de Ana [15] y otro basado en una selección hecha por el autor. En primer lugar, basaremos los resultados en el conjunto recopilado por Ana para permitir la comparación entre métodos.

Dado que los resultados finales dependen directamente de los resultados proporcionados con el método de Adaboost, decidimos en primer lugar analizar las prestaciones de este método independientemente. Por ello, en la tabla siguiente mostramos una relación de valores entre el número de etapas bajo análisis y las prestaciones de Adaboost por sí solo, como si los resultados finales fueran directamente los proporcionados por este método.

En la gráfica observamos como el método de Adaboost produce valores de *Precision* muy malos. Teniendo en cuenta la definición de este parámetro, deducimos que este método produce demasiadas falsas alarmas. En concreto, la probabilidad de falsa alarma normalizada al número real de imágenes positivas de test es muy alta. Esto se puede entender analizando el parámetro FP_ada/Ntest_pos, que nos indica el número de falsas alarmas que Adaboost proporciona por cada persona real en la imagen. Por ejemplo, vemos como para un número de etapas igual a 20, por cada imagen real Adaboost proporciona 5.1 falsas alarmas de media. Este valor se reduce para etapas mayores, llegando a valores de 0.3 para un número de etapas de 45. Sin embargo, aunque para este número de etapas se reduce mucho el número de falsas alarmas proporcionadas por el método, encontramos que el parámetro *Recall* ha descendido significativamente. Analizando la definición de este parámetro encontramos que se produce una elevada pérdida de candidatos con este número de etapas.

	Netap = 20	Netap = 25	Netap = 30	Netap = 35	Netap = 40	Netap = 45
Precision (%) / Recall (%)	-410.07, 59.94	-128.76, 79.33	-26.16, 80.60	34.28, 79.86	50.07, 75.52	69.14, 72.95
FP_ada/Ntest_pos	5.10	2.28	1.26	0.65	0.49	0.30

Tabla 6. Análisis de las prestaciones del método de Adaboost en función del número de etapas.

Hay que llegar, pues, a una solución de compromiso entre estos valores y la aportación los métodos de extracción y clasificación de características.

A continuación se realizan unos estudios entre los dos kernels gausianos seleccionados, para el mismo

número de etapas de la tabla anterior. Los resultados proporcionados por SVM sobre los candidatos que proporciona Adaboost son simples predicciones, cuyo módulo y signo van a determinar la naturaleza de la muestra: positiva o negativa. Por ello hacemos un estudio del mejor umbral para separar las predicciones de ambos conjuntos muestrales. Asimismo, dada la importancia del coste temporal en el sistema final, realizamos un estudio de tiempos asociado a cada subfunción para ver el coste que tiene cada una. Las funciones que analizaremos son las que se encargan de ejecutar cada método:

candidatos_ada = cvHaarDetectObjects(im, cascade, storage, 1.1, agrup, 0, cvSize(Nmin,Mmin));

descriptor_candidatos = calcula_hog(candidatos_ada, im ,P_pt)

predicciones = calcula_predicciones_svm(descriptor_candidatos, kernel)

El campo im es la imagen bajo análisis (320x240 px). El parámetro cascade indica el clasificador a utilizar por Adaboost, cada uno con un número diferente de etapas. Storage es un puntero a estructura donde se almacenan los candidatos propuestos por el método. El valor 1.1 es un factor de escalado. La variable agrup corresponde al factor de agrupación de candidatos que realiza el mismo método; se encarga de agrupar todas aquellas predicciones que están muy juntas unas de otras. CvSize es un tipo de datos de OpenCV y describe el tamaño mínimo de persona que estamos buscando.

Por otro lado, los parámetros del método que calcula los histogramas son: *candidatos_ada,* donde le pasamos las posiciones de los candidatos propuestos por el método de Adaboost, referidos a la imagen bajo análisis (todos los candidatos serán reescalados a 64x128 para poder ser extraidas las características). *im*, que corresponde con un puntero a la imagen bajo análisis. *P_pt* es una matriz que contiene los puntos sobre los que calcularemos el descriptor para cada candidato propuesto por Adaboost.

Respecto a los parámetros de la función que clasifica las muestras, le pasamos el kernel de la máquina y el descriptor con las características de los candidatos de Adaboost.

El resultado es una etiqueta con la predicción asociada a cada candidato de Adaboost.

	Kernel:g=0.175, agr=2					
	Netap = 20	Netap = 25	Netap = 30	Netap = 35	Netap = 40	Netap = 45
th	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)
-0.9	67.42/45.61	77.06/67.14	76.76/70.35	84.48/73.04	88.91/70.17	94.26/68.75
-0.8	73.36/44.48	80.28/65.74	79.69/69.08	**86.04/72.30**	90.12/69.61	94.85/68.37
-0.7	78.34/43.15	83.09/63.90	82.64/67.42	88.17/71.41	91.45/68.63	95.15/67.51
-0.6	82.58/41.35	85.42/62.42	85.04/65.62	89.94/70.23	92.49/67.57	95.53/67.01
-0.5	86.37/40.02	88.05/59.79	87.67/63.55	91.39/69.11	93.61/66.33	96.12/65.56
-0.4	89.18/38.51	90.45/57.37	89.77/61.86	92.69/67.33	94.56/64.76	96.54/64.41
-0.3	91.81/36.89	92.25/54.86	91.48/60.09	93.64/65.17	95.47/63.13	96.95/63.40
-0.2	92.87/35.14	93.85/53.00	92.69/57.93	94.73/62.66	96.33/62.07	97.04/62.34
-0.1	94.02/33.19	95.15/50.84	94.62/55.33	95.68/60.71	97.10/60.86	97.54/60.59
0	95.15/32.01	95.95/48.50	95.38/53.05	96.48/58.29	97.63/59.56	97.96/59.03
0.1	95.77/30.18	96.54/46.14	96.39/50.39	97.07/55.80	97.90/56.84	98.25/57.10
0.2	96.74/28.52	97.36/43.86	96.98/48.21	97.57/53.20	98.22/54.53	98.66/55.06
0.3	97.72/26.60	97.81/41.59	97.51/45.47	97.96/50.39	98.43/52.11	98.87/52.73
0.4	97.98/24.74	98.16/39.16	98.16/42.50	98.45/46.94	98.78/49.92	99.08/50.22
0.5	98.16/22.49	98.69/36.18	98.37/39.99	**98.66/44.10**	98.99/47.53	99.14/47.41

Tabla 7. Análisis de las prestaciones de un kernel gausiano con g=0.175 variando el número de etapas y el umbral de decisión.

	Kernel:g=0.2, agr=2					
	Netap = 20	Netap = 25	Netap = 30	Netap = 35	Netap = 40	Netap = 45
th	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)	(Prec/Rec)
-0.9	66.00/45.78	76.41/67.63	76.26/70.64	84.15/73.30	88.64/70.47	94.147/68.96
-0.8	72.39/44.63	79.84/66.06	79.27/69.55	**85.90/72.48**	89.94/69.7	94.827/68.48
-0.7	77.97/43.42	82.88/64.29	82.35/67.69	87.76/71.59	91.22/68.87	95.093/67.63
-0.6	82.41/ 41.73	85.10/62.57	84.74/65.88	89.71/70.49	92.22/67.69	95.536/67.10
-0.5	86.31/40.20	87.91/59.91	87.31/64.02	91.30/69.25	93.49/66.44	96.127/65.97
-0.4	89.09/38.66	90.18/57.61	89.62/61.95	92.63/67.51	94.50/64.76	96.512/64.73
-0.3	91.75/36.97	92.19/55.06	91.48/60.21	93.64/65.35	95.50/63.22	96.9258/63.4
-0.2	92.99/35.14	93.82/53.14	92.69/58.17	94.82/62.93	96.27/62.01	97.0736/62.4
-0.1	93.96/33.25	95.24/50.93	94.62/55.45	95.71/60.77	97.13/60.86	97.546/60.71
0	95.24/32.13	95.95/48.56	95.47/53.09	96.60/58.43	97.69/59.47	97.989/58.91
0.1	95.80/30.03	96.54/46.26	96.42/50.37	97.10/56.07	97.90/56.93	98.315/57.22
0.2	96.98/28.46	97.36/43.77	96.98/48.24	97.63/52.97	98.34/54.65	98.66/55.0
0.3	97.78/26.51	97.81/41.41	97.66/45.19	98.01/50.22	98.46/52.17	98.876/52.70
0.4	97.98/24.41	98.19/38.78	98.31/42.27	98.52/46.76	98.81/49.89	99.083/50.13
0.5	98.19/22.16	98.64/35.97	98.37/39.66	**98.75/43.83**	98.99/47.23	99.142/47.44

Tabla 8. Análisis de las prestaciones de un kernel gausiano con g=0.2 variando el número de etapas y el umbral de decisión.

Las tablas anteriores muestran resultados muy parecidos para ambos kernels. Sin embargo, se aprecia una ligera mejora en el *Recall* para el kernel entrenado con g=0.175 en ciertas celdas de la tabla más relevantes que a continuación justificamos. Sobre este kernel realizaremos la discusión siguiente y justificaremos la selección.

El análisis de los resultados para las varias etapas y varios umbrales nos muestra una amplia dispersión de valores. En los subapartados anteriores, donde hacíamos un análisis de las prestaciones de cada método por separado, nos interesaban resultados diferentes. Por ejemplo, en el caso del estudio individualizado del método de Adaboost, nos interesaban valores de *Recall* elevados, aunque ello fuera en detrimento del *Precision*. Por otro lado, en el estudio de la máquina clasificadora buscamos valores altos de *Precision*. Sin embargo, llegados a este punto de combinación de métodos lo que buscamos es una buena relación entre ambos parámetros.

Un análisis de las tablas nos muestra como ambos parámetros van inversamente relacionados. Para valores elevados de *Precision* se dan valores muy bajos de *Recall*, y viceversa. Hay que recordar que la combinación de estos dos métodos actúa directamente sobre los resultados del método de Adaboost, con lo que a priori no se puede mejorar las pérdidas que produce el primer método de la cadena. Sin embargo, si que se puede mejorar considerablemente la *Precision* con otros métodos que luego propondremos.

El análisis previo del método de Adaboost por aislado nos indica que el mejor clasificador de Adaboost (el que menos limita al subsistema total) es el que tiene 35 etapas. Con él se obtiene el menor ratio de pérdidas. Este hecho se nota directamente en las tablas, obteniéndose los valores más altos de *Recall* para umbrales en torno a -0.8 y -0.9. Y es que analizando la tabla 6 obtenemos que el límite que impone el método de Adaboost es de un $Recall_{opt}$ cercano al 80%, para 30 y 35 etapas. Sin embargo, vemos como el

parámetro *Precision* no alcanza su máximo. Es la combinación con SVM lo que mejora el *Precision* de Adaboost.

No obstante, el valor del parámetro *Precision* final no es demasiado elevado para el caso del óptimo *Recall*. Este hecho no implica que el sistema no pueda ser mejorado posteriormente, ya que ello indica que hay demasiada falsa alarma, con lo que se van a poder estudiar otras técnicas que actúen en la cadena y mejoren este valor.

Sin embargo, el hecho de que el *Recall* sea bajo es un punto más crítico. Ello indica que se pierden muestras, con lo que a priori resulta que no se van a poder recuperar.

Para completar el estudio representamos en una gráfica otros parámetros de interés, como son el MR y el FPPW, definidos en la sección de *Metodología*. La gráfica que sigue muestra esta relación entre parámetros, realizada para todas las etapas del clasificador, y modificando los umbrales de decisión.

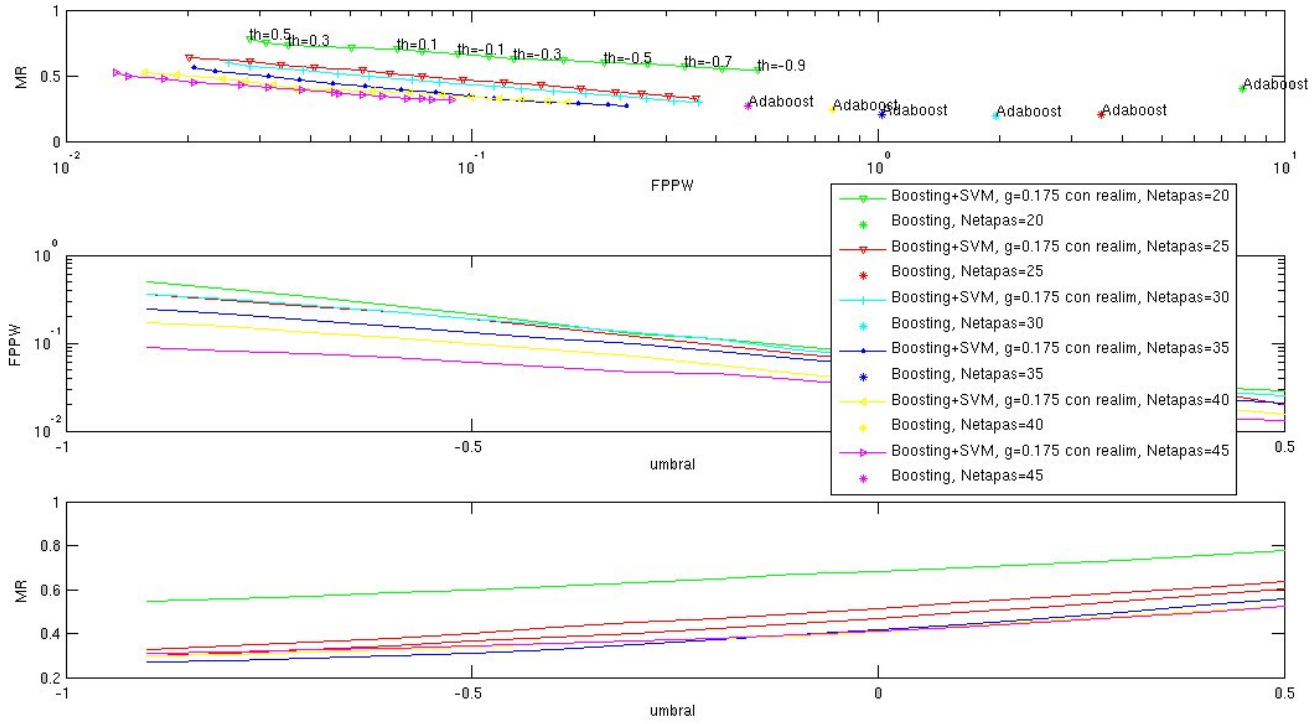

Fig 15. Comparación entre diversos clasificadores finales variando el número de etapas de Adaboost, para un kernel con g=0.175.

Nos centramos en primer lugar en la gráfica anterior. En ella observamos marcado con un asterisco los valores que proporciona el método de Adaboost por sí solo. Acto seguido, con el mismo color se muestra la relación entre los parámetros FPPW y MR, obtenidos tras juntar Adaboost con SVM, y variando el umbral de decisión de SVM. Cuanto más a la izquierda y más abajo, mejores resultados. Observamos cómo el comportamiento del MR tiene una tendencia a la inversa que el FPPW. Asímismo, vemos como los asteriscos indicativos de las prestaciones de Adaboost imponen una limitación del sistema. Se observa como valores menores en el MR que los marcados por Adaboost no se pueden alcanzar. Sin embargo, observamos como sí que se puede mejorar el parámetro FPPW. Este parámetro está ligado con las falsas alarmas, con lo que según lo citado anteriormente, observamos como puede ser mejorado.

El problema se manifiesta igual que en la tabla anterior: para optimizaciones del parámetro FPPW, empeoramos el MR (perdemos candidatos reales, hecho que no deseamos).

Las gráficas que siguen muestran la relación del FPPW y el MR con el umbral. Observamos como el FPPW mejora con etapas elevadas, pero en detrimento del MR. También observamos que el mejor MR producido para valores razonables de FPPW es proporcionado por un clasificador con 35 etapas.

Otro factor importante y que no hay que perder de vista es el análisis del coste computacional y su repercusión en el coste temporal. La tabla 9 siguiente muestra una relación entre costes para las diferentes etapas.

Kernel:g=0.175, agr=2						
tpos:	Netap = 20	Netap = 25	Netap = 30	Netap=35	Netap=40	Netap=45
Nº cand ada	8.84	4.80	3.24	2.27	1.97	1.64
t_ada	0.0488	0.0484	0.0490	0.0473	0.0492	0.0482
t_sift	0.1416	0.0781	0.0528	0.0364	0.0319	0.02587
t_svm	0.3149	0.1697	0.1168	0.0804	0.07139	0.05788
tiempo total	0.5055	0.2963	0.2187	0.1642	0.1526	0.1319

Tabla 9[8]. Análisis de tiempos del subsistema final utilizando un kernel gausiano con g=0.175 y variando el número de etapas de Adaboost.

Observamos que la mayor parte del tiempo es consumida en la clasificación de las muestras. Por ello, un menor número de muestras a clasificar aumentará considerablemente el tiempo de clasificación requerido por la máquina SVM. Los peores tiempos se dan para un bajo número de etapas. Esto es así debido a que adaboost da un elevado número de falsas alarmas que la máquina SVM deberá clasificar con posterioridad. Por contra, los mejores tiempos se dan para un número de etapas de clasificación en Adaboost elevados. Esto es así porque a mayor número de etapas se reduce la falsa alarma, con lo que el clasificador tiene menos muestras que comprobar. Sin embargo, como hemos visto antes, valores altos de etapas producen demasiadas pérdidas. Para el número de etapas óptimo encontrado por el momento (Netapas = 30 ó 35), los tiempos de computación son razonables. Sin embargo, encontramos una importante mejora en el caso de 35 etapas, ya que el tiempo total se reduce en casi 50 ms. Este hecho hace acumular más votos a la elección del clasificador con 35 etapas, aunque aun quedan más resultados por analizar.

La mejora en 50 ms es importante. Tenemos que recordar que el sistema está pensado para operar en tiempo real y ser capaz de procesar cada imagen en un intervalo menor a 100ms (10Hz) por las imposiciones del sistema. Por ello, una mejora en 50 ms es más que buena en este caso, cuando además no se pierden prestaciones.

Para completar el estudio de la mejor combinación entre métodos, hemos realizado un estudio visual

[8] *Los resultados para el kernel de g=2 pueden encontrarse en el apéndice 3, y presentan valores ligeramente superiores a los del kernel con g=0.175, lo cual refuerza su elección.*

del funcionamiento del subsistema sobre las secuencias de vídeo comentadas anteriormente. La Figura 16 muestra unos ejemplos de dichas secuencias.

De la figura se extraen diversas conclusiones acerca del funcionamiento de Adaboost y de SVM. En primer lugar, podemos observar cómo para menor número de etapas Adaboost da muchas más falsas alarmas. Además, no siempre encuadra de la misma forma al candidato. Dependiendo del número de etapas, realiza un encuadre mejor o peor. Esto es significativo para el funcionamiento de SVM, ya que la extracción de características sobre una persona cuyo cuadro está ligeramente desplazado hacia un lado puede llevar a la mala clasificación de la muestra, a pesar de que Adaboost la había reconocido. Vemos por ejemplo en la primera imagen cómo de la predicción con 20 etapas a la de 25, el bounding box de la persona varía ligeramente. Ello puede llevar a perder el candidato, por los motivos citados anteriormente. Igualmente, en la fila 4 encontramos un caso donde debido a una leve imprecisión en el marcado del bounding box por Adaboost, en las etapas 35, 50 y 45 no se detecta a un candidato, mientras que en las etapas anteriores (con diferente encuadre de las personas) sí que se hacía. De nuevo encontramos en la secuencia de la fila 5 cómo para etapas de 35, 40 y 45 el bounding box proporcionado por Adaboost varía ligeramente tanto de tamaño como de posición para la chica de la imagen. Las malas condiciones en el bounding box producen la mala clasificación por SVM y por consiguiente, la pérdida de la muestra en las etapas 40 y 45.

Podemos extraer del análisis visual de este grupo de imágenes de test que los mejores resultados globales se dan para un número de etapas del clasificador de Adaboost entre 30 y 35.

20 etapas 25 etapas 30 etapas

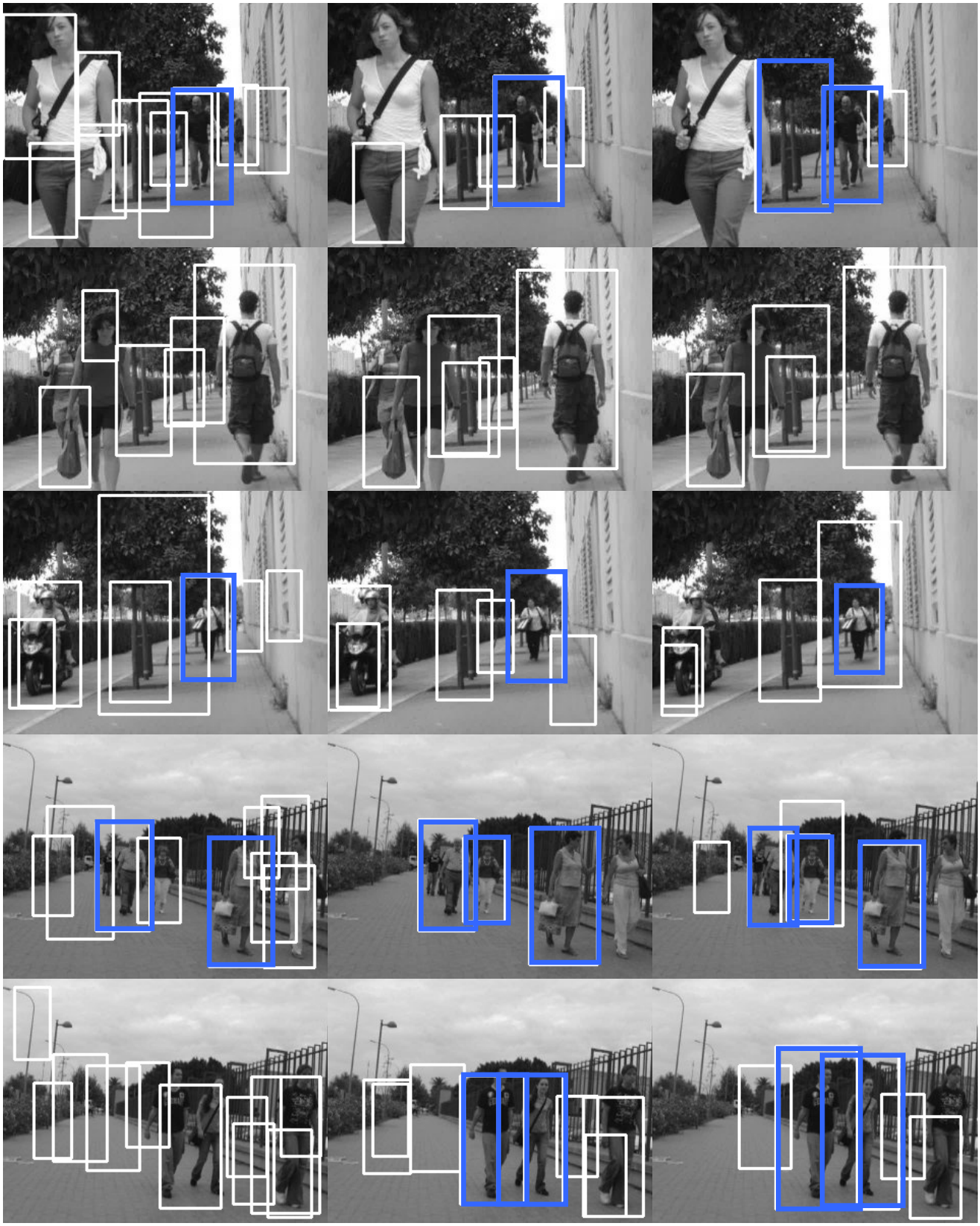

Fig 16. Ejemplo de funcionamiento del subsistema final para varias imágenes y varias etapas. En blanco aparecen representados los candidatos propuestos por Adaboost. En azul, los candidatos clasificados como “positivos” por SVM (con un kernel gausiano con g=0.175 y th=-0.5), y por tanto los candidatos finales propuestos por el subsistema total.

En el apéndice 4 se incluye otra recopilación de imágenes diferentes.

V. CONCLUSIONES Y PERSPECIVAS

Este trabajo se enmarca dentro del proyecto CASBLIP, Proyecto Europeo del 6º Proyecto Marco. La finalidad del proyecto CASBLIP es la de generar un prototipo que sirva de ayuda a las personas ciegas o con poca visión para poder guiarse en entornos abiertos. El sistema cuenta con un par de estéreo-cámaras, un sensor láser CMOS, un sistema de posicionamiento de la cabeza (HPS), un sistema inercial y un GPS. La concepción inicial de esta tarea era la elaboración de un modelo 3D capaz de integrar los diversos dispositivos fuente y de realizar una detección de objetos en tiempo real. Tras unos meses de estudio, se comprobó que la mejor forma de abordar el problema de reconocimiento de objetos para hacerla efectiva en tiempo real era el procesado de imagen en 2D, descartando la creación de modelos en 3D por su elevado coste computacional. Esta opción sigue permitiendo la integración de este subsistema de detección de objetos con el resto de dispositivos de entrada.

El primer paso realizado fue centrarse en la detección de personas y estudio de la viabilidad de aplicar métodos ya existentes. Se vio que los métodos actuales, a pesar de dar muy buenos resultados de predicción, no sirven para nuestro trabajo, en primer lugar porque pocos sistemas de detección de objetos van montados sobre un sujeto móvil, donde el fondo es cambiante y las imágenes empeoran su calidad, y en segundo lugar por el hecho de ser un sistema en tiempo real.

Por ello, decidimos recurrir a la combinación de dos métodos con propiedades complementarias. Por un lado, se ha comprobado que las técnicas de boosting relacionadas en detección de personas proporcionan resultados rápidos, aunque con demasiada falsa alarma. Para reducir esta elevada falsa alarma decidimos combinar estas técnicas con otras técnicas como son las de extracción y clasificación de características. Conocimientos previos sobre estas técnicas nos mostraban una gran precisión en el funcionamiento de las máquinas clasificadoras, aunque conocíamos su elevado coste computacional. Por ello, decidimos combinar ambas técnicas.

El primer paso fue seleccionar un algoritmo de extracción de características robusto, como es el HOG. En segundo lugar, tratamos de mejorar calidad de las características extraídas así como su coste temporal. Para ello eliminamos la dependencia del gradiente con el sentido del mismo, tomando sólo en consideración su dirección. En segundo lugar establecemos los puntos óptimos donde extraer las características. Ello implica seleccionar un solape entre bloques óptimo, no sólo en robustez de las características, sino en eficiencia temporal. Para ello se realizó un estudio de las dimensiones del vector de características frente a los resultados estadísticos y el coste computacional, que reveló que el tamaño óptimo de bloque es de 4x4 celdas, cada celda con 8 píxeles, distribuidos con un solape entre bloques de la cuarta parte, y no calculando el descriptor en aquellos puntos de la imagen situados en los bordes, por el hecho de contener información no característica de las muestras, además de aumentar inútilmente las dimensiones del vector. Una vez optimizado el algoritmo de extracción de características nos centramos en la elaboración de una máquina de clasificación de características robusta. Para ello hemos probado diferentes tipos de kernels, obteniendo un el óptimo para un kernel con función gausiana con valor de

gamma = 0.175. Hemos procedido a la realimentación de la máquina para hacerla más robusta, y así obtener la máquina final.

Una vez optimizado el algoritmo de extracción de características y elaborado una máquina de clasificación robusta, procedemos a su combinación con las técnicas de boosting.

El primer eslabón de esta combinación será el método de boosting, por las propiedades que presenta citadas anteriormente. El segundo eslabón de este subsistema lo compone la combinación de métodos HOG y SVM, que operan sobre los candidatos propuestos por Adaboost.

Hemos analizado los diversos parámetros de funcionamiento de ambos métodos para llegar a una solución óptima. Se han probado diferentes etapas en el algoritmo de boosting. Asimismo, se ha buscado el umbral óptimo en las predicciones de SVM. Se han tratado de optimizar los parámetros de *Precision* y *Recall*, que hacen referencia a la robustez del subsistema frente falsas alarmas y frente a la probabilidad de pérdida de candidatos, respectivamente.

El estudio de las prestaciones de este subsistema nos ha revelado que la combinación de estos dos métodos es satisfactoria, aunque no es óptima. Por una parte, el subsistema cumple con las especificaciones temporales de funcionamiento (~10 Hz en el procesado de cada frame). Sin embargo, en cuanto a la fiabilidad del método no se ha llegado a cumplir con lo esperado debido principalmente a que los candidatos proporcionados por Adaboost sufren ligeras desviaciones en su bounding box, con lo que la clasificación de estas muestras con SVM produce un elevado número de pérdidas en el sistema. El encaje de la persona en el bounding box predecido por Adaboost parece mejorar para un número de etapas mayor. Sin embargo, a mayor número de etapas existen muchas más pérdidas de candidatos. Por todo ello, se decide establecer el número de etapas definitivo (a falta de combinar estos métodos con otras mejoras) igual a 35. Aunque las prestaciones del subsistema final para un número de etapas de 30 son similares a las prestaciones con 35, observamos cómo existe una gran mejora en cuanto a coste temporal del subsistema final, llegándose a disminuir el tiempo de procesado por cada frame en 50 ms, hecho que es de vital importancia para cumplir con las especificaciones temporales.

Aunque el sistema da bastantes buenos resultados, está lejos de cumplir con los óptimos, y por tanto de ser un sistema definitivo. Para mejorarlo se proponen diversas opciones, tanto metódicas como de fusión con nueva información sobre la escena, como son el uso de la información proporcionada por los depth maps generado por las estéreo-cámaras o la información que proporciona el sistema de posicionamiento de la cabeza.

Como trabajo futuro se proponen las siguientes mejoras:

En primer lugar haremos un estudio del resto de variables de ambos métodos que no se han tenido en consideración en este trabajo, y se han tomado valores según bibliografía.

En segundo lugar, para mejorar el alto ratio de pérdidas que produce Adaboost y las ligeras desviaciones que presentan algunos bounding box sobre los candidatos, vamos realizar técnicas de

tracking. Esto consistirá en analizar el historial de candidatos encontrados en los instantes anteriores por si en algún momento dado Adaboost no nos proporciona el candidato, el algoritmo de extracción de características barra una cierta área alrededor de las predicciones anteriores para así comprobar si realmente no existe persona o es que Adaboost ha perdido el candidato. Asimismo, para mejorar las desviaciones en los bounding box de Adaboost, nos basaremos en el historial de las muestras para reajustar el encuadro de la persona.

En tercer lugar, vamos a calcular los histogramas sobre imágenes de tamaño 32x64, dado que el algoritmo clasifica bien personas con tamaños similares en la imagen. De esa forma, estaremos reduciendo el número de características del descriptor, y con ello acelerando el proceso de clasificación y posiblemente mejorando las prestaciones de clasificación del kernel.

En cuarto lugar, podemos realizar la descomposición de los candidatos en partes y analizar cada una de ellas independientemente. Estas partes son las más significativas, como las piernas, el tronco o la cabeza. Existen recientes estudios previos que usan estas técnicas y se consiguen buenos resultados [23]

En quinto lugar vamos a intentar reducir el número de vectores que usa el kernel definitivo mediante ciertas técnicas de optimización de máquinas clasificadoras. La reducción del número de vectores influye positivamente en el tiempo de procesado, y no afecta a la traza de la curva entre los dos conjuntos muestrales.

En sexto lugar, para aquellas personas que debido a su cercanía a la cámara no pueden ser detectadas, vamos a realizar un estudio de otras técnicas que nos puedan permitir su detección. A priori, técnicas basadas en Información Mutua sobre la imagen junto con un seguimiento de la persona en instantes anteriores puede darnos buenos resultados.

En séptimo lugar, debido al uso de estéreo-cámaras en el sistema completo, disponemos para nuestro aprovechamiento los mapas de profundidades de la escena. El tener en cuenta esta información va a ayudarnos a filtrar aquellas áreas donde no van a poder haber personas o bien donde no nos va a interesar buscarlas por estar demasiado lejanas. Ello va a permitir mejorar la precisión de las predicciones, eliminando falsas alarmas situadas en zonas no permitidas, y a la vez va a permitir eliminar bounding boxs demasiado elevados o demasiado pequeños, en función del tamaño esperado de la persona por encontrarse a una distancia u otra.

En último lugar, la utilización de la información proporcionada por el sensor de posicionamiento de la cabeza del usuario del sistema puede ser tenida en cuenta para realizar rotaciones necesarias de aquellas imágenes adquiridas con un cierto ‘roll’.

Hay que tener en cuenta que el problema de detección de objetos en tiempo real supone un verdadero reto en la actualidad. Al problema del coste temporal va unido el hecho de que el sistema va a funcionar montado sobre un sujeto móvil, hecho que lo hace todavía más desafiante. Además, la posibilidad de integración de todos los dispositivos citados anteriormente hace del sistema un conjunto muy atractivo capaz de dar muy buenos resultados, además de disponer de un elevado número de aplicaciones.

AGRADECIMIENTOS

Este trabajo queda enmarcado en el Proyecto Europeo CASBLIP (Cognitive Aid System for Blind People) del 6º Programa Marco y ha sido realizado en el Centro de Investigación en Tecnologías Gráficas (CITG) en paralelo con el Grupo de Procesamiento de Imagen (GPI) del ITEAM, ambos en la UPV.

En primer lugar quiero agradecer a mis directores de Tesina, Alberto y Guillermo, su impecable labor en la dirección técnica y científica de este trabajo. En particular, quisiera agradecer a Guillermo el haberme facilitado siempre todos los medios que he necesitado para desarrollar este trabajo con éxito, así como todo el conocimiento que he aprendido a su lado en muchos otros campos. A Alberto, agradecer su dedicación y empeño en darme una suficiencia científica e investigadora que va a constituir la base de mi futuro profesional. A ambos les agradezco haber depositado plenamente su confianza en mí.

Quiero dar las gracias también a todos mis compañeros del CITG por su agradable compañía. También a mis compañeros del laboratorio en Teleco, especialmente a David por toda su ayuda prestada. A mis compañeros del master por haber hecho el curso más agradable… Y cómo no a mis amigos de siempre.

A mi familia y a Nieves.

APÉNDICE 1. ESTUDIO DEL TAMAÑO MÍNIMO DE PERSONA PARA SER DETECTADO

El método de Adaboost permite especificar un parámetro indicando el tamaño mínimo de persona que se va a detectar. Para ello, realizamos un estudio tomando varias fotos de una persona en varias profundidades. Las profundidades testeadas han sido de 2, 4, 6, 8 y 10 m. La persona es de un tamaño medio (1.6 m), y la cámara está situada a una altura de 1.8m. Las dimensiones de la imagen son 320x240, por compatibilidad con las cámaras del sistema global CASBLIP. Por otro lado, el tamaño de la persona en píxeles ha sido tomado cogiendo distancia de la cabeza a los pies, sin dejar ningún margen adicional.

A continuación se detallan en la tabla los resultados.

distancia	2 m	4 m	6 m	8 m	10 m
ancho x alto	120x240	66x132	44x88	33x66	27x55

Tabla 10. Análisis del tamaño de una persona en píxeles en diferentes posiciones para una imagen 320x240

La distancia máxima a la que deseamos detectar personas es de 10 m, y viene marcada por los requerimientos del sistema global. Como en todos los ejemplos utilizados para entrenar tanto el algoritmo de Adaboost como la máquina SVM existen unos márgenes alrededor de 10 px alrededor de las muestras positivas, seleccionamos un tamaño mínimo de persona igual a 32x64. Hay que recordar que este valor va ligado al tamaño de la imagen inicial (320x240), con lo que para hacer el sistema compatible con otros tamaños de imagen, debemos establecer el tamaño mínimo como: $(anchura\_\min, altura\_\min) = (32 \cdot factor, 64 \cdot factor)$, siendo 'factor' la relación entre el nuevo tamaño de imagen y la imagen base 320x240.

APÉNDICE 2. ESTUDIO DEL EFECTO DE REALIMENTACIÓN DE LA MÁQUINA

La ventaja de este tipo de máquinas clasificadoras es la posibilidad que permiten de aprendizaje. Esto es, la posibilidad de ser re-entrenadas con todas aquellas muestras que en otras ocasiones ha clasificado mal. Este hecho hace a la máquina más robusta con cada entrenamiento.

Para probar el efecto que tiene el entrenamiento de la máquina, partimos de un kernel previo considerado como definitivo, en este caso un kernel paramétrico con d=3, entrenado con 5364 ejemplos positivos y ~12000 ejemplos negativos. Por otro lado, disponemos para el testeo de otro conjunto muestral extenso con ejemplos negativos. Clasificamos estos ejemplos negativos con la máquina, y obtenemos unas predicciones. Seleccionamos todas aquellas muestras mal predichas o cuyo valor se encuentre cercano a cero, y las incluimos en el conjunto muestral negativo con el que hemos entrenado inicialmente la máquina, aumentando el número de ejemplos.

Para ver con mayor claridad el efecto del re-entrenamiento, repetimos el proceso para un número de

muestras de test mayor, y rellenamos la tabla 11 (hemos considerado el umbral: th=0).

En la figura 17 mostramos un pequeño conjunto de muestras que son realimentadas. Apreciamos cómo existen ciertas formas que pueden asemejarse a las piernas y silueta de una persona... y es por eso que la máquina inicial las ha clasificado erróneamente.

Fig. 17. Ejemplos de imágenes que han sido clasificados como positivos por las primeras versiones del clasificador. Posteriormente han sido realimentados en la máquina para mejorar su robustez.

	kernel sin realimentar	kernel realimentando con 4800 muestras negativas	kernel realimentando con 15000 muestras negativas
Accuracy (%)	99.32	99.66	99.57
Precision (%) / Recall(%)	93.01 / 98.72	97.01 / 97.54	99.618 / 93.57
Tiempo de clasificación por muestra (s)	0.0052	0.0060	0.0127

Tabla 11: Resultados del efecto de realimentación de muestras en un kernel.

En la tabla 11 apreciamos como tras la realimentación del kernel con 4800 muestras negativas, obtenemos considerables mejoras en cuando al parámetro *Precision*. Sin embargo, disminuimos ligeramente el *Recall*. Esto es debido a que el parámetro *Precision* está ligado con las falsas alarmas, y debido a que hemos realimentado con muchos más ejemplos negativos, hemos reducido este problema.

Para el caso de realimentación con 15000 muestras, mejoramos mucho más el parámetro *Precision* de la máquina. Sin embargo, notamos un importante descenso en el *Recall*. Esto se debe a que el conjunto muestral de ejemplos negativos para el entrenamiento es mucho mayor que el conjunto de ejemplos positivos (~27000 frente a ~5400). Por ello, la máquina tiende a perder candidatos.

Sin embargo, esto no tiene por qué ser un problema. Si modificamos el umbral de decisión, podemos obtener resultados de *Recall* suficientemente buenos, sin llegar a perder excesiva *Precision*. La gráfica 18 muestra la relación entre el parámetro FPPW y el MR, variando el umbral de decisión.

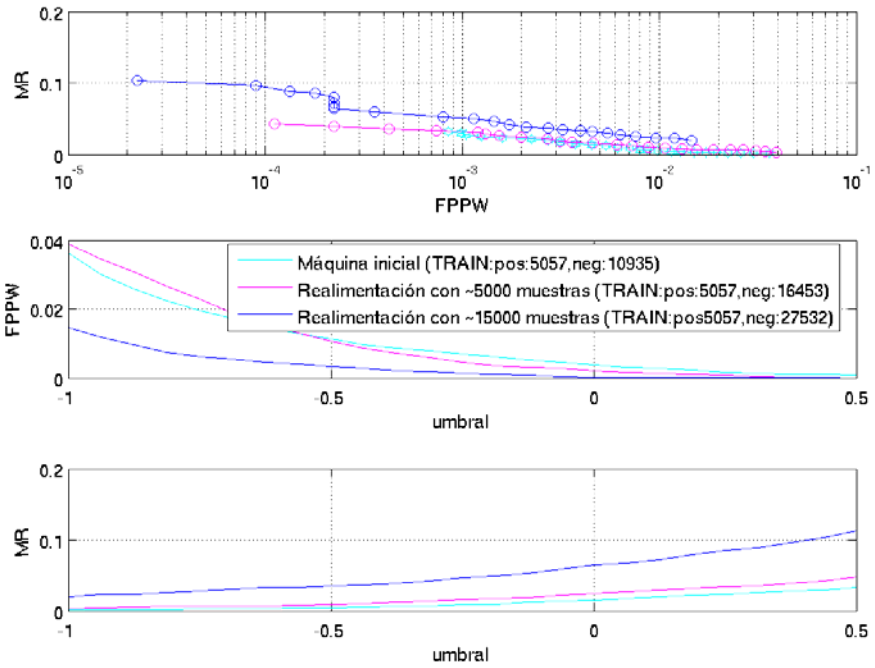

Fig18. Análisis de la influencia de la realimentación de un kernel paramétrico con d=3.

La figura de abajo muestra de nuevo el efecto de la realimentación, en este caso para los dos kernels gausianos propuestos para ser evaluados en el punto IV.2. Vemos como aunque el MR de las máquinas realimentadas alcanza valores más bajos, no puede alcanzar los valores de FPPW que alcanzan las máquinas realimentadas. Nótese que esta gráfica no puede compararse con la anterior, puesto que en esta estamos evaluando los dos métodos (Adaboost y SVM) conjuntamente, mientras que en la anterior solo evaluábamos las prestaciones de SVM. Además, el conjunto de imágenes de test no es el mismo.

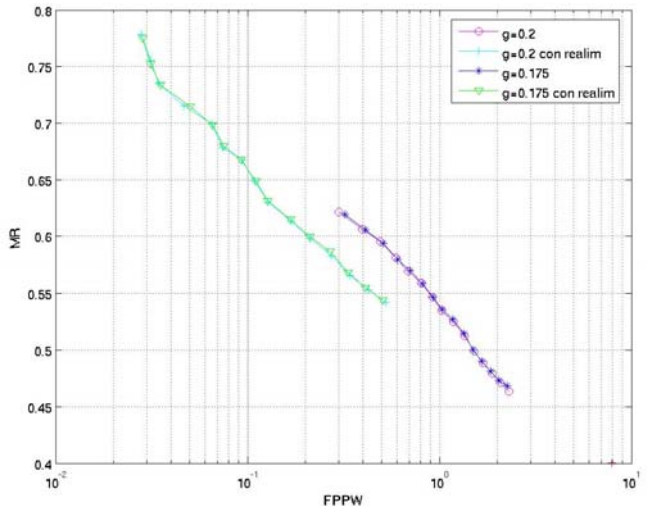

Fig18. Análisis de la influencia de la realimentación de un kernel paramétrico con d=3 y 20 etapas.

Por todo lo citado anteriormente, podemos concluir en que es necesario realizar el re-entrenamiento de la máquina para lograr los valores de *Precision* adecuados.

APÉNDICE 3. TABLA DE TIEMPOS PARA UN KERNEL GAUSIANO CON G=0.2

La siguiente tabla muestra el coste temporal medio para cada método en la detección de personas en una imagen 320x240, para la combinación de técnicas de boosting con extracción y clasificación de características. Se prueban varias etapas en el boosting. La máquina clasificadora ha sido calculada mediante una función gausiana con g=0.2.

	Kernel:g=0.2, agr=2					
tpos:	Netap = 20	Netap = 25	Netap = 30	Netap=35	Netap=40	Netap=45
Nº cand ada	8.84	4.80	3.24	2.27	1.97	1.64
t_ada	0.0485	0.0474	0.0478	0.0490	0.04811	0.0483
t_sift	0.1426	0.0787	0.0522	0.0368	0.0309	0.0258
t_svm	0.3270	0.1746	0.1184	0.0847	0.0715	0.0595
tiempo total	0.5183	0.3008	0.2185	0.1706	0.1505	0.1338

Tabla 12. Análisis de tiempos del subsistema final utilizando un kernel gausiano con g=0.2 y variando el número de etapas de Adaboost.

APÉNDICE 4. EJEMPLO DE FUNCIONAMIENTO DEL SUBSISTEMA DEFINITIVO

20 etapas 25 etapas 30 etapas

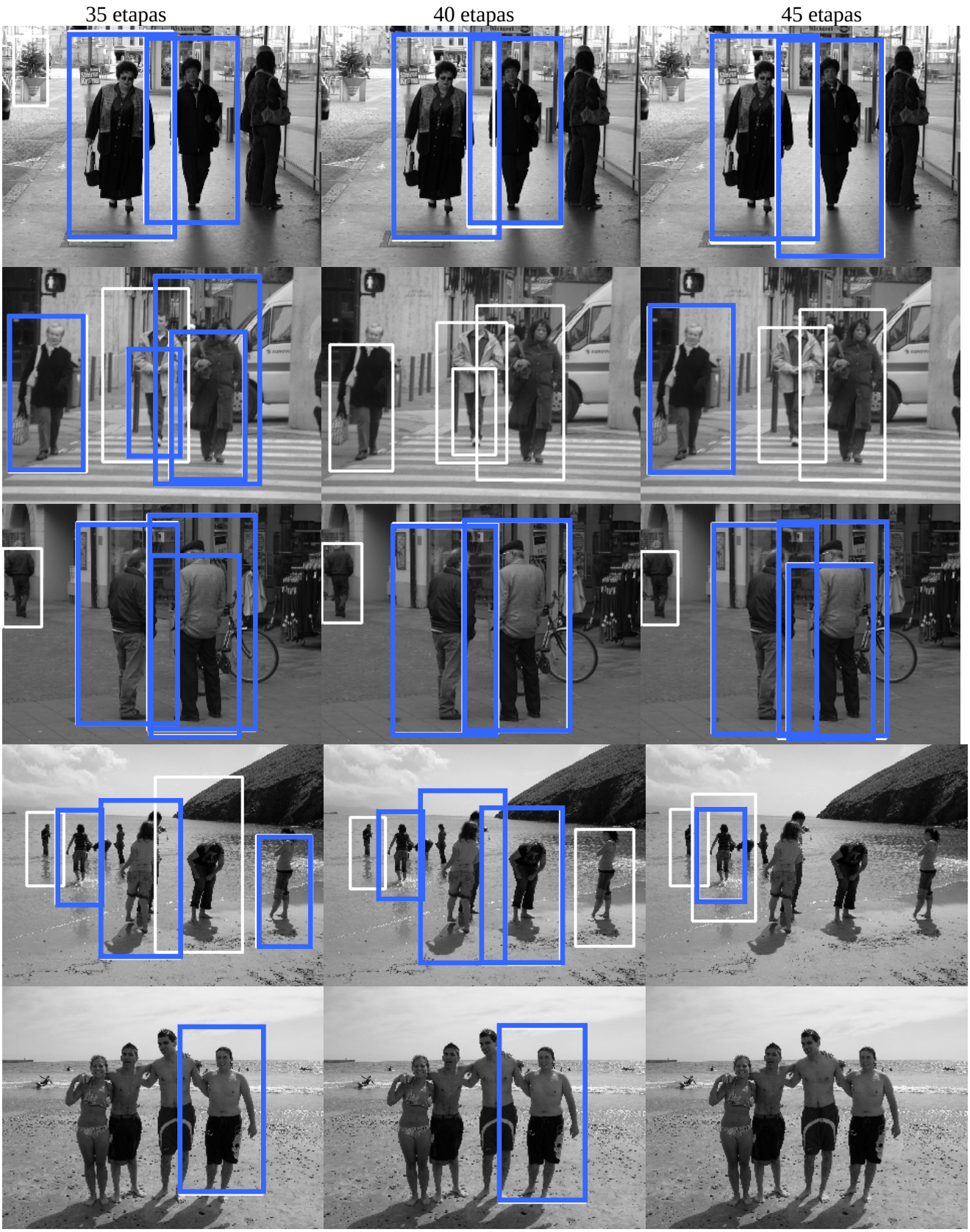

Fig 18. Ejemplo de funcionamiento del subsistema final para varias imágenes y varias etapas. En blanco aparecen representados los candidatos propuestos por Adaboost. En azul, los candidatos clasificados como “positivos” por SVM (con un kernel gausiano con g=0.175 y th=-0.5), y por tanto los candidatos finales propuestos por el subsistema total.

REFERENCIAS

[1] N. Dalal, *Finding People in Images and Videos*, Tesis Doctoral, Julio 2006.

[2] N. Dalal and Bill Triggs, *Histograms of Oriented Gradients for Human Detection*. Proceedings of IEEE Conference Computer Vision and Pattern Recognition, San Diego, USA, pages 886-893, June 2005

[3] A. Broggi, M. Bertozzi, A. Fascioli, and M. Sechi, *Shape-based pedestrian detection* in Proc. IEEE Intell. Veh. Symp., Dearborn, MI, Oct. 2000, pp. 215-220.

[4] C.Papageorgiou and T.Poggio, *A trainable system for object detection*, Int. J. Comp. Vis., vol. 38, no.1, pp.15-33, 2000

[5] A. Mohan, C. Papageorgiou, and T. Poggio, *Example-based object detection in images by components*, IEEE Trans. Pattern Anal. Mach. Intell., vol.23, no. 4. pp.349-361, Apr. 2001

[6] D. M. Gavrila, J. Giebel and S. Munder, *Vision-based pedestrian detection: The protector system*, in Proc. IEEE Intell. Veh. Sympo, Parma, Italy, Jun. 2004, pp.13-18

[7] H. Nanda and L. Davis, Probabilistic *template based pedestrian detection in infrared videos*, in Proc. IEEE Intell. Veh. Symp., Versailles, France, Jun. 2002, pp.15-20.

[8] H. Cheng, n. Zheng, and J. Qin, *Pedestrian detection using sparse Gabor filter and support vector machine*, in Proc. IEEE Intell. Veh, Symp. Las Vegas, NV, Jun. 2005, pp. 583-587

[9] F. Suard, A. Rakotomamonjy, A. Bensrhair, and V. Guigue, *Pedestrian detection using stereo-vision and graph kernels*, in Proc. IEEE Intell. Veh. Symp. Las vegas, NV, Jun. 2005, pp. 267-272.

[10] C. Curio, J. Edelbrunner, T. Kalinke, C. Tzomakas, and W. V. Seelen, *Walking pedestrian recognition*, IEEE Trans. Intell. Transp. Syst. Vol. 1, no. 3, pp. 155-163, sep. 2000

[11] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, *Autonomous driving goes downtown*, IEEE Intell Syst Their Appl. Vol. 13, no.6, pp. 40-48, Nov 1998.

[12] L. Zhao and C. E. Thorpe, *Stereo and neural network-based pedestrian detection*, IEEE Trans. Intell. Transp Syst, vol. 1, no. 3, pp. 148-154, Sept 2000

[13] Paul Viola and Michael Jones. *Rapid object detection using a boosted cascade of simple features*. IEEE Computer Vision and Pattern Recognition, 2001, pp. 25-29.

[14] Yoav Freund and Robert E. Schapire. *Experiments with a new boosting algorithm*. Machine Learning: Proceedings of the Thirteenth International Conference, 1996. p 17.

[15] A. Pardo, A. Albiol. *Detección Automática de Peatones usando técnicas de Boosting*, Proyecto Fin de Carrera, Dic 2006

[16] F. J. Canny. *A computacional approach to edge detection*. IEEE Trans. Pattern Analysis Mach. Intell. Vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986

[17] R.M. Haralick, Statistical *and structural approaches to texture*, Proc. IEEE, vol. 67, no.5, pp.786-804, May 1979.

[18] L. Wang, *Texture unit, texture spectrum and texture analysis*, IEEE Trans. Geosci. Remote Sens., vol. 28, no. 4, pp. 509-512, Jul. 1990

[19] Pai-Hsuen Che, Chih-Jen Lin and Bernhard Scholkopf. *A Tutorial on v-Support Vector Machines.*

[20] Thorsten Joachims. *Support Vector Machine*. University of Dortmund, Informatik, AI-Unit, Sept 2004.

[21] MIT Pedestrian database available at: http://cbcl.mit.edu/software-datasets/PedestrianData.html

[22] INRIA Person dataset available at http://pascal.inrialpes.fr/data/human/

[23] I. Parra, D. Fernández, M. A. Sotelo, *Combination of Feature Extraction Methods for SVM Pedestrian Detection*, IEEE Trans. On Int. Transp Syst, vol. 8, no. 2, Junio 2007

[24] H. Poor, *An Introduction to Signal Detection and Estimation*. York: Springer-Verlag, 1985, ch. 4.

[25] W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135

ANEXOS

[1] J. Oliver, A. Albiol, G. Peris, L. Dunai. *HOG descriptor improvement in person detection by jeans of the reduction of the space dimensions*. VIII Jornadas de Matemática Aplicada, UPV, September 2007. Poster

[2] L. Dunai, M. Fernández, G. Peris, J. Oliver. Spatial *Sound localization based on Fourier Transform.* VIII Jornadas de Matemática Aplicada, UPV, September 2007. Poster