Detección de caras en imágenes en tiempo real

Alexandre Paz Mena

Departamento de Inteligencia Artificial

U.N.E.D.

Tesis presentada para el

Master en Inteligencia Artificial Avanzada

y dirigida por

Margarita Bachiller Mayoral

Febrero de 2012

Resumen

La detección de caras en imágenes es un área de estudio sobre la que se han realizado múltiples investigaciones. De entre ellas, destaca notablemente la realizada por Viola y Jones. Dicha investigación presentó un algortimo que resultó ser muy interesante por la alta tasa de detección que proporcionaba y la velocidad a la que era capaz de procesar las imágenes. Se puede entender la revolución que causó comprobando que muchas técnicas actuales de visión artificial utilizan algunas de las técnicas que en dicho artículo se aplicaron al ámbito por primera vez. Sin embargo, las nuevas técnicas se han centrado en mejorar la eficacia de la clasificación dejando de lado la velocidad , siendo los últimos algoritmos prácticamente incapaces de trabajar en tiempo real.

En esta investigación se han implementado cuatro técnicas de detección de caras consideradas importantes en el ámbito para analizarlas y buscar formas de mejorar su velocidad. Con cada una de las técnicas se ha generado un clasificador y se ha realizado un análisis en profundidad de cada uno de los clasificadores para conocer como funcionan, que información utilizan y cuanto tardan en cada una de sus etapas. A partir de dicha información se han propuesto tres nuevos clasificadores que intentan mejorar a los antetiores. De entre dichas propuestas destaca la que utiliza gradientes de intensidad optimizados, al conseguir reducir a la mitad el tiempo utilizado por el clasificador generado con la técnica de Viola y Jones.

Contenido

Listado de figuras

LISTADO DE FIGURAS

LISTADO DE FIGURAS

Listado de tablas

Glosario

DSP Microprocesador optimizado para ejecutar operaciones necesarias para el procesamiento de señales digitales. Del inglés "Procesador de Señales Digitales".

Eigenvector Vector matemático de una matriz, que al ser múltiplicado por dicha matriz, sigue siendo paralelo al vector original

Frame Cada una de las imágenes discretas de una secuencia de video.

HOG Histograma que agrupa gradientes según su orientación, no según su intensidad.

K-Mean Algoritmo de clasificación que genera "k" grupos a partir de un conjunto de muestras.

LBP Estructura de datos que almacena la información alrededor de un punto de forma binaria. Del inglés "patrón binario local".

Megapixel Conjunto de un millón de píxeles. Medida utilizada para comparar cámaras de fotos digitales.

MHz Medida de frecuencia de millones de veces por segundos. Se suele utilizar para medir la velocidad de los procesadores de computadores.

Píxel Cada uno de los puntos que forma una imagen digitalizada. Una imagen se representa como una matriz de píxeles.

Red neuronal Algoritmo que permite generar, de forma supervisada, un clasificador. Su comportamiento está basado en el funcionamiento de las neuronas biológicas.

RGB Modelo de representación de colores. Del inglés "rojo", "verde" y "azúl".

ROC Representación gráfica de la sensibilidad respecto a la tasa de falsos positivos. Del inglés "Característica Operativa del Receptor".

SVM Algoritmo que permite generar, de forma supervisada, un clasificador. Del inglés "máquina de vectores de soporte".

YUV Modelo de representación de colores, que utiliza un índice para la luminosidad y dos para los colores.

Capítulo 1

Introducción

Este trabajo fin de master se engloba dentro del proyecto de investigación AVISA desarrollado en el departamento de Inteligencia Artificial de la Escuela Técnica Superior de Ingeniería Informática de la Universidad Nacional de Educación a Distancia. El objetivo global de este proyecto de investigación es modelar e implementar un conjunto de componentes reutilizables (agentes) en la síntesis de tareas de vigilancia y seguridad semiautomáticas, válidos en distintos escenarios, tanto interiores como exteriores, en los que hay personas, vehículos y otros objetos en movimiento.

El trabajo de investigación se encuentra integrado en los desarrollos realizados para este proyecto de investigación y están dedicados a la detección de humanos. La calidad de los resultados obtenidos en la detección de los humanos presentes en la imagen, es fundamental para poder realizar un diagnóstico preciso sobre la situación. Cuanta más información se disponga de la cantidad de humanos presentes en la imagen así como de dónde se encuentran situados, más fiable será su identificación y descripción y, en consecuencia, tendremos más garantía de éxito durante la fase de descripción de la escena.

En escenas reales es habitual encontrar un conjunto de humanos andando juntos esto es lo que se denomina multitud. En un momento dado los humanos integrados en la multitud se pueden separar de ésta, por ello es importante en cualquier aplicación de tracking conocer el número de humanos que integran la multitud en cada imagen de la secuencia.

Además, las salidas producidas por los algoritmos de segmentación, especialmente si trabajamos con escenas reales, contienen generalmente ruido. Las causas del ruido

son debidas, principalmente, al ruido intrínseco de la cámara, a reflejos indeseados, a objetos que presentan un color que coincide con el fondo total o parcialmente y a la existencia de sombras y cambios artificiales o naturales en la iluminación. Estos factores pueden producir que áreas que no pertenecen a los objetos en movimiento sean incorporadas a éstos o que áreas, que pertenecen a los objetos no se detecten durante la segmentación. Por ello, también desde este punto de vista resultaría muy interesante disponer de un módulo capaz de determinar si un blob es parte de un humano o no.

Si analizamos el cuerpo humano en movimiento, una parte de éste que está sujeta a un cambio mínimo en su forma dependiendo del punto de vista es la cabeza. Por este motivo, este trabajo se centra en la búsqueda de algoritmos para la detección de caras en la imagen. La utilidad de este módulo es mucho mayor ya que existen diversas aplicaciones dónde es importante disponer de una detector de cabezas tales como contar los asistentes a un evento, tracking de humanos, detección del objetivo para realizar un enfoque automático en cámaras digitales etc.

Han sido muchos los estudios dedicados a la detección de humanos en una imagen, y más concretamente a la detección de caras pero una de los métodos que cabe destacar es el trabajo realizado por Viola y Jones [27] . Los algoritmos existentes hasta la presentación de este artículo, eran lentos y con tasas de detección bajas. Sin embargo, el algoritmo de Viola and Jones resultó ser muy interesante por la alta tasa de detección que proporcionaba y la velocidad a la que era capaz de procesar las imágenes. Se puede entender la revolución que causó comprobando que muchas técnicas actuales de visión artificial utilizan algunas de las técnicas que aparecieron en dicho artículo por primera vez.

Por ello, el estudio realizado en este trabajo ha comenzado con un análisis en profundidad de dicho algoritmo. Esto nos llevó a realizar una serie de mejoras respecto al tiempo de ejecución, característica muy importante cuando se trabaja en tiempo real.

1.1. Motivación

Al analizar el algoritmo de Viola y Jones, así como otros derivados del mismo, se observó que algo en común en todos los estudios derivados es la utilización, prácticamente idéntica, de las características propuestas en el artículo original. Dichas características, que se explican más detalladamente en la sección 3.1, consisten en dividir zonas de

la imagen en rectángulos con el mismo área y, a continuación, comparar la intensidad acumulada de dichos rectángulos.

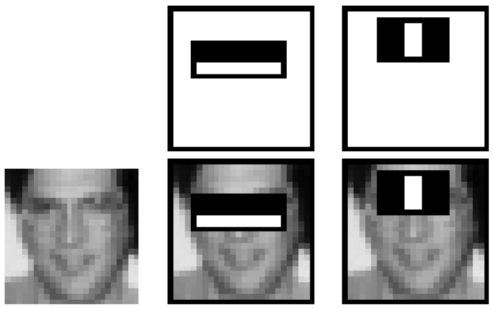


Figura 1.1: Ejemplo de características Haar.

En varios de los artículos analizados se proponen nuevas características para añadir a las originales. Por ejemplo, en [14] se añade una característica de forma "central" que utiliza la misma información que las originales y en el articulo [29] modifican las características para que comparen las mismas zonas en diferentes fotogramas.

El hecho de añadir características provoca que el detector pueda codificar una cara con más precisión al tener un mayor repertorio de herramientas. Sin embargo, se tienen que tener en cuenta las implicaciones de añadir nuevas características. Primero, nuevas características implican un mayor tiempo de entrenamiento. Partiendo del supuesto que en el artículo original tardan una semana en entrenar el clasificador, duplicar el número de clasificadores significa tardar dos semanas bajo las mismas condiciones. Segundo, y a modo de ejemplo, en el artículo [13] se presentan características que, en vez de utilizar la intensidad de la imagen, utilizan el gradiente de intensidad. Para ello el clasificador no solo necesita la información de intensidad sino que tiene que generar la información de gradiente, lo que conlleva un tiempo considerable, llegando a ser 8 veces más.

De todas formas, dada la información básica, la intesidad de la imagen, el número de características probadas sigue siendo insignificante comparado con el total de características posibles.

A la hora de buscar nuevas características, algo común a todas las investigaciones es la ausencia de utilización de información de color. Esto se debe principalmente a que una cara presenta un color uniforme, por lo que no tiene sentido comparar entre zonas

para buscar diferencias de color. Solo tendría sentido utilizar dicha información en un paso previo que indique si dado un color la imagen analizada podría ser una cara o no.

Por último, algo que se hecha en falta en todos los artículos es un análisis del resultado de utilizar cada una de las características. Como se puede comprobar en la sección 3.2, el añadir características con formas diferentes no implica que exista una mejora sustancial del rendimiento. Sin embargo, creemos que pararse a analizar los resultados obtenidos con un clasificador nos permitirían generar nuevos clasificadores con un nivel de eficacia mucho mayor.

1.2. Objetivos

El objetivo principal de este trabajo es abordar el problema de la detección de caras en imágenes 2D para el desarrollo de un módulo capaz de detectar las caras presentes en una imagen en el menor tiempo posible dado que su aplicación es en tareas de vigilancia. Para su resolución se van a realizar los siguiente objetivos parciales:

1. **Documentación bibliográfica**: Esto supone la recopilación de texto, artículos y referencias en general dedicados a la detección de caras en imágenes 2D. Este proceso nos llevará a seleccionar aquellos trabajos con mayor relevancia por sus resultados.

2. **Creación de una base de imágenes**: Esto implica la recopilación de imágenes para poder realizar los distintos experimentos.

3. **Análisis de los algoritmos seleccionados**: Se va a realizar un estudio, en profundidad, de los algoritmos para su implementación. Además, este estudio nos permitirá detectar posibles mejoras.

4. **Evaluación de resultados**: Se va a realizar una comparativa de los resultados de los diferentes métodos seleccionados.

5. **Propuesta para mejorar alguno de los algoritmos seleccionados**: Se va a intentar mejorar alguno de los algoritmos seleccionados desde el punto de vista de tiempo de ejecución, lo cual resulta de vital importancia en videovigilancia.

6. **Evaluación de resultados de la propuesta de mejora**: Se va a realizar una comparativa de las propuestas y los algoritmos originales para comprobar la eficacia de las mismas.

1.3. Estructura de la memoria

La memoria de la investigación que presentamos en este documento ha sido organizada en cinco capítulos y un apéndice. A continuación describimos brevemente el contenido de cada una de estas partes:

- **Capítulo 1 - Introducción**: este capítulo está orientado a situar al lector en el ámbito científico en el cual se ubica este trabajo. Además, se ha plantado la problématica que ha motivado el desarrollo de la investigación y los objetivos perseguidos.
- **Capítulo 2 - El problema de la detección de caras en imágenes**: En el segundo capítulo, por un lado, se realiza un estudio de las principales investigaciones llevadas a cabo dentro del ámbito de la detección de caras automatizada. Por otro lado, también se presentan las bases de datos de muestras que se van a utilizar en la investigación y las curvas ROC, una técnica que permite comparar clasificadores.
- **Capítulo 3 - Estrategias analizadas para la detección de caras**: En el tercer capítulo se analizan cuatro de las principales técnicas del ámbito. Además, se analizan en profundidad los resultados obtenidos con cada una de las técnicas para intentar mejorarlas.
- **Capítulo 4 - Estrategia propuesta para mejorar la detección de caras**: En el cuarto capítulo se proponen una serie de nuevas técnicas centradas en mejorar la velocidad de las investigaciones elegidas. También se comparan los resultados de las nuevas propuestas con los de las técnicas analizadas.
- **Capítulo 5 - Conclusiones y futuros trabajos de investigación**: En el último capítulo se exponen, a modo de resumen, las conclusiones obtenidas como resultado de la investigación y se proponen nuevas líneas de investigación que darán continuidad a este trabajo de fin de master.

- **Bibliografía**: En esta parte se pueden encontrar las referencias a todas aquellas investigaciones que se han utilizado para sustentar la actual.
- **Anexo A - Optimizaciones al algoritmo AdaBoost**: Este anexo contiene una serie de optimizaciones que se han realizado sobre el algoritmo AdaBoost para facilitar el entrenamiento de múltiples clasificadores.

Capítulo 2

El problema de la detección de caras en imágenes

2.1. Estado del arte

Para la resolución del problema de la detección de caras se han propuesto numerosas y diferentes técnicas a lo largo de toda la historia de la computación. Por ello, y para poder conocer qué técnicas se consideran las mejores, hemos partido de una serie de informes que permiten conocer, de forma superficial, el estado del arte de dicho problema. Los artículos [10] y [34] presentan los estudios más relevantes hasta el año 2002. La mayoría de esos estudios plantean el problema como una clasificación entre "caras" y "no caras" y para solucionarlo plantean utilizar redes neuronales, redes de Markov o SVMs. Por otro lado, en el artículo [35] se presentan las técnicas más conocidas hasta el año 2010, con la novedad del cambio de concepto desde un problema de "clasificación" a un problema de "detección de eventos poco comunes". El artículo que cambió el paradigma fue el presentado por Viola y Jones en [27], a partir de ese artículo la mayoría de las técnicas desarrolladas fueron derivadas de la presentada por esos autores.

Para mostrar la eficacia de las diferentes técnicas presentadas se compararán los resultados con el algoritmo de Viola y Jones, por ser éste el más representativo de todas. Lo idóneo sería utilizar curvas ROC para comparar clasificadores, que se explicarán en la sección 2.3. Sin embargo, la cantidad de figuras necesarias no aportaría mucha información, por lo que únicamente se compararán valores simples de la tasa de detección,

la tasa de falsos positivos y de velocidad:

- Tasa de detección: número de detecciones correctas entre el total de muestras positivas.
- Tasa de falsos positivos: número de detecciones incorrectas entre el total de muestras negativas.
- Velocidad: tiempo que tarda el algoritmo en analizar una imagen de 320 por 240 píxeles en un procesador a 700Mhz.

A continuación se hace un resumen, ordenado cronológicamente, de aquellas técnicas consideradas las más relevantes.

En [24] aparece la que está considerada como la primera técnica que permitió de forma eficaz la detección y el reconocimiento de caras, las "eigenfaces". Las "eigenfaces" son modelos que representan la cara media de un conjunto de caras. Con dichos modelos se puede llegar a definir una cara de dos formas. Por un lado, a partir de varias "eigenfaces" se puede definir una cara como, por ejemplo, un 33 % del primer modelo, un 50 % del segundo y un 17 % del tercero. Por otro lado, se pueden representar las diferencias de una cara respecto a los principales eigenvectores de la cara modelo, tal como se puede ver en la figura 2.1. La utilidad de está técnica reside en la reducción de características. La imagen de una cara puede llegar a estar formada por 10.000 puntos, aplicando la segunda opción se reducirían a unas 100 o 150 características y con la primera a menos de una decena. En este artículo no intentan detectar caras, sino que calculan el error acumulado de codificar una cara con "eigenvectores", que puede llegar a ser inferior al 5 %.

Figura 2.1: Utilización de "eigenfaces": la primera imagen es una "eigenface", la segunda una cara real y la tercera representa la diferencia de la cara al modelo.

Siguiendo la misma línea, en [25] se presenta una técnica que utiliza redes neuronales para realizar la detección. Primero, los autores generaron un modelo de distribución

de caras utilizando un algoritmo similar al "k-mean", el cual permite dividir una serie de datos en bloques obteniendo el punto medio de cada uno de los bloques. El modelo generado consistía en 6 imágenes con los valores medios de las caras de cada bloque, visibles en la figura 2.2, de forma similar a las "eigenfaces", y otros 6 modelos de no caras. A partir de dicho modelo, entrenaron un clasificador basado en redes neuronales que utilizaba las diferencias entre la imagen a clasificar y los diferentes centros, tanto de los modelos de caras como de los no-caras. El entrenamiento se realizó con imágenes de 19 por 19 puntos y si se querían analizar imágenes de tamaños diferentes primero se redimensionaban antes de pasarlas por el clasificador. Este algoritmo no utiliza la misma base de datos de pruebas que Viola y tampoco indica el número de ventanas de detección analizadas, por lo que es imposible calcular la tasa de falsos negativos para la comparación.

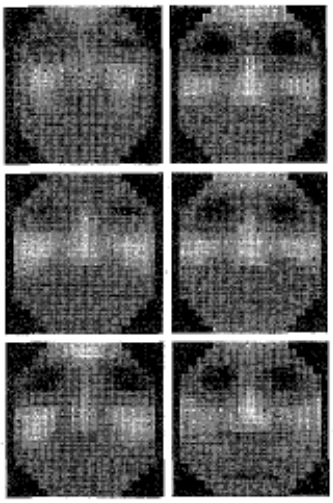


Figura 2.2: Centroides de la técnica propuesta en [25].

En el artículo [17], además de "eigenfaces", utilizan una nueva técnica denominada "correspondencia de plantillas". Esa técnica permite comparar dos zonas del mismo tamaño para comprobar si contienen la misma imagen. Básicamente, consiste en acumular las diferencias al cuadrado entre los puntos situados en la misma posición en dos imágenes, para todos los puntos. A menor diferencia acumulada, más parecidas son las imágenes. Volviendo al artículo [17], los autores plantean comparar las muestras con diferentes modelos de "eigenfaces" y aplicar una distribución gausiana para indicar la probabilidad de que la muestra sea una cara o no. Aunque no utilice la misma base de

datos que Viola y Jones una comparación de ambos algoritmos se puede ver en la tabla 2.1.

Artículo	Detección	Falsos positivos	Velocidad
[17]	80 %	1 %	Sin datos
[27]	80 %	2×10^{-5} %	0,067s

Tabla 2.1: Comparación de rendimiento entre los artículos [17] y [27].

En [20] cambian de técnica y utilizan una serie de redes neuronales para detectar caras en porciones de imágenes. La primera red neuronal asume que la imagen es una cara y estima la rotación de la misma. Esta información sirve para corregir la rotación de la imágen y aplicar el resto del algoritmo a una imagen con una orientación vertical fija. Para decidir finalmente si la ventana analizada contiene una cara o no, se pondera el resultado de varias redes neuronales. Dichas redes están entrenadas utilizando la intensidad de cada punto de una ventana de 20 por 20 puntos. Además, en este artículo se utiliza una novedosa técnica denominada "bootstrap", presentada por primera vez en [26], que consiste en emplear imágenes que no contengan caras, como la que se puede ver en la figura 2.3, como fuentes de muestras negativas de entrenamiento. Gracias a esta técnica se pueden generar, de forma automática, decenas de millones de muestras negativas para realizar entrenamientos. Dada su importancia, una explicación más detallada de esta técnica se puede ver en la sección 2.2. En este artículo los autores presentan resultados para diferentes configuraciones. Una comparación de la mejor configuración y el algoritmo de Viola y Jones se puede ver en la tabla 2.2.

Artículo	Detección	Falsos positivos	Velocidad
[20]	90,3 %	$2,2 \times 10^{-4}$ %	0,57s
[27]	90 %	$4,6 \times 10^{-5}$ %	0,067s

Tabla 2.2: Comparación de rendimiento entre los artículos [20] y [27].

En [33] se utiliza una combinación de características simples para detectar caras. Cada una de dichas características simples se basa en comprobar que la intensidad de un punto concreto de la ventana de detección, de 20 por 20 puntos, no supere un umbral dado. La polaridad de la característica indica si el umbral se tiene que superar por encima o por debajo. Además, cada característca tiene asignado un peso. Si el peso acumulado de las características activas supera el umbral del clasificador final,

Figura 2.3: Muestra negativa obtenida de una imagen.

se considera que una cara está presente. Tanto los pesos como los diferentes umbrales se calculan durante el entrenamiento. Una comparación de capacidad de detección se puede ver en la figura 2.3. Sin embargo, no se menciona ni el número de elementos analizados ni la velocidad del algoritmo.

Artículo	Detección	Falsos positivos	Velocidad
[33]	94,8 %	78	Sin datos
[27]	94 %	120	0,067s

Tabla 2.3: Comparación de rendimiento entre los artículos [33] y [27].

Otra técnica diferente se presenta en [23], donde el objetivo es detectar caras, caras de perfil y coches. Para lograrlo proponen utilizar patrones con diferentes probabilidades de aparación. Cada patrón es un histograma de frecuencias de diferentes tipos de características como Wavelets, gradientes o la intensidad. La ventaja de los histogramas de frecuencias es que se pueden aplicar independientemente del tamaño de la sección, haciendo al detector independiente del tamaño de la imágen. No utilizan la misma base de datos que Viola y Jones, ni proporcionan información del número de muestras negativas ni de la velocidad. Sin embargo, una comparación aproximada se puede ver en la tabla 2.4.

Artículo	Detección	Falsos positivos	Velocidad
[23]	92,7 %	700	Sin datos
[27]	92,5 %	80	0,067s

Tabla 2.4: Comparación de rendimiento entre los artículos [23] y [27].

En [1] plantean un clasificador que, una vez detectadas las caras, sea capaz de discriminar entre caras frontales, de perfil y con posiciones intermedias. Para ello utilizan AdaBoost, un algoritmo que genera un clasificador fuerte a partir de un conjunto de

clasificadores simples. Este algoritmo se presentó por primera vez en [7] y se ha convertido en uno de los algoritmos básicos para la detección de caras. Dada su importancia, una explicación más detallada del mismo se puede ver en la sección 3.1. Los clasificadores simples con los que se alimenta a AdaBoost para generar el clasificador final son comparaciones entre la intensidad de puntos sueltos de la imagen.

En el artículo [27] Viola y Jones presentaron un algoritmo que cambió radicalmente el paradigma de detección de caras y se convirtió en la base de las investigaciones que se realizaron en años siguientes. Por ese motivo, se ha explicado más detalladamente en la sección 3.1. De forma resumida, en dicho artículo se cambia el estilo de trabajo, pasando de un problema de clasificación entre caras y no caras a un problema de detección de eventos poco frecuentes. Este cambio se planteó debido a que en una imagen las caras suelen ser pocas comparadas con el total de ventanas de detección posibles. Para resolver dicho problema proponen utilizar un clasificador compuesto de una serie de clasificadores dispuestos en cascada. Capa capa de la cascada es más compleja que la anterior, lo que permite eliminar inicialmente las muestras más sencillas para que sean las últimas capas, las más complejas, las que analicen las muestras más difíciles. Cada capa se entrena utilizando una variante del algoritmo AdaBoost, que busca una tasa de detección objetivo en vez de buscar utilizar todas las características simples. Por otro lado, las características que utilizan en una imagen son, de forma simple, comparaciones entre la intensidad de zonas adyacentes entre sí, como se pueden ver en la figura 2.4. Dichas caracteríticas se las conoce comúnmente como características tipo "Haar", por su similitud con las onda Haar. Para calcular la intensidad de una zona utilizan una estructura de datos conocida como "imagen integral", que consiste en calcular el sumatorio de intensidades hasta un punto de la imagen. Con dichas características se puede ver que buscar caras en ventanas de detección de diferentes tamaños se vuelve trivial, simplemente teniendo que indicar al clasificador el nuevo tamaño de la zona de búsqueda.

Una de las grandes ventajas de este algoritmo es la velocidad, algo particularmente interesante para los sistemas en tiempo real. Hasta la aparición de este algoritmo, las técnicas presentadas eran pesadas y poco eficaces. Esta nueva técnica permitía, en su día, analizar imágenes de 384 por 288 píxeles en 0.067 segundos con un procesador a 700MHz, lejos de otras propuestas contemporáneas, como los 40 segundos de [23] o del segundo de [21].

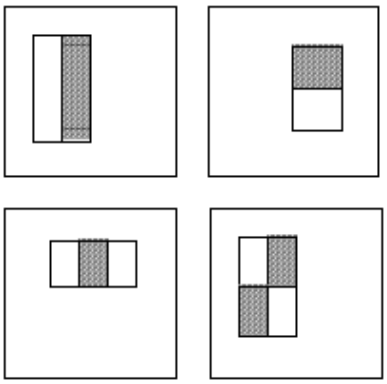


Figura 2.4: Ejemplos características tipo 'Haar' posicionadas en una ventana de detección.

Otro artículo de los mismos autores, [12], utiliza información de varios "frames" de un video para detectar personas en movimiento dentro de secuencias de imágenes. Para ello plantean nuevas características, similares a las planteadas en su artículo original, que son capaces de utilizar no solo información de la imagen actual sino que utilizan también información de la imagen anterior y de una serie de filtros de movimiento como los presentados en la figura 2.5;

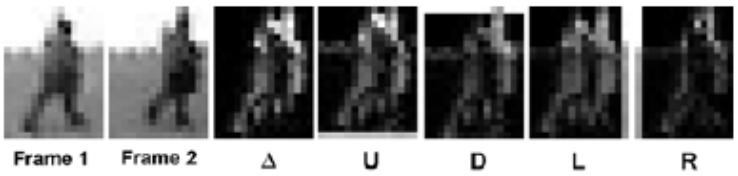


Figura 2.5: Dos frames utilizados en el clasificador y el resultado de los filtros posibles que evaluan el movimiento general (δ), ascendente (U), descendente (D), hacia la izquierda (L) y hacia la derecha (R).

Tal como se ha mencionado anteriormente, la aparición de la técnica de Viola y Jones motivó una serie de investigaciones con el objetivo de mejorar la capacidad de detección. Por ejemplo, en [14] se amplía la técnica de Viola y Jones añadiendo característica que son capaces de comparar zonas rectangulares con una rotación de 45º. Este cambio implica que, además de detectar líneas horizontales y verticales como el detector original, son capaces de detectar líneas diagonales. Este artículo también se ha convertido en uno de los más referenciados en el ámbito por la mejora de eficacia que proporciona a cambio de un entrenamiento más largo. Una comparación entre este

algoritmo y el de Viola y Jones lo realizan en el propio arículo y se puede ver en la tabla 2.5.

Artículo	Detección	Falsos positivos	Velocidad
[14]	92,5 %	0,2 %	0,57s
[27]	92,5 %	0,25 %	0,067s

Tabla 2.5: Comparación de rendimiento entre los artículos [14] y [27].

En el artículo [36] se utiliza un tipo de característica denominada MB-LBP, del inglés "Patrón Binario Local MultiBloque". Dichas características codifican una matriz binaria centrada en un punto determinado, en el que cada valor representa si el punto codificado tiene mayor o menor intensidad que el central. Un ejemplo más claro se puede ver en la imagen 2.6. Al igual que otros artículos mencionados, esta técnica presenta una capacidad de detección ligeramente mejor que el algoritmo de Viola y Jones. Sin embargo y según los autores, tarda alrededor de 0.1 segundos en procesar una imagen de 320 por 240 píxeles en un procesador de 3000MHz, unas 60 veces más lento. Este artículo también presenta una comparación con el algoritmo de Viola y Jones, tal como se puede ver en la tabla 2.6.

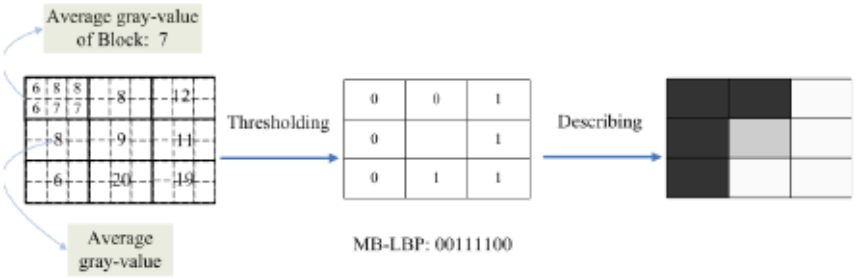


Figura 2.6: Generación de una característica MB-LBP.

Artículo	Detección	Falsos positivos	Velocidad
[36]	91,9 %	136	0,42s
[27]	91,8 %	167	0,067s

Tabla 2.6: Comparación de rendimiento entre los artículos [36] y [27].

Otra técnica que también utiliza características basadas en LBP se propone en [32]. A las nuevas características que plantean las denominan LABs, características binarias

enlazadas localmente, y las utilizan para comparar la intensidad de dos zonas de forma binaria. Las características originales tipo Haar, por el contrario, devuelven el valor de la diferencia de intensidad y es el clasificador simple el que tiene un umbral para tomar la decisión. Por otro lado, estas características LABs se combinan formando grupos. Algo novedoso de este artículo es que utilizan un dispositivo DSP para realizar las operaciones. Este tipo de procesadores están especialmente diseñados para realizar operaciones con señales digitales. En concreto, los autores indican que consiguen procesar una imagen de 320 por 240 píxeles en 30ms con un dispositivo capaz de trabajar a 600MHz, la mitad del tiempo que necesita la técnica de Viola y Jones. Un estudio interesante para el futuro sería probar la eficacia de diferentes técnicas con este tipo de dispositivos para poder comparar correctamente la velocidad de procesamiento de cada una de ellas. Por terminar con este artículo, una comparación con la técnica de Viola y Jones se puede ver en la tabla 2.7.

Artículo	Detección	Falsos positivos	Velocidad
[32]	92 %	10	0,025s (DSP)
[27]	92 %	50	0,067s

Tabla 2.7: Comparación de rendimiento entre los artículos [32] y [27].

La agrupación de características también aparece en [16], donde añaden un nuevo tipo de características que son, básicamente, agrupaciones de características simples. Los autores plantean dichos agrupamientos para añadir robusted al forzar la aparición simultánea de varias características simples. Un ejemplo de estos agrupamientos se puede ver en la figura 2.7. Este artículo no indica claramente como se han realizado las pruebas por lo que no se puede comparar correctamente.

Otra modificación se propone en [18], donde los autores plantean modificar el algoritmo de entrenamiento del clasificador. Tal como se explica en la sección 3.1, cada capa del clasificador de Viola y Jones se compone de una serie de clasificadores débiles. Cada iteración del entrenamiento añade un nuevo clasificador a la última capa y, para ello, todos y cada uno de los clasificadores débiles planteados tiene que ser entrenado para obtener el valor umbral que necesitan, siendo este entrenamiento uno de los mayores cuellos de botella del algoritmo. Para evitarlo, en el artículo proponen un nuevo algoritmo de entrenamiento que consiste en seleccionar los clasificadores débiles en base a estadísticas de las muestras de entrenamiento. A costa de un pequeño descenso en

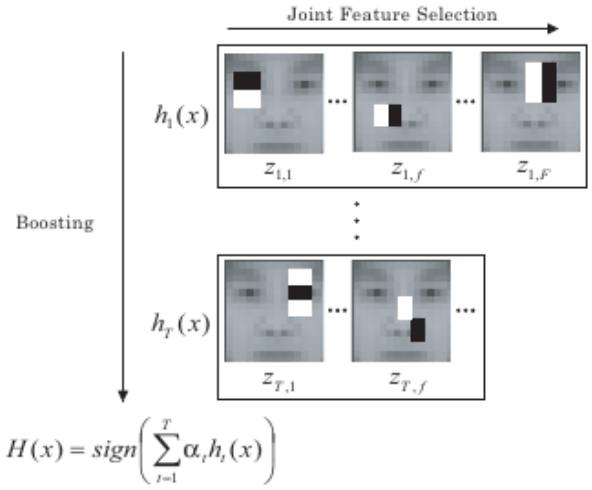


Figura 2.7: Parte del entrenamiento del algoritmo propuesto en [16].

la eficacia se consigue pasar de una complejidad $O(NT \log N)$ en el entrenamiento a $O(Nd^2 + T)$. En la tabla 2.8 se puede ver una comparación con el algoritmo de Viola según los propios autores.

Artículo	Detección	Falsos positivos	Velocidad
[18]	85 %	37	No hay datos
[27]	85 %	44	0,067s

Tabla 2.8: Comparación de rendimiento entre los artículos [18] y [27].

En el artículo [15] los autores señalan que uno de los problemas del algoritmo AdaBoost es la no convergencia en determinadas situaciones. Es decir, puede darse un caso en el que, al añadir nuevos clasificadores débiles al conjunto que se va generando, la eficacia del mismo baje. Lo que proponen es un nuevo algoritmo de entrenamiento, llamado "KLBoosting", que ajusta los coeficientes de los diferentes clasificadores débiles de forma que garantiza que la eficacia del clasificador aumenta con cada clasificador débil. Una comparación con el algoritmo de Viola y Jones según los propios autores se puede ver en la tabla 2.9.

Otra investigación muy interesante es la presentada en [30]. En dicho artículo se plantean modificaciones a casi todos los puntos de la técnica original. Primero, cambian el algoritmo de entrenamiento a Real Adaboost, un algoritmo presentado en [22] que, en vez de utilizar valores discretos de positivo o negativo, utiliza valores reales entre

Artículo	Detección	Falsos positivos	Velocidad
[15]	85 %	2×10^{-5} %	1,02s
[27]	85 %	2×10^{-5} %	0,067s

Tabla 2.9: Comparación de rendimiento entre los artículos [15] y [27].

0 y 1 que representan la confianza del clasificador. El siguiente cambio es utilizar un nuevo tipo de clasificador débil que sirva para Real Adaboost. Dicho clasificador, al que denominan LUT, devuelve la confianza de que una determinada muestra sea positiva. Por último, modifican el clasificador en cascada de forma que cada capa forme parte de la siguiente, tal como se puede ver en la figura 2.8. Esto permite que la confianza de una capa se pueda trasladar a la siguiente. Una comparación con el algoritmo de Viola y Jones según los propios autores se puede ver en la tabla 2.10.

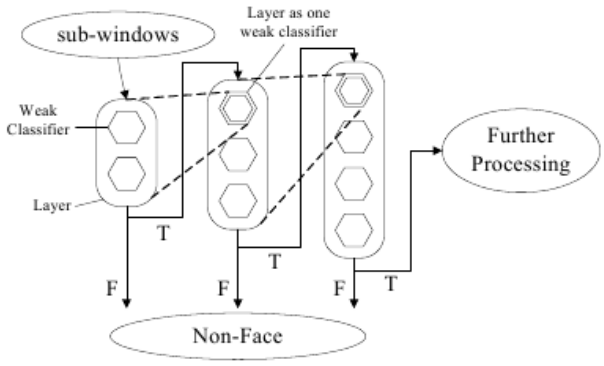


Figura 2.8: Clasificador en cascada en el que cada capa forma parte de la siguiente.

Artículo	Detección	Falsos positivos	Velocidad
[30]	90 %	$0,07 \times 10^{-6}$ %	0,274s
[27]	90 %	$0,5 \times 10^{-6}$ %	0,067s

Tabla 2.10: Comparación de rendimiento entre los artículos [15] y [27].

Una alternativa diferente al planteamiento de Viola y Jones se propone en [9], donde utilizan un clasificador basado en una jerarquía de SVMs. La ventaja de dicha jerarquía es que los niveles más bajos realizan un primer paso que elimina grandes partes del fondo de la imagen, mientras que los niveles más altos se dedican a detectar las caras, por lo que son más complejos y tardan más en procesar las distintas partes. El entrenamiento consiste en dos pasos que son, primero, generar la jerarquía de clasificadores y, segundo,

realizar un repaso utilizando un conjunto de prueba. El repaso sirve para eliminar aquellos clasificadores que menos aportan a la clasificación, logrando un incremento significativo en velocidad respecto a otros clasificadores monolíticos basados en SVM. Aunque en el artículo no aparece una comparación con el algoritmo de Viola y Jones, una aproximación se puede ver en la tabla 2.11.

Artículo	Detección	Falsos positivos	Velocidad
[9]	80 %	$2,5 \times 10^{-3}$ %	0,666s
[27]	80 %	$1,06 \times 10^{-4}$ %	0,067s

Tabla 2.11: Comparación de rendimiento entre los artículos [9] y [27].

En el artículo [11] se plantea la cuestión de que los algoritmos de clasificación, como AdaBoost, SVM o redes neuronales, están planteados de forma "offline", es decir, una vez entrenados no modifican su comportamiento. Los autores proponen un nuevo algoritmo basado en RealAdaboost que actualiza el clasificador en base a la nueva información obtenida al clasificar nuevas muestras. Este planteamiento "online" está dirigido a evitar los problemas de otros clasificadores con situaciones de luminosidad extremas, como las que se pueden ver en la figura 2.9. Aunque este artículo no presenta una comparación con el algoritmo de Viola y Jones, una aproximación se puede ver en la tabla 2.12.

Figura 2.9: Situaciones extremas que se plantean evitar en el artículo [11].

Artículo	Detección	Falsos positivos	Velocidad
[11]	95 %	0	Sin datos
[27]	95 %	120	0,067s

Tabla 2.12: Comparación de rendimiento entre los artículos [11] y [27].

Por otro lado, y dejando aparte los diferentes derivados del algoritmo de Viola y

Jones, en los últimos años han ido surgiendo técnicas de reconocimiento de objetos que se pueden aplicar al ámbito de la detección de caras. Por ejemplo, en [5] se intenta realizar un detector de personas completas. Para ello, los autores proponen un clasificador basado en SVMs que utilice HOGS, histogramas de gradientes orientados. La técnica es perfectamente aplicable al ámbito de la detección de caras y comparte características con la presentada en [13], donde se propone utilizar los HOGS mencionados anteriormente en un clasificador con la misma aquitectura que el de Viola y Jones. Aunque no se utilice para detectar caras, sino para detectar personas completas, los autores indican que el clasificador generado tarda algo menos de 1 segundo en procesar una imagen de 320 por 240 píxeles. Ese tiempo es similar al resto de técnicas contemporáneas, pero notablemente superior al de la técnica de Viola y Jones.

2.2. Imágenes utilizadas

2.2.1. Muestras positivas de entrenamiento

A la hora de generar cualquier tipo de clasificador, uno de los elemenos más importantes, son las muestras, tanto positivas como negativas. En nuestro caso dichas muestras corresponden a caras, por lo que habrá que obtener un número suficiente como para realizar el entrenamiento correctamente.

En los artículos analizados se utilizan un número bastante variable de muestras positivas, desde 100 en [13] hasta las 5.000 que se mencionan en el articulo original de Viola y Jones. Un detalle a tener en cuenta es que el número de muestras está duplicado gracias a la utilización de imágenes espejadas. Es decir, por cada cara de muestra se puede obtener otra utilizando una versión volteada horizontalmente de la misma.

Conviene mencionar que el algoritmo AdaBoost utiliza una técnica llamada Validación Cruzada. Esta técnica divide el total de imágenes en dos grupos, uno para entrenar el clasificador fuerte y otro para comprobar la eficacia del mismo, por lo que el número de muestras realmente utilizado en el entrenamiento es la mitad de los seleccionados.

Un método muy sencillo para disponer de un conjunto de caras consiste en utilizar bases de datos existentes generadas para estudios anteriores. Para elegir entre las diferentes bases de datos posibles, se ha partido del artículo [8], donde aparece un listado de bases de datos de caras disponibles de forma pública.

En particular, en este trabajo nos vamos a centrar en la base de datos FERET [19], dado que tiene una cantidad razonable de muestras y resulta fácil de obtener. Dicha base de datos contiene fotografías, tanto en blanco y negro como en color, de 1.200 individuos. Además, muchas de las fotografías están repetidas en diferentes condiciones de luz y en distintos periodos de tiempo, esto último para reflejar los distintos cambios que se pueden dar en la cara con el paso del tiempo. Dos ejemplos de caras incluidas en esta base de datos se muestran en la figura 2.10, donde se percibe la variedad de casos que incorpora. En la primera la cara del individuo está de frente gesticulando y la segunda la cara está mal iluminada.

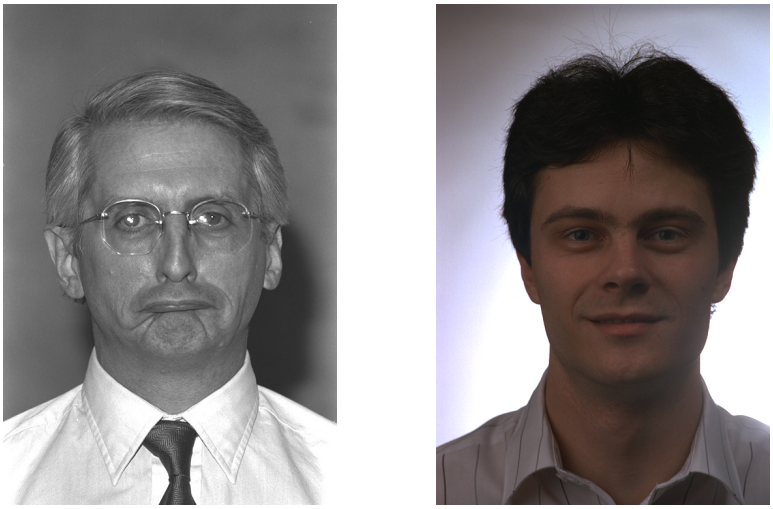


Figura 2.10: Muestras de la base de datos ColorFeret.

Un problema de esas bases de datos es que contienen la información de la forma más genérica posible, incluyendo fotografías completas y con la mayor resolución posible. Eso implica que los datos no están en el mismo formato del que se puede alimentar el clasificador. Por ejemplo, el entrenamiento planteado por Viola y Jones utiliza una sección cuadrada de la cara centrada sobre la nariz e incluyedo las cejas y la boca. Esta sección se puede considerar la más uniforme de la cara, ya que es fácilmente definible y, a excepción de las gafas, no suele tener elementos variables como el pelo. Conseguir la sección cuadrada utilizada por Viola & Jones requiere un trabajo manual de selección y corte muy costoso en tiempo.. Un ejemplo de secciones cuadradas de caras se muestra en la figura 2.11.

Figura 2.11: Ejemplo de caras utilizadas para generar el clasificador.

2.2.2. Muestras negativas de entrenamiento

Un punto importante del clasificador es la capacidad de evitar falsos positivos. Si se pretende conseguir una tasa de falsos positivos inferior a 10^{-6} se necesitan, como mínimo, 10^6 muestras negativas.

Figura 2.12: Ejemplo de imagen sin caras y una sección de la misma.

Una técnica muy eficaz para generar grandes cantidades de muestras negativas es el "bootstraping", mencionado por primera vez en [26]. Dicha técnica consiste en partir de imágenes de gran tamaño, en las que está garantizada la no presencia de caras, e ir tomando aleatoriamente diferentes trozos como muestras negativas. Un ejemplo de como se pueden obtener muestras negativas se puede ver en la figura 2.12.

Supongamos que tenemos una fotogradía de 4752 por 3168 puntos, el equivalente a una cámara de 15 megapíxeles, en la que está garantizado que no aparecen caras. Si se toman todas las posibles secciones de 24 por 24, el mismo tamaño que las muestras positivas, se generan alrededor de 15 millones de muestras negativas. Si además añadi-

mos secciones de diferentes tamaños y luego los redimensionamos, el total aumenta en varios órdenes de magnitud. Esta cantidad es más que suficiente para alcanzar la tasa de falsos positivos anteriormente mencionada.

2.2.3. Imágenes de prueba para comparar clasificadores

Para probar los clasificadores que se van a generar en este estudio, se ha decidido utilizar la base de datos conocida como CMU+MIT y que fue recogida para el artículo [21]. Esta base de datos consiste en fotografías reales en las que aparece gente, tal como que se puede ver en la figura 2.13. Se ha seleccionado esta base de datos por ser la utilizada por la mayoría de investigadores.

Figura 2.13: Ejemplo de imagen de la base de datos CMU+MIT.

A diferencia de las muestras de entrenamiento, estas imágenes no consituyen muestras positivas o negativas, sino que son ejemplos reales de imágenes que puede tomar una cámara. Por ello, el clasificador tiene que recorrer la imagen con ventanas de diferentes tamaños para localizar las caras.

Por ejemplo, el clasificador de Viola y Jones utiliza muestras de entrenamiento de 24 por 24 píxeles. Utilizando ese tamaño como mínimo de la ventana, con diferencias entre tamaños de 12 puntos y teniendo en cuenta que se analiza comenzando por el tamaño más grande para poder descartar zonas, para una imagen de 320 por 240 tendríamos los siguientes tamaños de ventana: 240, 228, 216, 204, 192, 180, 168, 156, 144, 132, 120, 108, 96, 84, 72, 60, 48, 36, 24.

Un detalle que afecta al rendimiento de los clasificadores es la gestión por parte de los mismos del tamaño de la ventana. Por ejemplo, los clasificadores de tipo Viola son independientes del tamaño porque parten de un cuadrado y puntos sobre dicho cuadrado, al escalar el cuadrado los puntos a los que acceden se modifican, a su vez, de forma propocional sin penalizar el rendimiento. Por otro lado, los clasificadores que no son independientes del tamaño necesitan reescalar la imagen antes de procesarla, con el coste computacional que supone.

2.2.4. Normalización de las imágenes

Para el correcto funcionamiento del clasificador, las zonas analizadas tienen que estar normalizadas para disminuir el efecto que puede tener la presencia de diferentes fuentes de luz. En este estudio se ha realizado la normalización respecto a la varianza, al igual que en el artículo original de Viola y Jones. La normalización consiste en, primero, calcular el valor medio y la varianza de la ventana de detección y, a continuación, por cada valor quitar el valor medio y dividir entre la varianza, tal como se puede ver en la ecuación 2.1.

$$x^{norm}(x,y) = \frac{x(x,y) - m}{\sigma} \tag{2.1}$$

siendo σ la desviación estándar, m la intensidad media de la zona a normalizar y x el valor de la intensidad en un punto. Dada la complejidad computacional que supone calcular constantemente la varianza para cada zona a analizar, se ha optado por utilizar la aproximación planteada en la fórmula 2.2.

$$\sigma^2 = m^2 - \frac{1}{N}\sum x^2 \tag{2.2}$$

La normalización de la imagen se puede realizar de una forma muy sencilla utilizando una imagen integral adicional que acumule los valores al cuadradado de cada píxel de la imágen. La imagen integral es una estructura de datos utilizada por Viola y Jones y que se explica en más detalle en la sección 3.1. Con todo ello, obtener la media y la varianza supone un total de 8 accesos a memoria.

2.3. Curva ROC

Una curva ROC, del inglés "Característica Operativa del Receptor", es una representación gráfica de la tasa de detección frente a la tasa de falsos positivos que presenta un clasificador binario de sensibilidad ajustable. La principal utilidad de esta curva es la comparación entre diferentes clasificadores. Por ejemplo, si un clasificador sólo proporcionase un valor para la tasa detección y un valor de tasa de falsos negativos, como los visibles en la tabla 2.13, las comparaciones resultarían problemáticas. En dicha tabla se puede ver que el clasificador A tiene mayor tasa de detección mientras que el clasificador B tiene una tasa de falsos positivos mucho menor. Con esa información resulta imposible decidir que clasificador funciona mejor.

	Detección	Falsos Positivos
Clasificador A	90 %	0,003 %
Clasificador B	50 %	0,0001 %

Tabla 2.13: Resultados de clasificadores de ejemplo.

Sin embargo, si se utilizan curvas ROC se puede ver claramente si un clasificador es mejor que otro. Por ejemplo, en la figura 2.14 se puede ver como, dada la misma tasa de detección, el clasificador B tiene menor tasa de falsos positivos que el clasificador A.

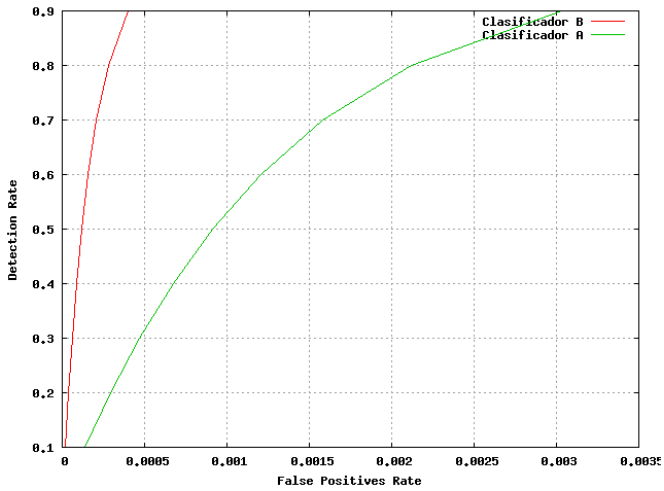


Figura 2.14: Ejemplo de comparación de clasificadores utilizando curvas ROC.

Visto eso, la creación de curvas ROC se vuelve de crucial importancia para poder comparar clasificadores o para poder analizar las mejoras realizadas en los mismos.

Las imágenes de prueba que se suelen utilizar para generar estas curvas son las mencionadas en la sección 2.2.3. Sin embargo, la generación de una curva ROC no es tan simple como contar cuantas caras se han detectado y cuantos falsos positivos ha habido. Al realizar las curvas ROC de los clasificadores entrenados en esta investigación se han visto una serie de problemas que requieren ser explicados con más detalle a continuación. Dichos problemas, como se verá más adelante, provocan que una curva ROC pueda ser generada con diferentes configuraciones, por lo tanto, solo se pueden comparar curvas entre sí en el caso de que haya sido creadas con las mismas opciones de configuración.

2.3.1. Curvas ROC con tasas de detección entre 0 y 1

Si una curva ROC avanza desde una tasa de detección de 0 hasta una tasa de 1, 100 % de detección, eso significa que el clasificador, ajustando la sensibilidad, es capaz de llegar a detectar todas las muestras positivas de la prueba. Sin embargo, si se tienen en cuenta muestras como las que se pueden ver en la figura 2.15, las cuales son muestras reales de la base de datos de prueba, es imposible que un detector entrenado únicamente con fotos reales sea capaz de detectarla. En particular, no nos interesa que nuestro clasificador sea capaz de detectar dibujos, por eso las muestras de entrenamiento solo incluyen imágenes reales.

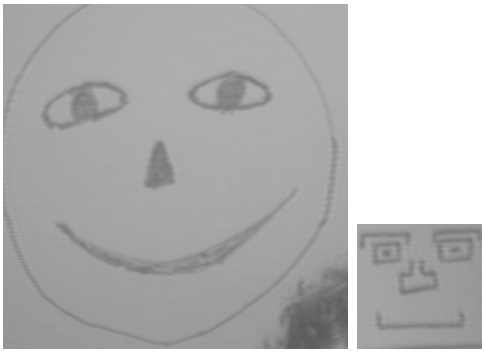


Figura 2.15: Caras dibujadas de la base de datos CMU+MIT.

Principalmente, hay que tener en cuenta que el clasificador Viola y Jones está compuesto por varias capas y la sensibilidad del clasificador solo afecta a la última. Una cara dibujada es muy probable que sea eliminada en las primeras capas del clasificador sin que el umbral establecido pueda afectar a tal decisión. Todo esto quiere decir que

una curva ROC que llegue al 100 % de detecciones, como la presentada en [31] y visible en la figura 2.16, es muy sospechosa de estar mal generada. Por ejemplo, al realizar las pruebas iniciales de los clasificadores generados, las curvas ROC siempre se llegaban a un 100 % de detección, de ahí surgieron las dudas sobre la generación de curvas ROC planteadas en esta sección.

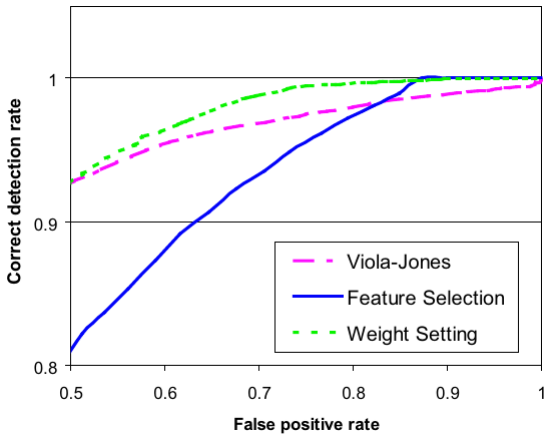


Figura 2.16: Curva ROC del artículo [31] con 100 % de detecciones.

2.3.2. Múltiples positivos

Otro punto importante es la forma en la que se decide si una ventana de detección contiene una cara o no. La base de datos CMU+MIT proporciona, por cada cara, la posición de los ojos, la nariz y las comisuras de la boca. A partir de estas posiciones, se puede considerar que una cara es toda aquella ventana de detección que contenga todos los puntos de una cara. El otro punto sería el tamaño máximo aceptable para considerar un positivo. En nuestro caso se ha decidido que el área máxima sea igual al doble del área marcada por dichos puntos.

Por otro lado, y tal como explican en el artículo original, el clasificador no es sensible a pequeños cambios de tamaño y posición. Por ello, alrededor de una cara aparecerán múltiples detecciones, tal como se puede ver en la figura 2.17. Al aparecer una sola cara en varias ventanas de detección, hay diferentes posibilidades a la hora de definir la curva ROC. La primera posibilidad es tomar cada una de las ventanas posibles y considerar que cada una es una detección objetivo. Esto provocaría que clasificadores sensibles a la posición y al tamaño, mostrasen un mayor número de fallos que los clasificadores que

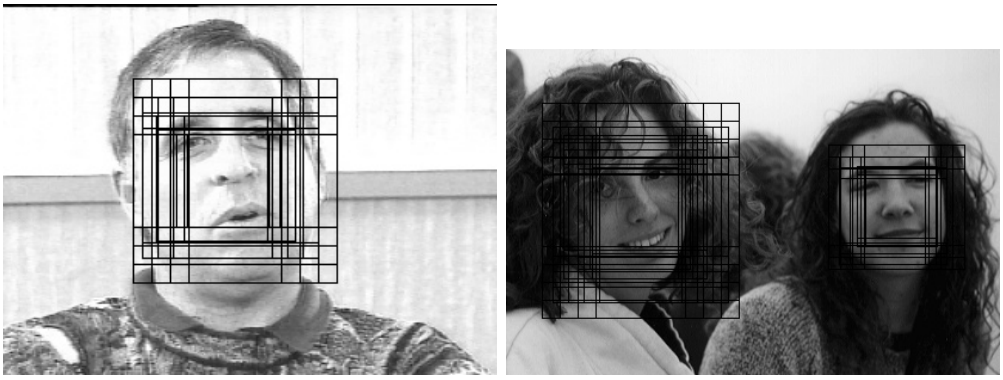


Figura 2.17: Ventanas de detección alrededor de las caras.

no sean sensibles. La principal ventaja de los clasificadores sensibles es que no haría falta otro algoritmo de agrupamiento de detecciones. La segunda posibilidad es tomar el conjunto de ventanas alrededor de una cara y considerar que se ha detectado la cara si al menos una de las ventanas ha sido detectada. El problema de esta opción es que clasificadores con una mala capacidad de detección podría mostrar la misma eficacia que un clasificador mejor.

Para solucionar el problema anterior, en el artículo original proponen agrupar las detecciones que tengan parte de su área en común y devolver la media de las detecciones agrupadas. Sin embargo, este agrupamiento implica que la curva ROC no sería una línea continua ascendente, sino que tendría forma de sierra como la que se puede ver en la figura 2.18. Para explicarlo mejor vamos a plantear tres situaciones de sensibilidad:

- **Sensibilidad baja**: Pocas detecciones que en la mayoría de los casos corresponden a caras reales. Apenas hay casos de agrupamiento y no afectan a las tasas de detección.

- **Sensibilidad media**: Bastantes detecciones tanto de caras reales como falsos positivos. Todavía no hay suficientes casos de agrupamiento como para afectar a las tasas de detección y de falsos positivos.

- **Sensibilidad alta**: Muchas detecciones. Si se producen alrededor de una cara y la media resultante sigue detectando la cara no hay problema, sin embargo la media puede provocar que deje de detectarse la cara, bajando la tasa de detecciones. Si se producen agrupamientos de falsos positivos el número total de los mismos se podría reducir al aumentar la sensibilidad.

Es la última situación la que provoca que la curva ROC tome forma de sierra al llegar a la zona de alta sensibilidad, al variar de forma descontrolada tanto la tasa de detección como la de falsos positivos

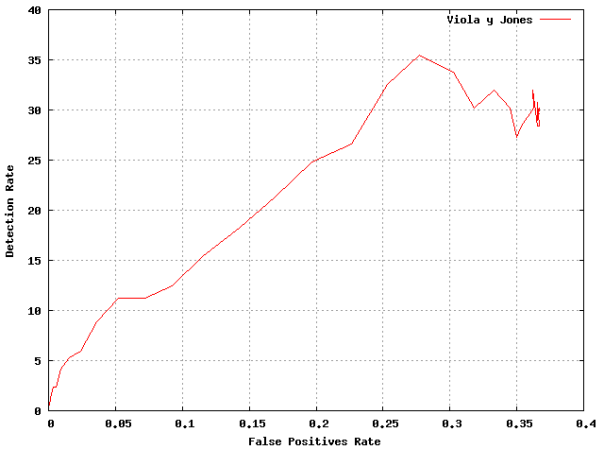


Figura 2.18: Ejemplo de curva ROC con forma de sierra.

Sin embargo, aunque en el propio artículo original indican que utilizan agrupamiento, la curva ROC que muestran es suave y continua, figura 2.19. Esto complica la comparación entre nuestros resultados y los planteados en los diferentes artículos.

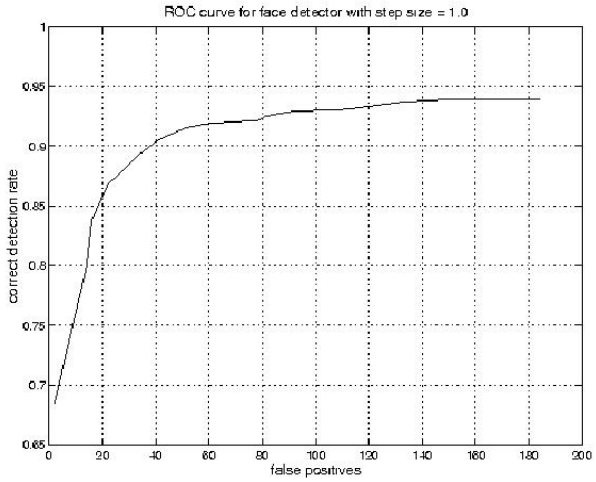


Figura 2.19: Curva ROC extraída del artículo [27].

2.3.3. Configuración de las curvas ROC

Para evitar los problemas planteados anteriormente, hemos decidido generar las curvas ROC de una forma poco ortodoxa, pero que, en nuestra opinión, facilita las comparaciones entre técnicas.

Figura 2.20: Muestra positiva utilizada para la generación de curvas ROC.

Primero se han separado las caras de las imágenes, tal como se puede ver en la figura 2.20. Dichas caras se mantienen en su tamaño original para no favorecer ninguno de los clasificadores. Esto provoca que la comparación de capacidad de detección entre diferentes clasificadores sea muy sencilla, puesto que las muestras positivas están perfectamente acotadas.

Figura 2.21: Muestra negativa utilizada para la generación de curvas ROC.

A continuación se han eliminado dichas caras de las imágenes de pruebas, como se puede ver en la figura 2.21. Estas nuevas imágemes no tienen caras detectables, por lo que todas las detecciones obtenidas al analizarlas serán falsos positivos. Esta forma de analizar la imagen también garantiza un número acotado de muestras negativas, mientras se mantegan los mismos cambios de desplazamiento y tamaño en las pruebas de todos los clasificadores.

Por terminar, indicar que en nuestro caso hemos comenzado con una ventana de detección de 24 por 24 píxeles, aumentándola en pasos de 12 por 12, y el desplazamiento en pasos de una octava parte del tamaño.

2.4. Conclusiones

Después del estudio bibliográfico se toma la decisión de implementar la técnica desarrollada por Viola y Jones dado sus resultados así como por ser la fuente de inspiración de multitud de trabajos posteriores.

La segunda técnica elegida, presentada por Lienhart y Maydt en [14], se ha seleccionado por ser utilizada por muchas librerías dedicadas al análisis de imágenes, como la librería de visión artificial OpenCV [2].

La tercera técnia seleccionada fue presentada por Levi y Weiss en [13]. Por un lado, se han seleccionado por utilizar un tipo de información totalmente diferente comparado con el artículo original. Por otro lado, el otr motivo de selección son los prometedores resultados que indicaban en el artículo.

Finalmente, se va a implementar una técnica totalmente diferente que, aunque no haya sido aplicada directamente al problema de la detección de caras, ha sido usada en un problema semejante como es la detección de humanos dando muy buenos resultados. De esta forma se dispone de una técnica diferente con la que comparar el resto de técnicas basadas en AdaBoost.

Capítulo 3

Estrategias analizadas para la detección de caras

3.1. Primera técnica: Algoritmo de Viola y Jones

El detector propuesto por Viola y Jones en [27] se ha planteado no sólo para detectar caras, sino que permite detectar prácticamente cualquier tipo de objeto mientras se disponga de una base de datos de imágenes para entrenar al clasificador. Dicho detector, tal como se describe en el artículo original, se basa en una serie de técnicas que influyeron notablemente en algoritmos posteriores: la imagen integral, el clasificador en cascada, el algoritmo AdaBoost y las características Haar.

Este algoritmo se puede extender en cualquiera de los puntos mencionados, aunque los dos que más se han trabajado han sido el algoritmo de aprendizaje AdaBoost y las características. Nuestro interés se centra en analizar el funcionamiento de los clasificadores finales para intentar mejorar su velocidad de clasificación. Por ello, el resto de técnicas que hemos decidido analizar se centran en extender las características, punto que más afecta al rendimiento final del clasificador. Modificar el algoritmo de aprendizaje no se ha planteado ya que dicha modificación no suele afectar notablemente al resultado final del clasificador, sino al tiempo que se tarda en generar dicho clasificador.

3.1.1. La imagen integral

La técnica de la "imagen integral" aparece por primera vez en el artículo [4] bajo el nombre de "tabla de área acumulada" y es utilizada en algoritmos de generación de

gráficos 3D para facilitar el cálculo de texturas a distintas resoluciones. Veinte años después Viola y Jones presentaron una estructura de datos basada en los mismos principios para utilizarla en su clasificador de caras. La imagen integral consiste en una matriz con las mismas dimensiones que la imagen original, pero en el que cada punto de la matriz contiene la intensidad acumulada de todos los puntos del cuadrado que va desde el origen de la imagen hasta el propio punto:

$$II(a,b) = \sum_{x=0}^{x=a} \sum_{y=0}^{y=b} I(x,y) \tag{3.1}$$

siendo $II(a,b)$ el valor en un punto de la imagen integral e $I(x,y)$ la intensidad en un punto de la imagen original. Un ejemplo de dicho cálculo se puede ver en la figura 3.1. La gran ventaja de esta imagen es que se puede calcular de forma muy rápida y, además, permite calcular características similares a las Haar de forma muy eficiente. Dichas características, explicadas en un apartado posterior, constituyen otro de los pilares del clasificador. Otra de las ventajas de esta estructura, tal como se explica en la sección 2.2, es la facilidad con la que se pueden normalizar las imágenes.

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

(a) Valores de intensidad de la imagen original.

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

(b) Valores de la imagen integral.

Figura 3.1: Ejemplo de cálculo de una imagen integral.

Además, esta estructura ha sido utilizada en otros artículos para contener otros tipos de información. Por ejemplo, en [13] se almacenan gradientes de intensidad para poder realizar comparaciones entre diferentes zonas.

A continuación se muestra un ejemplo de cómo se utiliza dicha estructura de datos y de la ventaja que supone. Por ejemplo, partiendo de la figura 3.2, calcular la intensidad acumulada del rectángulo formado por los puntos A, B, C y D se puede realizar con

cuatro accesos a memoria:

$$\sum_{x=x_A}^{x=x_D} \sum_{y=y_A}^{y=y_D} I(x,y) = II(x_A, y_A) + II(x_D, y_D) - II(x_B, y_B) - II(x_C, y_C) \tag{3.2}$$

mientras que sin dicha estructura se necesitarían tantos accesos como puntos en la zona calculada. Una vez obtenida la intensidad acumulada, ésta se puede utilizar para realizar comparaciones entre distintas zonas y, tal como se hace en el artículo, buscar patrones de intensidad en la imagen a clasificar.

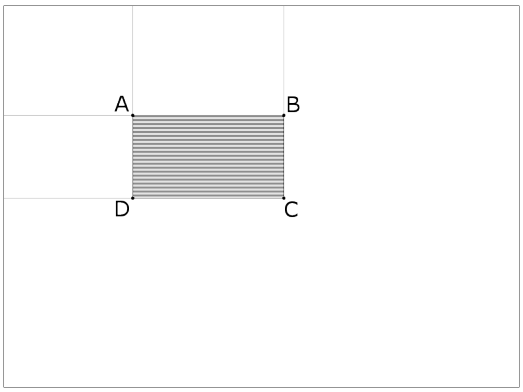


Figura 3.2: Cálculo de la intensidad acumulada de un rectángulo utilizando la imagen integral.

3.1.2. Clasificador en cascada

El clasificador en cascada surge por el cambio de concepto del problema de la detección de caras. Dicho problema dejó de considerarse como un un problema de clasificación para convertirse en un problema de detección de eventos poco comunes.

El concepto de "cascada" sustituye al de clasificador monolítico, en el que un sólo clasificador tiene que decidir si la imagen es una muetra positiva o no. El nuevo clasificador, por otro lado, se compone de una serie de clasificadores más pequeños que van aumentando progresivamente en complejidad. Tal como se puede ver en la figura 3.3, si una muestra es rechazada por alguna de las capas se considera negativa. Por el contrario, si atraviesa todas las capas se considera positiva. Esto permite que los primeros clasificadores sean muy rápidos y eliminen la mayor parte de muestras negativas mientras que los últimos, más lentos, realicen un procesado más fino para poder filtrar

las muestras más complejas.

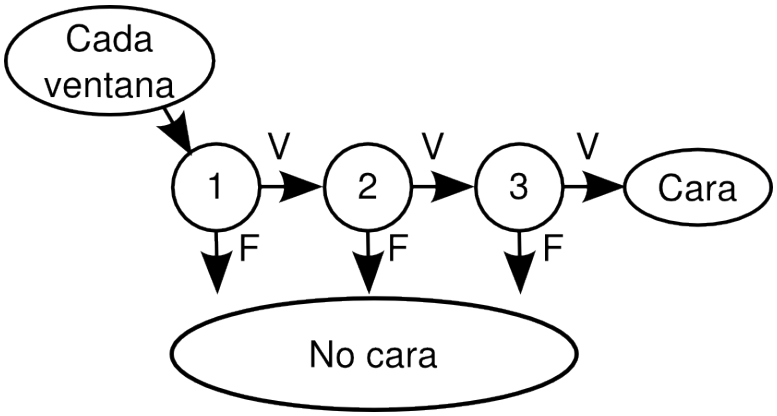


Figura 3.3: Estructura del clasificador en cascada.

Cada nivel consiste en un clasificador entrenado utilizando AdaBoost, técnica que también se explica en esta sección. AdaBoost permite generar un clasificador potente a partir de clasificadores sencillos. Si tenemos en cuenta el clasificador en cascada y el entrenamiento AdaBoost, tenemos un clasificador dividido en varios niveles en el que cada nivel consta de un número cada vez mayor de clasificadores débiles.

Uno de los puntos fuertes de este sistema, es que aunque todos los niveles utilizan el mismo conjunto de muestras positivas, las muestras negativas son más dinámicas. Dichas muestras se obtienen de forma aleatoria bajo la condición de que la capa anterior las clasifique como "positivas". El entrenamiento en cascada está planteado para que detecte, como mínimo, un porcentaje de los positivos verdaderos del nivel anterior y filtre, como máximo, otro porcentaje de los falsos positivos.

Para entrenar la cascada se tiene que decidir, aproximadamente, el número de capas y las tasas de detección y de falsos positivos finales. Por ejemplo, tal como se puede ver en la figura 3.4 para un clasificador de 20 capas, con una tasa de detección de 90 % y 0,0001 % de falsos positivos, cada capa tendría que ser capaz de detectar el 99,5 % de aciertos de la capa anterior y eliminar el 50 % de falsos positivos.

Un detalle importante a la hora de entrenar es la tasa relativa máxima. Aunque se desee una tasa de detección entre capas de $0,999$, si sólo disponemos de 100 muestras positivas, la tasa máxima sería de $0,99$, puesto que un valor más alto sería equivalente a utilizar un valor de 1. Aparte, la tasa relativa máxima aumenta con cada capa. Siguiendo con el ejemplo, la primera capa necesitaría una tasa relativa de $99/100 = 0,99$. La

- $R_T = R_L^N$
- $R_L = R_T^{1/N}$
- $DR_L = 0,995 = 0,9^{1/20}$
- $FP_L = 0,5 = (10^{-6})^{1/20}$

Siendo R_T la tasa final objetivo, N el número de capas, R_L la tasa relativa de cada capa, DR la tasa de detección y FP la tasa de falsos positivos.

Figura 3.4: Cálculo de las tasas de detección y de falsos positivos por cada capa.

segunda capa, por otro lado, necesitaría $98/100 = 0,98$. Sin embargo, según las fórmulas de la figura 3.4, la segunda capa tendría una tasa objetivo de $0,9801$, lo que equivale a no poder disminuir la tasa de detección en cada capa.

Por último, las tasas de detección y de falsos positivos no son fijas, por ejemplo, el primer nivel podría presentar una detección del 100 % y una tasa de falsos positivos del 30 %. Esto supone que para el siguiente nivel, y siguiendo los ejemplos anteriores, se tendría que conseguir una tasa de detección del 99,5 % y una tasa de falsos negativos del 15 %. Al acumularse estas pequeñas diferencias, al final se podría llegar a la tasa objetivo de falsos positivos con menos capas de las planteadas inicialmente.

3.1.3. Aprendizaje por AdaBoost

Adaboost es un algoritmo presentado en [6] que permite generar un clasificador fuerte a partir de una combinación de clasificadores débiles. Dicho algoritmo ser puede ver en la 3.5.

En el caso planteado por Viola y Jones, los clasificadores débiles que se utilizan para componer el clasificador final utilizan una única característica de la imagen. Dada su importancia, dichas características se explican en profundidad más adelante. Sin embargo, lo importante es que devuelven un valor numérico al ser aplicadas a una ventana de detección. Al final, el clasificador débil decide entre muestras positivas y negativas según la siguiente fórmula:

$$H(x) = p * t(x) >= p * T \tag{3.3}$$

- Dadas las muestras $(x_1, y_1), ..., (x_n, y_n)$ en las que $y_1 = -1, 1$ para los ejemplos negativos y positivos.
- Establecer el peso inicial de cada muestra a $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_1 = 0, 1$, donde m es el número de muestras positivas y l el número de muestras negativas.
- Para $t = 1, ..., T$, siendo T el número de clasificadores
  1. Normalizar los pesos: $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$.
  2. Por cada clasificador débil, calcular el error en base a los pesos normalizados: $\epsilon_{t,j} = \sum_i w_{t,i} \left| h_j(x_i) - y_i \right|$
  3. Elegir el clasificador, h_t con el menor error ϵ_t.
  4. Actualizar el peso de de cada muestra: $w_{t+1,i} = wt, i\beta_t^{1-\epsilon_i}$ siendo $e_i = 0$ si x_i se ha clasificado correctamente, $e_i = 1$ en caso contrario y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
- El clasificador final se calcula:

$$h(x) = \begin{cases} 1 & \text{si } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{en caso contrario} \end{cases}$$

Figura 3.5: Algoritmo AdaBoost.

donde p es la polaridad de la inecuación, T es el valor umbral por el que dividir valores positivos de negativos y $t(x)$ es el valor que devuelve la característica asociada al clasificador para la imagen dada. Las características planteadas en el artículo están explicadas en esta misma sección. Al asociar una única característica con cada clasificador débil, se puede ver que se generan tantos clasificadores como características.

El algoritmo AdaBoost original realiza tantos ciclos como clasificadores débiles existen, siendo el clasificador final una combinación lineal de todos los clasificadores posibles. Sin embargo, en el artículo de Viola y Jones se plantea realizar tantos ciclos como sean necesarios hasta que la capa cumpla con los requisitos de detección y de falsos positivos. Gracias a que los niveles anteriores eliminan gran parte de las muestras negativas, cada nivel se puede generar con una complejidad menor de la que tendrían si fuesen monolíticos.

El mayor cuello de botella del algoritmo es que, en cada ciclo del algoritmo AdaBoost, se tienen que entrenar de forma individual cada uno de los clasificadores débiles

propuestos para ser añadidos al clasificador fuerte. Esto supone que el tiempo total de entrenamiento es directamente proporcional al número de características, $O(n)$, y al número de muestras de entrenamiento, en el peor de los casos $O(n^2)$. Una forma de realizar dicho entrenamiento lo proponen en [28] donde plantean el siguiente algoritmo:

1. Se calcula el peso total de los elementos positivos de entrenamiento T_p y de los elementos negativos T_n.

2. Se ordenan todos los elementos de entrenamiento en base al valor devuelto por la característica asociada al clasificador.

3. Se recorre la colección de elementos de entrenamiento, en orden ascendente, mientras se calculan S_p, el peso acumulado de las muestras positivas aparecidas al recorrer la colección, y S_n, el peso acumulado de las muestras negativas aparecidas al recorrer la colección.

4. Además, por cada elemento de la colección, se calculan dos posibles errores según las siguientes fórmulas:
$$\begin{cases} e_p = S_p - (T_m + S_m) \\ e_m = S_m - (T_p + S_p) \end{cases}$$

5. Al final se obtiene el error más pequeño posible para el clasificador. Siendo el umbral el valor devuelto por la característica asociada en el elemento de entrenamiento donde se calculó el error más pequeño. La polaridad depende del error utilizado:
$$p = \begin{cases} 1 \text{ si } e_p < e_m \\ -1 \text{ si } e_m < e_p \end{cases}$$

3.1.4. Características propuestas por Viola y Jones

Las características que se utilizan en cada clasificador débil de los propuestos anteriormente recuerdan a "Wavelets" Haar y por ello se las suele denominar, simplemente "características tipo Haar". Estas características tienen la capacidad de codificar dentro de si mismas conocimiento del dominio, en este caso particular, diferencias entre las

intensidades acumuladas de regiones diferentes de la imagen. En concreto, en el artículo original proponen los siguientes tipos de características:

- Características dobles: dividen una zona rectangular en dos partes iguales y devuelven la diferencia entre las intensidades de las dos zonas, ver figura 3.6. Se pueden plantear tanto verticales como horizontales. Gracias al cálculo de la imagen integral esta operación se puede resolver con seis accesos a memoria.

$$t(x) = I(\text{derecha}) - I(\text{izquierda}) \tag{3.4}$$

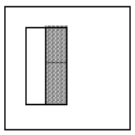


Figura 3.6: Ejemplo de característica doble horizontal.

- Características triples: dividen una zona rectangular en 3 partes, en las que la las partes exteriores ocupan lo mismo y la central ocupa lo mismo que las dos exteriores juntas, ver figura 3.7. Al igual que las características dobles, se pueden plantear tanto verticales como horizontales. Estas características devuelven la diferencia entre la zona central y la suma de las zonas exteriores.

$$t(x) = I(\text{central}) - (I(\text{derecha}) + I(\text{izquierda})) \tag{3.5}$$

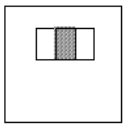


Figura 3.7: Ejemplo de característica triple horizontal.

- Características cruzadas: dividen una zona rectangular en 4 partes iguales, se suma la intensidad de las partes opuestas y se restan las sumas, ver figura 3.8.

$$t(x) = (I(\text{NE}) + I(\text{SO})) - (I(\text{NO}) + I(\text{SE})) \tag{3.6}$$

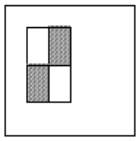


Figura 3.8: Ejemplo de característica cruzada.

3.1.5. Análisis del clasificador generado

Para poder comparar correctamente las diferentes técnicas, se ha generado un clasificador utilizando el algoritmo planteado en esta sección. La configuración utilizada es muy similar a la planteada por los autores. Se han utilizado muestras de 24 por 24 píxeles y un total de 162.336 posibles características. El número de características del artículo original es de algo más de 180.000, sin embargo las 20.000 características de diferencia no se pueden localizar debido a que el artículo no explica claramente como obtiene sus 180.000 características.

A continuación se presenta un estudio del clasificador generado para analizar en profundidad como funciona y poder generar un clasificador mejor. El estudio se centra en analizar tres posibles mejoras:

- Centros de atención: Buscando elementos más pequeños se podrían generar más fácilmente clasificadores mucho más complejos pero manteniendo la velocidad.
- Especialización de características: Eliminar aquellas características que no aportan información para aumentar el número de aquellas que sí lo hacen.
- Búsqueda de nuevas características: En base al clasificador actual buscar características que aumenten la eficacia.

Los centros de atención del clasificador se pueden obtener buscando aquellas partes de la ventana de detección más utilizadas por el clasificador. En la figura 3.9 se ven las zonas utilizadas por el total del clasificador. A mayor utilización mayor claridad. Lo importante de esta imagen es que se puede ver que las características de la cara que más información contienen son la nariz, los ojos y una de las mejillas. Sin embargo, la boca apenas es utilizada. En vez de generar un clasificador que detecte la cara completa, podría ser más eficaz centrarse en los ojos y la nariz y generar clasificadores parciales con una eficacia mucho mayor.

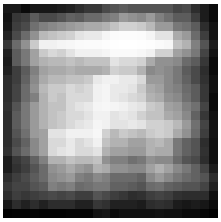


Figura 3.9: Partes de la ventana de detección analizadas por el clasificador.

Por otro lado, para eliminar aquellas características que apenas aportan información, lo más conveniente es ver el índice de utilización de cada característica en el clasificador final. Tal como se puede ver en la tabla 3.1, la mayoría de las características utilizadas son las dobles y las cruzadas. Estos datos chocan con un problema señalado en el artículo [14]. En dicho artículo, los autores decidieron excluir del clasificador las características cruzadas, al poderse representar dichas características utilizando las demás. Sin embargo, con los datos que hemos obtenido, habría sido más útil eliminar las características triples ya que aportan menos información.

Horizontal doble	45.771 %
Horizontal triple	7.068 %
Vertical doble	4.751 %
Vertical triple	7.764 %
Cruzada	34.647 %

Tabla 3.1: Uso por cada tipo de característica del clasificador de Viola y Jones.

Por otro lado, también se pueden utilizar las propiedades geométricas de las características Haar. Las propiedades que se han analizado son las siguientes: posición, área y relación entre anchura y altura. En la tabla 3.2 se pueden ver los valores máximos de dichas propiedades. A partir de esa información se pueden extrapolar los siguientes datos:

- *Posición*: estos datos indican que hay que aprovechar el total de la imagen, en el total del clasificador, todos los tipos de característica aprovechan cada punto de la imagen.
- *Área*: el área mínima de los clasificadores es de 1 píxel por cada zona de la característica. Por otro lado, el área máxima de las características es, como mucho, el 70 % de la ventana de detección.

Propiedad	*Horizontal doble*	*Horizontal triple*	*Vertical doble*	*Vertical triple*	*Cruzada*	*Todas*
Pos. máx. izquierda	0	0	0	0	0	0
Pos. máx. derecha	24	24	24	24	24	24
Pos. máx. arriba	0	0	0	0	0	0
Pos. máx. abajo	24	24	24	24	24	24
Área mín.	2	3	2	3	4	2
Área máx.	380	342	192	360	324	384
Máx. rel. altura/anchura	10,5	7	8	24	10	24
Máx. rel. anchura/altura	16	24	5	7	10	24

Tabla 3.2: Propiedades máximas de las características Haar del clasificador Viola.

- *Relación entre anchura y altura*: este dato resulta bastante curioso. Las características horizontales tienden a tener mayor anchura que altura y las verticales al revés, mientras que las características cruzadas no presentan ninguna tendencia.

Este análisis sirve para intentar atacar el proceso de clasificación reduciendo el número de características. Partiendo del supuesto de que estamos utilizando una ventana de 24 por 24 píxeles, si eliminamos las características que superan las propiedades máximas indicadas, ya que no se van a utilizar en el entrenamiento, reduciríamos en un 4,14 % el número de características sin afectar a la calidad del clasificador. Este recorte apenas afecta al tiempo de entrenamiento, con lo que se podría descartar intentar simplicar el clasificador con esta técnica.

La última opción es buscar características nuevas que puedan ser utilizadas para entrenar al clasificador. Lo interesante de utilizar característiticas más amplias es que para hacer los cálculos se ahorran accesos a memoria. Por ejemplo, las características cruzadas analizan 4 zonas utilizando 9 puntos en vez de 16. En este estudio hemos buscado características del clasificador final, dentro de cada nivel, que compartan puntos entre sí. Esa búsqueda se ha realizado con la esperanza de encontrar sugerencias

que indiquen nuevos tipos de características. A continuación se presenta una lista con aquellos grupos que se han considerado más interesantes.

- Grupos de dos características dobles en L.

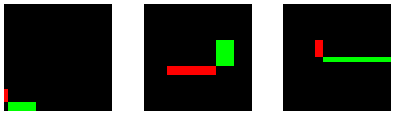
  Este tipo de conjunto aparece en muchas ocasiones dentro del clasificador original. La forma conjunta recuerda a las caracterísitticas triples pero, en vez de estar las tres zonas alineadas, forman una estructura en L en la que se comparan la zona central con las dos exteriores que son las puntas de la L.

- Subzonas de mayor peso.

  Aunque estos conjuntos son escasos en el clasificador final, parecen una posibilidad simple de desarrollar. Consisten en características centradas encima de otras del mismo tipo, pero ligeramente más pequeñas. Esto permite que partes de las zonas adquieran mayor peso que el resto.

- Características en diagonal.

  Son los conjuntos más comunes pero también los que menos información aportan. Simplemente son dos características con una de las esquinas en común. La única ventaja es que se reduce 1 el número de accesos a memoria, pero no parece haber ningún tipo de patrón en su combinación, ya que se juntan características de cualquier tipo y tamaño.



El resto de combinaciones son un conjunto variopinto de características sin ningún tipo de sentido, con la única propiedad en común que comparten algún punto de la imagen.

3.1.6. Resultados

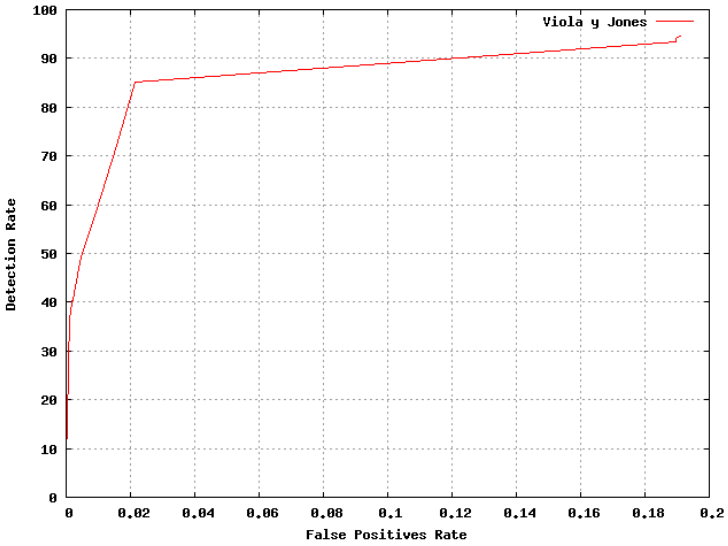


Figura 3.10: Curva ROC del clasificador entrenado con la técnica de Viola y Jones probado con la base de datos de imágenes CMU+MIT.

En la figura 3.10 se puede observar la curva ROC de una prueba del clasificador entrenado utilizando la técnica del artículo original. Para generar la curva se parte del hecho de que cada capa es un clasificador AdaBoost. Dichos clasificadores tienen un valor umbral que permite ajustar la sensibilidad del mismo que se ajusta a su valor óptimo, durante el entrenamiento. Sin embargo, para generar la curva ROC se puede modificar el umbral de sensibilidad de la última capa para obtener diferentes relaciones entre la tasa de detección y la tasa de falsos positivos.

Por otro lado, el otro punto a valorar es la velocidad de clasificación. El tiempo que se ha tardado en analizar el conjunto completo de imágenes de la base de datos CMU+MIT, formado por 40 imágenes en blanco y negro, se puede ver en la tabla 3.3. Este tiempo será muy útil para terminar de comparar los clasificadores. El problema del sistema es que a cada capa se le exige una capacidad de detección, con lo que los diferentes clasificadores, al ser entrenados de forma similar, acabarán con una capaci-

dad de detección similar. Sin embargo, si un clasificador realmente mejora su eficacia, necesitará menos tiempo para analizar las imágenes.

Creación de las imágenes integrales	0,89”
Análisis de las imágenes	14,07”
Total	14,96”

Tabla 3.3: Tiempo de análisis de la base de datos de imágenes CMU+MIT utilizando el clasificador entrenado con la técnica de Viola y Jones.

Estos tiempos han sido obtenidos al probar el clasificador generado con la base de datos CMU+MIT completa, indicando por un lado el tiempo que se han necesitado para generar las imágenes integrales y por otro lado el tiempo que se ha tartado en analizar todas las ventanas posibles. Por último, y a modo informativo, indicar que estas pruebas se han realizado utilizando un computador con un procesador a 2.500MHz.

3.2. Segunda técnica: Giro de 45º en las características de Viola y Jones

La siguiente técnica a analizar es una evolución directa de la planteada por Viola y Jones. Lo interesante de esta técnica es que ha pasado a convertirse en una de las bases de los detectores de caras modernos. Por ejemplo, la librería OpenCV [2], dedicada a proporcionar funciones relacionadas con la visión artificial, presenta para la detección de caras una serie de funciones basadas en esta técnica.

La modificación propuesta en el artículo [14] por Rainer Lienhart y Jochen Maydt consiste en añadir un nuevo tipo de característica central y en utilizar una versión de la imagen integral rotada 45º con un conjunto nuevo de características. Además, descartan utilizar las características “cruzadas” del artículo original.

La principal ventaja de la nueva imagen integral rotada 45º es que proporciona información en diagonal, lo que se añade a la información vertical y horizontal proporcionada por las características originales.

Otra diferencia con el artículo original consiste en asignar pesos a cada una de las zonas con las que trabaja cada característica. Así tenemos, por ejemplo, que para las características triples se plantean dos opciones:

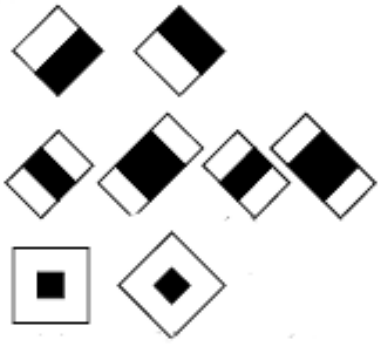


Figura 3.11: Nuevas características propuestas por Rainer Lienhart y Jochen Maydt.

- Tres zonas iguales, en las que la zona central tiene el doble de peso que las zonas exteriores.
- La zona central tiene el doble de tamaño que los bordes, en las que todas las zonas tienen el mismo peso.

Las características originales, por el contrario, asignaban el mismo peso a todas las zonas. El hecho de asignar pesos diferentes podría plantear nuevas características con diferentes configuraciones de peso. Sin embargo, las características planteadas por Viola y Jones estan diseñadas de forma que, durante el entrenamiento, se obtiene un valor que representa la diferencia numérica necesaria entre cada zona para detectar una cara. Esta diferencia realiza las mismas funciones que la relación de pesos entre zonas. La excepción podrían ser las características con más de dos zonas, en las que las zonas agrupadas podrían sumarse con pesos diferentes. Por ejemplo, en la característica triple horizontal, el borde derecho podría tener un peso diferente al borde izquierdo.

3.2.1. Características centrales

Estas nuevas características comparan el valor central de una zona contra el resto del área, tal como se puede ver en la figura 3.11. Un particularidad de esta característica, tal como la plantean los autores, es que divide el área en 9 zonas del mismo tamaño y se compara la zona central contra el área total, incluyendo otra vez la propia zona central.

Respecto a los pesos que se mencionan al principio de la sección, el área total tiene peso 1 y la zona central 9. Esto se hace para facilitar el cálculo, podemos obtener el

valor de la característica simplemente calculando la intensidad acumulada del centro multiplicada por 9 y restándole la intensidad acumulada del área total.

3.2.2. Imagen integral con giro de 45º

El concepto de imagen integral girada se puede ver en la imagen 3.12. En la integral original, tal como se puede ver en la sección 3.1, un punto de la imagen integral representa la intensidad acumulada de la imagen desde el origen de coordenadas hasta dicho punto. En la integral girada, por otro lado, un punto representa la intensidad acumulada en el área izquierda que resulta de cortar la imagen con dos rectas inclinadas 45º y 135º respecto al eje horizontal.

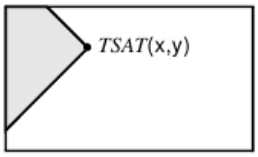


Figura 3.12: Imagen integral con rotación de 45º.

En el articulo se plantea que generar dicha integral en dos pasadas, tal como se puede ver en la figura 3.13.

- Primero de izquierda a derecha y de arriba a abajo:
  $RI'(x,y) = I(x,y) + RI'(x-1,y-1) + RI'(x-1,y) - RI'(x-2,y-1)$
  siendo $RI'(-1,y) = RI'(-2,y) = RI'(x,-1) = 0$.
- Segundo de derecha a izquierda y abajo a arriba:
  $RI(x,y) = RI'(x,y) + RI'(x-1,y+1) - RI(x-2,y)$

Figura 3.13: Algoritmo original para generar la imagen integral con un giro de 45º.

Sin embargo, en las pruebas realizadas se comprobó que dicho algoritmo no genera valores correctos. Por ejemplo, para una imagen de 5 por 7 puntos con todas las intensidades a uno, la integral con rotación, teórica tal como se plantea en el artículo, debería ser tal como se puede ver en la tabla 3.4. Sin embargo, con el algoritmo que proponen los autores se genera una imagen integral con valores como los que se ven en la tabla 3.5.

1	3	6	10	15
1	4	8	13	19
1	4	9	15	22
1	4	9	16	23
1	4	9	15	22
1	4	8	13	19
1	3	6	10	15

Tabla 3.4: Imagen integral con rotación teórica.

1	3	5	7	9
1	4	7	10	13
1	4	8	12	16
1	4	8	13	18
1	4	8	13	19
1	4	8	13	19
1	3	5	7	9

Tabla 3.5: Imagen integral con rotación generada con la fórmula de Lienhart y Maydt.

El resultado de dicho descenso gradual respecto al valor real implica que imágenes grandes con caras colocadas en la zona sureste serán más complicadas de detectar, ya que el clasificador se entrena con caras situadas en el extremo noroeste de una imagen pequeña.

Para poder probar las características de este artículo, se desarrolló un algoritmo diferente, visible en la figura 3.14, que permite generar la integral de 45º en una sola pasada.

$$RI_1(x,y) = I(x,y) + RI(x-1,y+1) + I(x-1,y)$$

$$RI_2(x,y) = I(x,y) + RI(x-1,y-1) + I(x-1,y)$$

$$RI_3(x,y) = I(x,y) + RI(x-1,y-1) + RI(x-1,y+1) + I(x-1,y) - RI(x-2,y)$$

$$RI(x,y) = \begin{cases} RI_1(x,y) & \text{si } RI(x-1,y-1) = 0 \\ RI_2(x,y) & \text{si } RI(x-1,y+1) = 0 \\ RI_3(x,y) & \text{en caso contrario} \end{cases}$$

Figura 3.14: Algoritmo propuesto para generar correctamente la imagen integral con un giro de 45º.

Por último, para utilizar la información que proporciona la integral girada se utiliza

una nueva serie de características equivalentes a las originales pero con un giro de 45º. Sin embargo, la utilización de estas características es menos intuitiva visualmente que las originales. Por ejemplo, una característica triple original se podía asociar fácilmente con los dos ojos: dos zonas oscuras con una zona más clara en el centro que es la nariz. Sin embargo, ciertas líneas como los labios o los ojos, no son totalmente horizontales o verticales sino que tienen una ligera inclinación que se puede codificar con las nuevas características propuestas.

3.2.3. Análisis del clasificador generado

Con esta técnica también se ha desarrollado un clasificador de caras al igual que se hizo en la sección 3.1. De igual modo, se va a intentar realizar un estudio para buscar los puntos fuertes y puntos débiles del clasificador para poder mejorar su eficacia.

Empezando con los centros de atención, el uso de ventana por parte del clasificador se puede ver en la figura 3.15. Al igual que con el clasificador de Viola y Jones, los centros de atención se sitúan principalmente en los ojos, la nariz y en una mejilla.

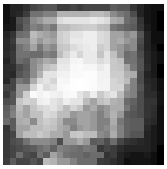


Figura 3.15: Partes de la ventana de detección analizadas por el clasificador generado con la técnica de Lienhart y Maydt.

Continuando con las características del clasificador, si se analiza el índice de utilización de las diferentes características, visibles en la tabla 3.6, las conclusiones son diferentes que para el clasificador de Viola y Jones. Por un lado, las características centrales apenas se utilizan, menos del 5 % si se juntan las normales y las rotadas. Este dato contradice claramente la decisión de los autores de sustituir las características cruzadas por las centrales. Por otro lado, las características rotadas sí se utilizan considerablemente, siendo estas casi un 40 % del total de características. También se puede ver que las verticales dobles prácticamente han desaparecido del clasificador final. Por último, aunque se mantiene la superioridad de uso de las características horizontales dobles respecto a las demás, la proporción es más suave.

Horizontal doble	38.231 %
Horizontal triple	8.980 %
Vertical doble	0.544 %
Vertical triple	10.068 %
Central	3.810 %
Horizontal doble 45º	8.707 %
Horizontal triple 45º	8.436 %
Vertical doble 45º	9.660 %
Vertical triple 45º	10.885 %
Central 45º	0.680 %

Tabla 3.6: Uso de cada tipo de características en el clasificador de Lienhart y Maydt.

Por último, al buscar correlaciones entre las diferentes características utilizadas en el clasificador final se obtienen los mismos resultados que con los clasificadores normales: formas en L y diferentes pesos para diferentes zonas. Algunos ejemplos de dichas correlaciones se pueden ver en la figura 3.16.

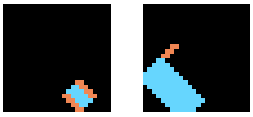


Figura 3.16: Correlaciones encontradas en el clasificador generado con la técnica de Lienhart y Maydt.

3.2.4. Resultados

El clasificador generado también se ha probado utilizando el set de imágenes CMU+MIT. Para comparar con el generado en la sección 3.1 se han reproducido las curvas ROC de ambos en la figura 3.17. En dicha figura se puede ver como la curva ROC no tiene porque siempre tener la misma silueta. La forma de escalera que se puede observar probablemente sea debida a un grupo de clasificadores que, al aumentar la sensibilidad, han detectado una serie de falsos positivos simultáneamente. Particularmente se pueden ver tres zonas: en la zona de sensibilidad baja ambos clasificadores se comportan igual, en la zona de sensibilidad media la técnica de Viola y Jones es bastante mejor que la de Lienhart y Maydt y en la zona de sensibilidad alta es al revés, el clasificador de Lienhart y Maydt es bastante mejor que el de Viola y Jones.

Respecto a la velocidad del clasificador, ésta se puede ver en la tabla 3.7. Algo de esperar es la duplicación del tiempo de generación de las imágenes integrales, al tener

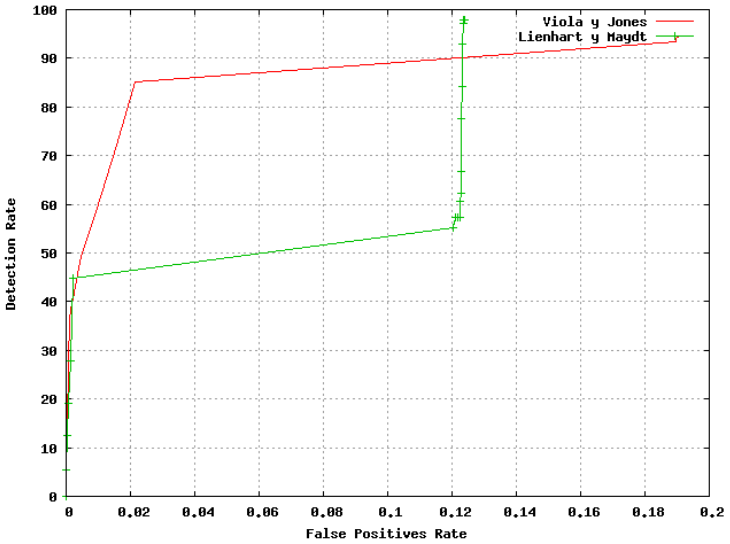


Figura 3.17: Curva ROC del clasificador entrenado con la técnica de Lienhart y Maydt probado con la base de datos de imágenes CMU+MIT.

que generar una para las características originales y otra para las giradas. Por otro lado, se puede ver que, además de conseguir mejores resultados, también se obtiene un algoritmo ligeramente más rápido, probablemente debido a que necesite un menor número de características para codificar la información de las caras.

Creación de las imágenes integrales	1,74”
Análisis de las imágenes	9,31”
Total	11,05”

Tabla 3.7: Tiempo de análisis de la bae de datos de imágenes CMU+MIT utilizando el clasificador entrenado con la técnica de Lienhart y Maydt.

3.3. Tercera técnica: Histograma de orientaciones de bordes locales o EOH

Otra modificación del algoritmo de Viola y Jones la presentan Kobi Levi y Yair Weiss en [13] donde propusieron añadir nuevas características, totalmente diferentes a las del artículo original. En esta nueva propuesta utilizan como fuente de información de las características el gradiente de intensidad, en vez de la propia intensidad de la imagen.

3.3.1. Gradiente de intensidad

Tal como la intensidad es la base de información para las características originales, la base de todas las nuevas características es el gradiente de intensidad. El grandiente de intensidad de una imagen es, en cada punto, el vector de crecimiento de la intensidad respecto a los puntos de alrededor. Si consideramos a la imagen como una función sobre dos dimensiones, el gradiente sería la derivada de dicha función.

La gran diferencia de esta información respecto a la utilizada por las características originales, es que se tienen dos valores: la magnitud del vector gradiente y su ángulo.

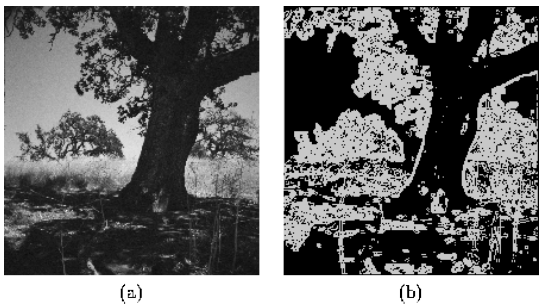


Figura 3.18: Efecto del operador Sobel sobre una imagen.

La generación del vector gradiente se realiza utilizando diferentes tipos de técnicas, siendo la más común la máscara sobel. El efecto de aplicar dicha máscara sobre una imagen se puede ver en la figura 3.18.

Dicha máscara permite calcular dos valores G_x y G_y correspondientes al cambio de la intensidad en la dirección de la coordenada X y de la coordenada Y, respectivamente. Seguidamente, mediante una combinación de éstos, se puede determinar la magnitud del gradiente y su dirección. En la ecuaciones (3.7), (3.8),(3.9) y (3.10) se explica el cálculo matemático del operador, siendo $*$ la operación de convolución de dos dimensiones y A la imagen original representada como una matriz de intensidades. Un ejemplo de aplicación de estas fórmulas se puede ver en las tablas 3.8, 3.9 y 3.10. En las que la primera tabla representa los valores de intensidad de una imagen, la segunda la magnitud del vector gradiente y la tercera el ángulo de dicho vector.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \tag{3.7}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \tag{3.8}$$

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{3.9}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.10}$$

154	152	157	173	185
105	77	76	86	102
108	134	110	91	72
131	110	79	69	61
126	69	98	107	29

Tabla 3.8: Valores de intensidad de la imagen original.

477,00	335,78	319,10	359,47	416,18
453,54	139,46	194,04	326,71	441,51
462,87	122,28	120,03	99,03	155,56
423,99	179,65	113,56	144,83	121,69
394,92	209,48	158,61	263,07	211,29

Tabla 3.9: Magnitudes del vector gradiente.

Para poder utilizar la información proporcionada por el vector gradiente, en el artículo se propone agrupar los vectores en diferentes rangos según el ángulo, creando un histograma de magnitudes del vector gradiente. Por ejemplo, dividiendo en 4 grupos tal

36,99º ↗	93,92º ↑	80,80º ↑	76,81º ↑	79,33º ↑
165,96º →	67,66º ↑	88,81º ↑	97,38º ↑	98,20º ↑
10,58º →	129,02º ↖	169,43º →	46,63º ↗	81,86º ↑
176,07º →	43,64º ↗	39,99º ↗	9,13º →	43,00º ↗
16,32º →	144,71º ↖	76,13º ↑	134,07 ↖	130,77º ↖

Tabla 3.10: Ángulo del vector gradiente en el rango de 0º a 180º.

como recomiendan los autores, agrupamos los vectores según su orientación: horizontal, vertical, diagonal derecha y diagonal izquierda.

Por cada grupo, se genera una matriz con las magnitudes del gradiente en cada punto, para aquellos vectores que pertenecen al grupo. Siguiendo el ejemplo dado y tal como se ve en las tablas 3.11, solo una de las cuatro matrices tendría valor para cada punto de la imagen original.

0	0	0	0	0
453,54	0	0	0	0
462,87	0	120,03	0	0
423,99	0	0	144,83	0
394,92	0	0	0	0

(a) Matriz de magnitudes de la dirección →

477,00	0	0	0	0
0	0	0	0	0
0	0	0	99,03	0
0	179,65	113,56	0	121,69
0	0	0	0	0

(b) Matriz de magnitudes de la dirección ↗

0	335,78	319,10	359,47	416,18
0	139,46	194,04	326,71	441,51
0	0	0	0	155,56
0	0	0	0	0
0	0	158,61	0	0

(c) Matriz de magnitudes de la dirección ↑

0	0	0	0	0
0	0	0	0	0
0	122,28	0	0	0
0	0	0	0	0
0	209,48	0	263,07	211,29

(d) Matriz de magnitudes de la dirección ↖

Tabla 3.11: Matrices de las magnitudes por cada grupo de ángulos.

Además, los autores proponen establecer un umbral de valor para tener en cuenta dicha magnitud. En particular, los autores recomiendan 80 sobre un máximo de 255, por debajo del cual no se tiene en cuenta la magnitud.

Finalmente, por cada matriz se genera una imagen integral utilizando el mismo algoritmo planteado para generar la imagen integral de las intensidades. Dichas imagenes integrales serán la fuente de información de las nuevas características.

3.3.2. Nuevas características

A continuación hay un listado de las nuevas características que proponen los autores y que se alimentan de la información del gradiente de intensidad.

- Relación entre gradientes

  Esta característica devuelve la relación entre la intensidad acumulada de los gradientes de un rango y los gradientes de otro rango de una zona dada. A diferencia de las características originales que dividían una zona en partes iguales, estas características utilizan la información de toda la zona, pero de rangos de ángulos diferentes.

  Un ejemplo gráfico de la utilidad de esta característica es, por ejemplo, en la zona de los ojos, donde la intensidad acumulada de gradientes horizontales o diagonales tiene que ser notablemente superior a la de los verticales.

- Gradiente dominante

  Aquí se analiza la relación entre un rango particular respecto al total de todos los rangos. Por ejemplo, en la zona de la boca el gradiente horizontal debería ser mucho más intenso que todos los demás.

- Simetría.

  Una información extra que se puede obtener gracias al cálculo de gradientes es la simetría de una imagen. Dadas dos zonas situadas simétricamente respecto al centro de la imagen, resulta sencillo calcular sus gradientes y comprobar si son simétricos, o no. Aplicado a la detección de caras, esto permite detectar la simetría en los ojos o los carrillos y detectar la falta de simetría entre la frente y la barbilla.

Un punto a tener en cuenta a la hora de generar las características que utilizan gradientes, es que no se puede utilizar el borde de la imagen ya que el operador Sobel, tal como se ve en la tabla 3.10, provoca que los bordes de una imagen siempre tengan un gradiente perpendicular al mismo. Dado que las muestras positivas de entrenamiento ocupan la imagen completa, los bordes provocarán que el clasificador se especialice en detectar dichos bordes, ya que las muestras negativas no presentan dicho borde al estar tomadas de forma aleatoria del interior de la imagen.

3.3.3. Análisis del clasificador generado

Con esta técnica también se ha desarrollado un clasificador de caras al igual que se hizo en las secciones 3.1 y 3.2. De la misma forma, también se ha realizado un análisis del clasificador generado. Como novedad, en este clasificador también hay que analizar de que forma se utilizan los grupos de ángulos en los que se dividen los gradientes.

La primera diferencia de este clasificador son los centros de atención. En la figura 3.19 se puede ver que, de forma general, los centros se sitúan en la boca, los ojos y la nariz. Este cambio puede deberse a que las nuevas características, al utilizar información de gradiente, se centran en cualquier parte de la cara que presente bordes más definidos.

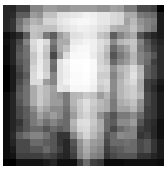


Figura 3.19: Partes de la ventana de detección analizadas por el clasificador generado con la técnica de Lienhart y Maydt.

A modo de resumen de los tres clasificadores, se puede ver claramente que los clasificadores tienden a concetrar los esfuerzos en ciertas zonas que son las que proporcionan más información. En particular, parece que para un rostro la búsqueda se centra en los ojos, nariz, boca y mejillas. Si en vez de crear un clasificador general se creasen pequeños clasificadores especializados en estas partes, probablemente se conseguiría un clasificador mucho más óptimo y con mejores capacidades.

Continuando con el análisis, la utilización de las nuevas características, visibles en la tabla 3.12, aporta también datos interesantes. Primero, el porcentaje de uso de las características originales es completamente diferente al de los demás artículos, rompiendo la tendencia. Por otro lado, las características que utilizan información de gradiente son las más utilizadas, llegando a un 58 % del total. Este dato sugiere que sería interesante probar a realizar un clasificador únicamente con características de este tipo, para comprobar si la reducción en tiempo de generar la imagen integral de las características normales compensa la pérdida de información por parte de las características originales. Para continuar, las características que analizan la simetría son mínimas, apenas un 3 % del total. Aunque el planteamiento inicial de estas características tenía sentido,

si se quiere generar un clasificador más complejo, sería recomendable eliminarlas para acelerar el proceso de entrenamiento.

Horizontal doble	10.049
Horizontal triple	7.084 %
Vertical doble	1.483 %
Vertical triple	11.532 %
Cruzada	12.191 %
Simetría Horizontal	1.483 %
Simetría Vertical	1.318 %
Orientación Dominante	24.382 %
Relación entre horientaciones	30.478 %

Tabla 3.12: Uso de cada de tipo de característicaen en el clasificador de Levi y Weiss.

La tabla 3.13 muestra el uso de los diferentes grupos de ángulos. Tal como recomendaban los autores, para este clasificador se utilizaron 4 grupos de ángulos, a los que previamente se les había normalizado entre 0º y 180º. Cada grupo se podría asociar con las siguientes orientaciones: horizontal, vertical, diagonal derecha y diagonal izquierda. Un detalle curioso es que el uso de los diferentes grupos está más o menos igualado. En un futuro sería interesante ver como afectaría un número diferente de grupos, ya que podría ser interesante hacer discriminaciones más específicas. Por otro lado, también sería interesante normalizar respecto a 360º, en vez de los 180º que proponen los autores.

0º, 180º	26.641 %
45º, 225º	24.517 %
90º, 270º	25.290 %
135º, 305º	23.552 %

Tabla 3.13: Porcentaje de uso de cada grupo de ángulos en el clasificador de Levi y Weiss.

Por último, al buscar correlaciones de características han aparecido los grupos ya encontrados en los demás clasificadores, formas en L y subzonas de mayor peso como las que se pueden ver en la imagen 3.20. Algo a tener en cuenta es que las nuevas características no dividen una región de la ventana en zonas, sino que comparan dicha ventana en diferentes planos, cada grupo de ángulos en los que se han dividido los gradientes.

Figura 3.20: Correlaciones encontradas en el clasificador generado con la técnica de Levi y Weiss.

Aparte, la aparición de las características simétricas ha hecho aparecer algunas correlaciones curiosas, como la que se puede ver en la figura 3.21. Dicha correlación sugiere que para las características que utilizan el vector gradiente, secciones con formas diferentes a la cuadrada pueden llegar a aportar mucha información.

Figura 3.21: Correlaciones simétricas en el clasificador generado con la técnica de Levi y Weiss.

3.3.4. Resultados

El resultado de utilizar este clasificador con la base de datos de prueba, muestran que tiene una capacidad de detección muy superior al resto de algoritmos analizados, tal como se puede ver en la 3.22. Dicha figura ha sido recortada para poder apreciar el aumento de falsos positivos que sufre la técnica de Levi y Weiss con sensibilidad alta, algo que no se podría ver si la curva ROC de la técnica de Viola y Jones apareciese completa.

Sin embargo, el problema de este clasificador reside en el tiempo de generación de las imagenes integrales. Tal como se puede ver en la tabla 3.14 dicho tiempo supone más del 50 % del tiempo. Si lo comparamos con la técnica de Viola y Jones, ésta solo utiliza un 6 % del tiempo en la generación. Por otro lado el tiempo de análisis es sensiblemente inferior al resto de técnicas. Con un algoritmo mejor de generación de la imagen integral se podría llegar a conseguir un clasificador mucho más eficaz y mucho más rápido.

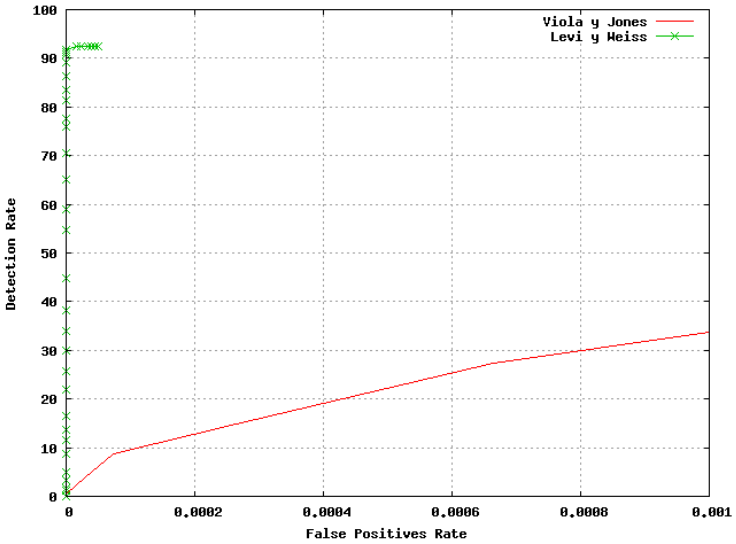


Figura 3.22: Curva ROC del clasificador entrenado con la técnica de Levi y Weiss probado con la base de datos de imágenes CMU+MIT.

Creación de las imágenes integrales	7,02”
Análisis de las imágenes	6,97”
Total	13,99”

Tabla 3.14: Tiempo de análisis de la base de datos de imágenes CMU+MIT utilizando el clasificador entrenado con la técnica de Levi y Weiss.

3.4. Cuarta técnica: Histograma de gradientes orientados con SVM

En esta sección se ha desarrollado una cuarta técnica totalmente diferente a la utilizada hasta ahora, esto es, una técnica que no esté inspirada en el método desarrollado por Viola and Jones. En concreto, se eligió la técnica presentada en [5] por Dalal y Triggs en donde se presentan resultados satisfactorios en la detección de humanos en una imagen. Nosotros vamos a aplicarla a la detección de caras teniendo en cuenta que el problema es semejante. La novedad de este trabajo es que utiliza como característica para entrenar a la máquina SVM los histogramas de gradientes orientados. Se basa en el hecho de que los contornos de un objeto pueden representarse por una distribución local de gradientes de intensidades y dirección de sus contornos. Esto se logra dividiendo la ventana en células o celdas sobre las que se calcula un histograma local de gradientes de los píxeles contenidos en dicha celda

3.4.1. Histograma de gradientes orientados

En este artículo, las características que se utilizan para detectar personas están basadas en el concepto de gradiente de intensidad. A continuación vamos a explicar con más detalle todo el proceso a seguir para generar las características. El proceso completo se puede ver en la figura 3.23. Algo a tener en cuenta es que los autores, por cada paso, provaron diferentes posibilidades antes de decantarse por una en particular. Por ejemplo, para una distribución gaussiana provaron diferentes valores de σ.

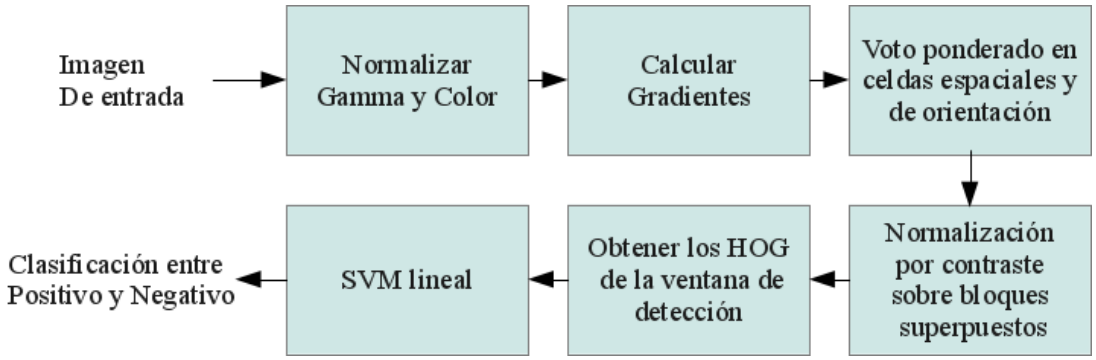


Figura 3.23: Secuencia de procesado de la técnica presentada en [5].

- **Normalización**: partiendo de imágenes tanto en blanco y negro como en color, utilizan equalización gamma para normalizar las imágenes, $V_{out} = \alpha V_{in}^{\gamma}$, aunque no indican los parámetros de dicha función. El objetivo de la normalización es evitar posibles desajustes debido a las diferentes condiciones de luminosidad en las que puede operar el clasificador. De todas formas, según los autores, la normalización apenas parece influir en la efectividad del clasificador. Por otro lado, utilizar imágenes en blanco y negro reduce la efectividad del clasificador.

- **Cálculo de los vectores gradientes**: los autores partieron de un listado bastante extenso de técnicas para calcular el vector gradiente y los compararon para averiguar cual es la que mejor ayudaba a clasificar personas. En particular, el gradiente 1-D $[-1, 0, 1]$ con $\sigma = 0$ parece funcionar mejor que el resto de propuestas de los autores, superando también al operador sobel mencionado en 3.3. Un detalle importante es que para las imágenes en color, para cada punto calculan el gradiente de cada canal de color, y utilizan aquel que tenga mayor magnitud.

- **Celdas o células espaciales y de orientación**: en esta fase se divide la imagen en celdas, para cada una de las cuales se calcula el histograma de gradientes. Las distintas magnitudes y orientaciones del gradiente de los píxeles de cada celda se usan para determinar las orientaciones de su histograma. En el artículo comentan que con 9 orientaciones es suficiente. Estas nueve orientaciones resultan de agrupar los grados de 20 en 20, es decir, 0, 20, 40, 60, 80, 100, 120, 140, 160. Nótese que el resto de los ángulos presentan una dirección igual a uno de los anteriores, esto es, 180 grados es equivalente a 0 grados o 20 grados a 200. La figura 3.24 muestra estas orientaciones. Seguidamente, se asocia la orientación de cada píxel de la celda a una de estas nueve de manera que un píxel ponderará a la orientación que coincide con su dirección de gradiente. Sin embargo, en la mayoría de los casos, la dirección del gradiente no coincide con ninguna de estas orientaciones. Para resolver este problema se realiza una interpolación entre las dos orientaciones vecinas. De esta forma se genera un HOG para cada celda de la imagen y cada celda contendrá 9 orientaciones de gradiente.

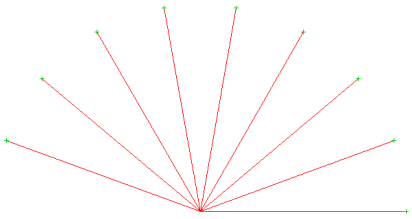


Figura 3.24: Agrupaciones de ángulos para las distintas celdas.

- **Normalización de bloques**: En el artículo demuestran que se consiguen mejores resultados cuando agrupamos las celdas en bloques. Los bloques son conjuntos solapados de células. Los autores consideraron bloques rectangulares, cuadrados y circulares, aunque al final parece que los bloques cuadrados ofrecen la mejor eficacia en la detección. Aparte de la forma de los bloques, también provaron diferentes formas de normalización, como por ejemplo: $v' = \frac{v}{\sqrt{||v||_2^2+\epsilon^2}}$.
- **Ventana de detección**: el paso final para generar una muestra es agrupar los bloques en ventanas de 64 por 128 puntos. Dicha ventana incluye un margen de 16

píxeles alrededor de la persona a detectar, dejando un espacio de 32 por 96 puntos para representar la persona. Este margen es algo que los autores remarcan como importante, por lo que sería interesante analizar la eficacia de este clasificador ante figuras solapadas en las que no hay opción de margen en la imagen.

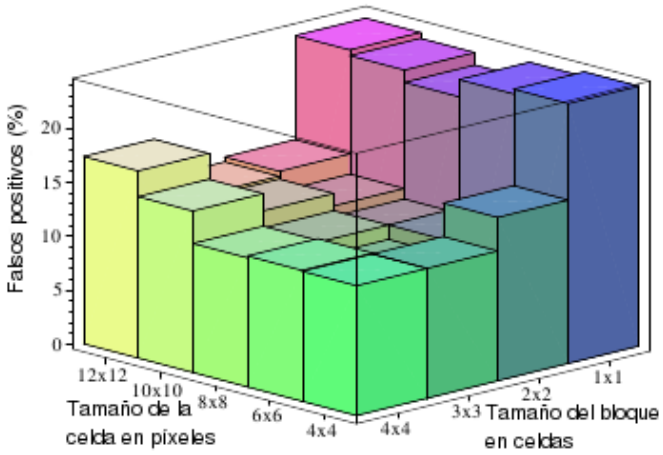


Figura 3.25: Falsos negativos según el tamaño de la célula y del bloque en las características HOG.

Los autores analizaron distintas combinaciones de tamaños tanto para las celdas como para los bloques con el fin de seleccionar la mejor combinación. En la figura 3.25 se muestran los resultados obtenidos en dichas pruebas. Se aprecia como con tamaños de celda pequeños y tamaños de bloque grandes se consiguen menores porcentajes de falsos positivos.

El último paso del proceso, el clasificador SVM, está explicado en más detalle en el siguiente punto de la sección.

3.4.2. Aprendizaje por SVM

SVM, del inglés Máquina de Vectores de Soporte, es un clasificador englobado dentro de los llamados "métodos de aprendizaje supervisado" y fue presentado por Vladimir Vapnik y Corinna Cortes en [3]. Este método de aprendizaje permite clasificar los datos entre dos clases. La idea del método es determinar el hiperplano que separa las dos clases de manera óptima.

El funcionamiento de un clasificador SVM depende, en mayor parte, del kernel que se utilice para evaluar las muestras y, dentro de los posibles kernels que se pueden utilizar destacan principalmente dos:

- SVM lineal: utiliza un kernel simple en el que cada característica de una muestra se corresponde con un parámetro del clasificador. El entrenamiento busca ajustar el peso de cada parámetro de las muestras de forma que se minimice el error cometido al clasificar las muestras de entrenamiento. El clasificador generado con este kernel divide el plano de las muestras según una línea recta, con lo que no tiene mucha utilidad para casos complejos.

- SVM con distancia Gaussiana: utiliza un kernel basado en distancias que utiliza la fórmula (3.11). Las distancias se calculan respecto a las muestras de entrenamiento, que de esa forma se convierten en los parámetros del clasificador. Este kernel permite dividir el plano de las muestas según patrones complejos a costa de un mayor tiempo de cómputo.

$$k(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2}) \tag{3.11}$$

La gran ventaja de los clasificadores SVM es que se disponen de librerías muy optimizadas para su cálculo. Por lo que para utilizarlas simplemente hay que escoger el kernel a utilizar y los parámetros de normalización dados.

En el caso particular del artículo analizado, plantean utilizar un SVM lineal con un parámetro de normalización $C = 0{,}01$. Según los autores, utilizar un kernel Gaussiano aumenta ligeramente la eficacia del clasificador pero aumenta el tiempo de clasificación de forma prohibitiva. El valor de normalización también ha sido seleccionado entre diversas opciones según la eficacia del mismo.

3.4.3. Resultados

Este clasificador no dispone de un parámetro con el que ajustar la sensibilidad. Por lo tanto, solo se ha podido obtener un valor de detección y un valor de la tasa de falsos positivos. Aparte, el tiempo de procesado de las imágenes no es relevante en este caso, dado que los anteriores clasificadores fueron creados pensando en la velocidad como

objetivo, mientras que el actual está planteado como un prototipo para comparar los resultados con las demás técnicas.

Clasificador	Detección	Falsos positivos	tiempo de procesado
Viola y Jones	55,62 %	0,005 %	14,96”
Dalal y Triggs (SVM)	53,14 %	0,15 %	-

Tabla 3.15: Comparación de capacidades de los diferentes clasificadores.

La tabla 3.15 se ha generado ajustando la sensibilidad del clasificador de Viola y Jones de forma que los valores de detección quedasen a la par del clasificador generado con la técnica de Dalal y Triggs. En dicha tabla se puede apreciar que la tasa de falsos positivos es 30 veces mayor que con el resto de clasificadores, lo que indica que este clasificador puede no ser el más adecuado para la tarea.

3.5. Conclusiones

Una vez generados los clasificadores para cada una de las técnicas y gracias al análisis en profundidad realizado por cada uno de ellos hemos llegado a una serie de conclusiones que nos ayudarán a realizar propuestas de mejora.

La técnica de Viola y Jones, analizada en la sección 3.1, genera clasificadores rápidos y eficientes. Al analizar el mismo se ha visto que las características triples apenas son utilizadas y que las características cruzadas, eliminadas en otros artículos, son una parte importante del clasificador final. También se ha visto que analizando apariciones conjuntas de características se pueden generar nuevas formas, como características con forma de L o características con pesos variables.

La técnica de Lienhart y Maydt, analizada en la sección 3.2, aporta dos novedades que son las características centrales y la imagen integral con rotación. Consideramos bastante importante el hecho de que la fórmula que proponen para calcular dicha imagen integral es errónea, necesitando una versión alternativa para realizar las pruebas. Sin embargo, gracias al análisis se ha visto que, aunque generar dicha imagen integral supone un mayor tiempo de preparación, la información que aporta ayuda a la eficacia y velocidad del clasificador final. Respecto a las características centrales, se ha visto que en este caso en particular, la detección de caras, apenas aportan inforamción.

La tercera técnica analizada, propuesta por Levi y Weiss y visible en la sección 3.3, utiliza la información del gradiente de intensidad en vez de utilizar la intensidad directamente. Tal como se puede ver en el análisis, este cambio supone una mejora sustancial en la eficacia del clasificador respecto a la técnica de Viola y Jones. Sin embargo, tiene el problema de aumentar en casi 10 veces el tiempo necesario para generar las imágenes integrales.

Por último, la técnica analizada en la sección 3.4, basada en SVM y presentada por Dalal y Triggs, aunque presenta unos resultados muy prometedores en el ámbito de la detección de personas completas, no parece adaptarse bien al problema de la detección de caras.

Capítulo 4

Estrategia propuesta para mejorar la detección de caras

4.1. Introducción

Tras realizar el análisis de los clasificadores generados con las técnicas selecciondas, se han considerado las siguientes opciones como un medio para mejorar la eficacia de la detección de caras. Dicha mejora esta orienta a generar un clasificador con como mínimo la misma capacidad de detección que los planteados pero tardando menos tiempo en analizar una imagen.

- Características dobles: generar un nuevo clasificador utilizando únicamente las características dobles planteadas por Viola y Jones. Aunque las verticales dobles apenas se utilizan en el artículo original, se pueden mantener en esta opción ya que son igual de rápidas que las horizontales. Aparte, para aumentar la variedad de características se propone utilizar asimetría. Es decir, que cada característica divida la región analizada en zonas de diferente tamaño, a diferencia del planteamiento original.

- Características en L: generar un nuevo clasificador que añada un nuevo tipo de característica con forma de L, tal como aparece en algunas de las correlaciones encontradas.

- Gradiente en solitario: generar un nuevo clasificador que utilice únicamenete información del gradiente de intensidad. Aparte, utilizar aproximaciones de las opera-

ciones matemáticas más pesadas para intentar acelerar la generación de la imagen integral.

- Clasificadores especializados: está opción es la que parece va a dar los mejores resultados. Sin embargo, es un cambio de tal magnitud que necesitaría una investigación propia. Por ejemplo, solo crear la base de datos de muestras requeriría un mínimo de 4 veces el tiempo que se ha necesitado para crear las muestras de este estudio.

Cada una de estas modificaciones se va a probar en solitario para comprobar si realmente ayudan a mejorar la efictividad. Probar todas las modificaciones a la vez no permitiría distinguir cuales son válidas y cuales no.

4.2. Propuesta 1: utilizar únicamente características dobles

Este clasifidor está planteado para utilizar únicamente las características dobles del clasificador original, por ser las más rápidas y las más utilizadas. Sin embargo, un problema es la reducción en la diversidad que esto supondría, pasando de un total de 160.000 características a solo 86.400. Para evitar este problema se ha optado por dar más variedad a las características dobles sin modificar su velocidad.

Las características dobles originales dividían la sección analizada en dos partes iguales a comparar. La modificación propuesta permite que las secciones sean de diferentes tamaño, tal como se puede ver en la figura 4.1.

Figura 4.1: Características dobles asimétricas.

Con esta modificación se han conseguido un total de 1.380.000 características para utilizar en el entrenamiento, casi 10 veces el número original.

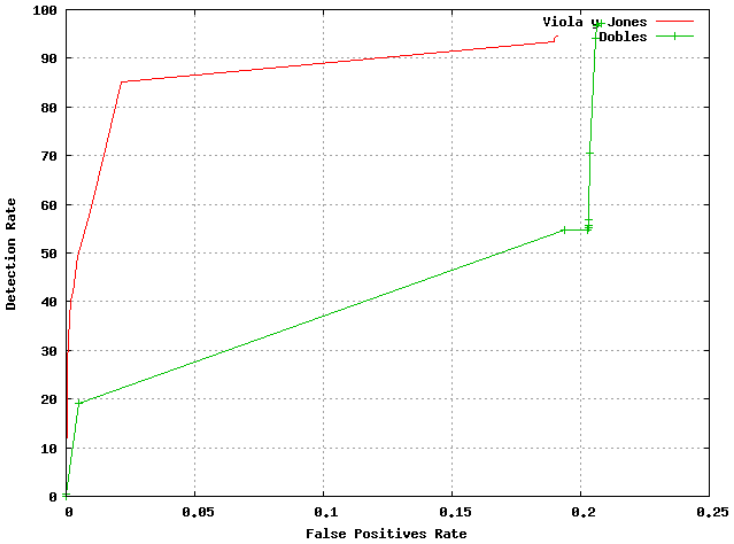


Figura 4.2: Curva ROC del clasificador con solo características dobles probado con la base de datos de imágenes CMU+MIT.

Tal como se puede ver en la figura 4.2, la curva ROC de este clasificador sigue aproximadamente la del clasificador realizado con la técnica de Lienhart y Maydt. Sin embargo, presenta una tasa de falsos negativos ligeramente superior, incluso en la zona de sensibilidad alta.

Creación de las imágenes integrales	0,89”
Análisis de las imágenes	12,78”
Total	13,57”

Tabla 4.1: Tiempo de análisis de la base de datos de imágenes CMU+MIT utilizando el clasificador entrenado con solo características dobles.

De todas formas, algo que también es interesante es el tiempo de procesado. Tal como se puede ver en la tabla 4.1, tarda 1,5 segundos menos que el clasificador entrenado con la técnica de Viola y Jones. Aunque no es una mejora muy sustancial, sí que indica una posible dirección para desarrollar clasificadores más rápidos.

4.3. Propuesta 2: añadir las nuevas características en L

Las nuevas características en L planteadas en esta sección son una mezcla de las características cruzadas y las triples. Las características triples parten una sección recta

en tres trozos y comprueban si el central es más o menos oscuro. Las características cruzadas dividen en cuatro partes y comparan las zonas situadas en una diagonal con la otra diagonal. Las nuevas características en L dividen una zona en 4 partes y desechan una de ellas, dejando una L. Las tres zonas que forman la L se comparan de la misma forma que las características triples: comparando si la zona central es más o menos oscura que las de los bordes. Ejemplos de estas características se pueden ver en la figura 4.3. El número de características añadidas con esta técnica es reducido comparado con las demás técnicas, siendo únicamente 82.944 nuevas opciones.

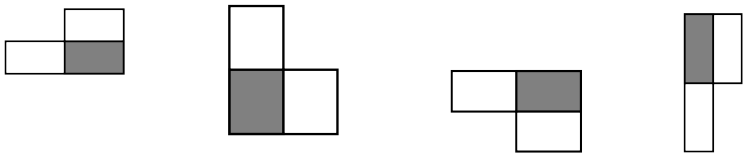


Figura 4.3: Características en L.

En la figura 4.4 se puede ver la curva ROC del clasificador generado. Dicha curva también sigue, de forma aproximada, la curva del clasificador realizado con la técnica de Lienhart y Maydt, en donde se utilizaron las características centrales y la imagen integral rotada. Al igual que en dicho caso, en la zona de sensibilidad alta consigue una menor tasa de falsos positivos.

Con relación al tiempo y tal como se puede ver en la tabla 4.2, el tiempo de generación de las imagenes integrales se mantiene al no haberse modificado nada. Por otro lado, el tiempo de procesado ha mejorado ligeramente, probablemente al necesitar un menor número de clasificadores débiles para codificar la cara.

Creación de las imágenes integrales	0,89”
Análisis de las imágenes	13,64”
Total	14,53”

Tabla 4.2: Tiempo de análisis de la base de datos de imágenes CMU+MIT utilizando el clasificador entrenado con las nuevas características en L.

Otro dato que nos interesa conocer para este clasificador es el índice de utilización de las nuevas características, visible en la tabla 4.3. Las nuevas características suponen un 10 % del total del clasificador. Un dato que confirma la utilidad de buscar estas nuevas características.

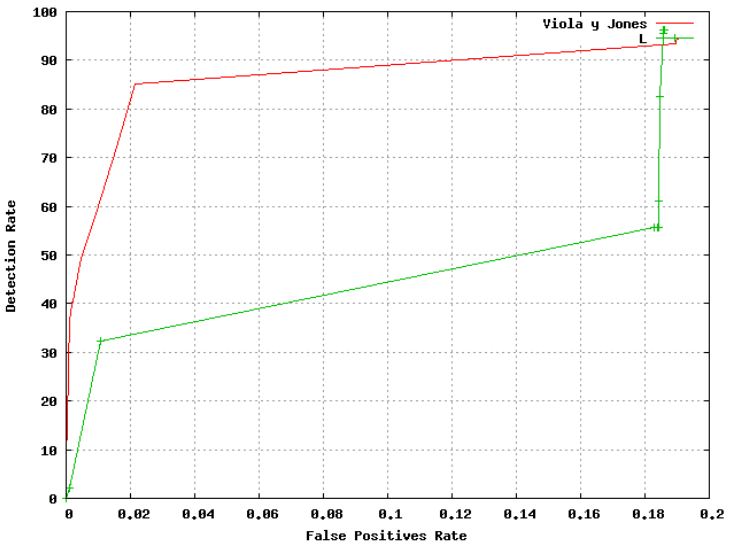


Figura 4.4: Curva ROC del clasificador con las características en L probado con la base de datos de imágenes CMU+MIT.

Horizontal doble	51.631 %
Horizontal triple	2.950 %
Vertical doble	3.101 %
Vertical triple	3.101 %
Cruzada	28.635 %
L	10.581 %

Tabla 4.3: Uso por cada tipo de características en el generado clasificador con las nuevas características en L.

4.4. Propuesta 3: optimización de EOH

Este clasificador parte del generado en la sección 3.3 e intenta optimizarlo para conseguir otro clasificador igual de eficaz pero mucho más rápido.

Partiendo del supuesto de que para realizar el clasificador no hacen falta valores concretos, sino aproximaciones, se ha modifcado el cálculo de la magnitud del vector gradiente, pasando de la ecuación (4.1) a la aproximación (4.2).

$$G = \sqrt{{G_x}^2 + {G_y}^2} \tag{4.1}$$

$$G = |G_x| + |G_y| \tag{4.2}$$

.

El otro gran cambio es el cálculo del ángulo de dicho vector. En el planteamiento original se calcula el ángulo del vector y luego se asigna a uno de los grupos planteados. Sin embargo, sabiendo de antemano que sólo hay 4 grupos se puede realizar el siguiente cálculo para evitar utilizar la función *arctn*:

$$k = \begin{cases} 0 \text{ si } |G_y| * 5 <= |G_x| * 2 \\ 2 \text{ si } |G_x| * 5 <= |G_y| * 2 \\ 1 \text{ si } G_x * G_y >= 0 \\ 3 \text{ en el resto de casos.} \end{cases}$$

El primer caso agrupa aquellos vectores con ángulos desde los 338º a los 22º y de los 158º a los 202º. Para ello, la relación máxima entre el componente vertical y el componente horizontal es la misma relación que existe entre el $sin(22)$ el $cos(22)$, aproximadamente $0,4$ o $2/5$. La segunda opción plantea la misma situación, pero con los componentes invertidos. Por último, para diferenciar entre ambas diagonales se comprueba si tienen mismo signo o no.

Aparte, también se ha eliminado el umbral establecido por los autores. Es decir, en este clasificador no hay un umbral por debajo del cual se ignoran las magnitudes del gradiente. Ese cambio se debe a que vemos motivos para ignorar información, si las magnitudes demasiado suaves no aportan información, el algoritmo de entrenamiento se encargaría de desecharlas.

Respecto a la eficacia del clasificador generado, en la figura 4.5 se puede ver como, al igual que el clasificador de la sección 3.3, mejora notablemente la eficacia del clasificador de Viola y Jones.

Por último, el tiempo de generación de integrales se ha reducido en un segundo, un 15 % menos del tiempo que se necesitaba con la técnica original. Además el tiempo de análisis se ha reducido notablemente, probablemente debido a que al eliminar el umbral mínimo, los clasificadores han podido gestionar mejor la información.

4.5. Comparación de resultados

En esta sección se va a comparar la eficacia de las tres técnicas originales planteadas con respecto a las mejoras propuestas en este capítulo.

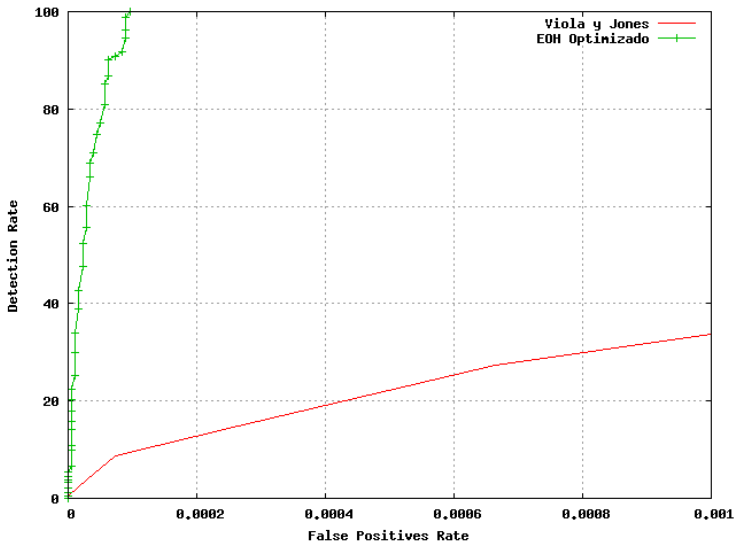


Figura 4.5: Curva ROC del clasificador con la técnica EOH optimizada probado con la base de datos de imágenes CMU+MIT.

Creación de las imágenes integrales	5,92”
Análisis de las imágenes	3,21”
Total	9,13”

Tabla 4.4: Tiempo de análisis de la base de datos de imágenes CMU+MIT utilizando el clasificador generado con la técnica EOH optimizada.

El primer punto a comparar es la eficacia de los clasificadores en cuanto a relación entre porcentaje de detección y tasa de falsos positivos. Para ello vamos a ver las curvas ROC de la figura 4.6. Algo que se puede apreciar es que los nuevos clasificadores de la propuesta 1, características dobles, y de la propuesta 2, características en L, no consiguen superar en eficacia al resto y solo mejoran al algoritmo de Viola y Jones con sensibilidad alta.

Por otro lado, la propuesta 3, basada en optimizar la técnica de EOH, al igual que la ténica original, supera a las demás con creces. Aunque, tal como se puede ver en la figura 4.7, la nueva técnica no consigue llegar al nivel de falsos positivos que la original.

Por último, el dato que más nos interesa es la velocidad de clasificación, algo que se puede consultar en la tabla 4.5. De las propuestas investigadas, destaca notablemente el tiempo conseguido por la técnica del EOH optimizado, siendo notablemente inferior a todas las demás técnicas.

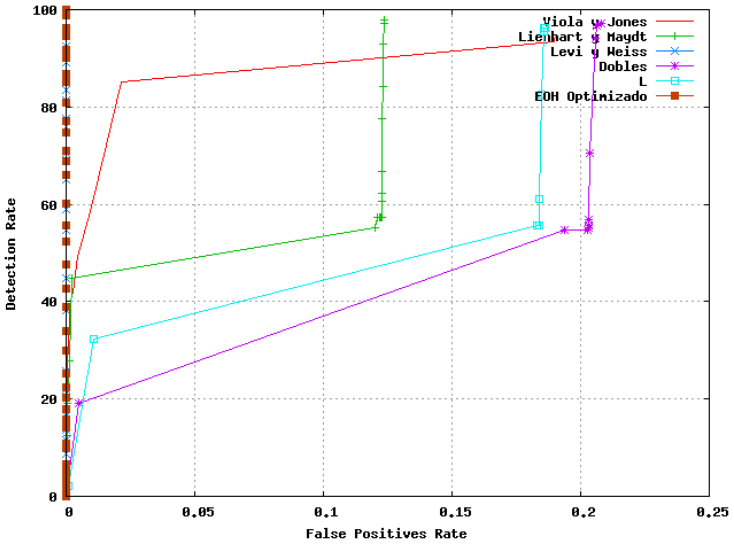


Figura 4.6: Curvas ROC de todos los clasificadores generados.

Clasificador	Tiempo total de análisis
Viola y Jones	14,96”
Lienhart y Maydt	11,05”
Levi y Weiss	13,99”
Características dobles	13,57”
Características en L	14,53”
EOH optimizado	9,13”

Tabla 4.5: Comparación de tiempos de todos los clasificadores generados.

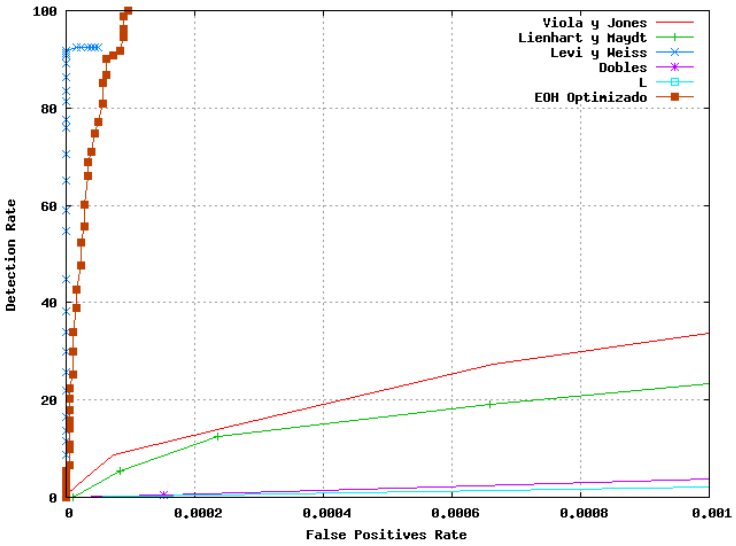


Figura 4.7: Curvas ROC de todos los clasificadores generados, centrada en las técnicas de grandiente de intensidad.

4. ESTRATEGIA PROPUESTA PARA MEJORAR LA DETECCIÓN DE CARAS

Capítulo 5

Conclusiones y futuros trabajos de investigación

5.1. Conclusiones

Con este trabajo se ha intentado generar un detector de caras que tenga la misma eficacia, en cuanto a la detección de caras, que otras técnicas actuales pero reduciendo el tiempo de análisis, algo fundamental en tareas en tiempo real.

En nuestra opinión, algo que ha influido notablemente en los clasificadores es la base de datos de entrenamiento. En nuestro caso, se han tardado varios días en conseguir 250 caras cortadas y normalizadas para el entrenamiento, mientras que Viola y Jones utilizaron cerca de 5000, algo que llevaría varias semanas. Este desfase puede indicar porque nuestros clasificadores no han alcanzado el mismo nivel de falsos positivos que se alcanzan en los artículos originales.

En concreto, se han cubierto los objetivos planteados. Se ha hecho un estudio bibliográfico profundo relacionado con el problema de la detección de caras en imágenes. Se han analizado los algoritmos más relevantes tomando el algoritmo de Viola y Jones [27] como punto de partida dada su influencia en las investigaciones posteriores. Dicho análisis se ha realizado en profundidad, implementando cada uno de los algoritmos, presentando resultados y analizando sus deficiencias o mejoras respecto a la propuesta original de Viola y Jones. Precisamente, otro de los puntos importantes de nuestra investigación ha sido analizar los clasificadores que se han generado con cada técnica. Este análisis provoca conocer mejor como funciona por dentro cada clasificador, así como

la información que utiliza para tomar decisiones. En otras investigaciones parece que las nuevas características surgen como conceptos novedosos, pero sin haber un estudio que indique que esa nueva característica es útil. Lo que es aún peor, no hay un estudio posterior que compruebe hasta que punto han sido realmente útiles al clasificador generado.

Como resultado de este estudio, en este trabajo se han definido tres propuestas de mejora. Todas ellas han sido valoradas obteniendo resultados muy importantes en cuanto al tiempo total. En particular, para el caso de la propuesta tercera, la cual optimiza el cálculo de vectores gradientes, en donde, con la misma eficiencia en la detección, el tiempo de análisis de la base de datos de imágenes de CMU+MIT es de 9,13" frente a los 14,96" del método presentado por Viola y Jones.

Otro punto importante de la investigación ha sido la obtención de la base de datos de muestras Aunque, tal y como se explica en la sección 2.2, existen bases de datos de caras disponibles para investigaciones, éstas no presentan el mismo formato del que se alimenten los algoritmos de entrenamiento. Para esta investigación se han obtenido 250 caras cortadas y normalizadas, lo cual ha llevado varios días de trabajo. Un mayor número de muestras habría supuesto un mayor tiempo de entrenamiento pero también que los clasificadores finales se conseguirían mejores resultados. Sin embargo, la obtención de más muestras implicaría semanas de trabajo dedicadas a esa tarea, reduciendo el tiempo disponible para la propia investigación.

Finalmente, en el anexo se proponen varias optimizaciones al algoritmo de AdaBoost que disminuirían el tiempo de cómputo durante el entrenamiento. Aunque dichas optimizaciones no resulten de vital importancia para un clasificador ya entrenado, si son importantes desde el punto de vista de la investigación. Hay que tener en cuenta que en esta investigación se han generado seis clasificadores AdaBoost. Dado que cada uno de ellos supone varios días de entrenamiento en un computador, todas las optimizaciones que se realicen permiten dedicar más tiempo a analizar el clasificador final generado.

Por todo ello, considero que mi propuesta cumple con el objetivo que se tenía al principio de la investigación: mejorar la velocidad de detección de los algoritmos de detección caras.

5.2. Futuros trabajos de investigación

Algo que se ha mencionado a lo largo de la investigación son los centros de atención de los diferentes clasificadores. Es decir, que partes de la ventana de detección son las que aportan más información. En particular, se ha visto que las zonas más analizadas son los ojos, la nariz y la mejilla. Desarrollar clasificadores independientes para cada una de estas zonas podría generar un nuevo clasificador mucho más rápido y eficaz que los actuales. Además, tendría la ventaja añadida de poder utilizar pequeños detectores para dichas zonas en vista de perfil, lo que permitiría discriminar la orientación actual de la cara, algo bastante complicado a día de hoy. Esta propuesta tendría dos problemas principalmente. Por un lado, habría que generar un algoritmo de agrupamiento para decidir entre varios elementos detectados cuales forman un cara y cuales otra. Por otro lado, habría que generar la base de datos de muestras de entrenamiento, algo que llevaría meses de trabajo.

Por último, y vista la eficacia de los clasificadores basados en gradientes de intensidad, sería recomendable realizar una investigación en profundidad de la técnica. Analizando que ángulos son los más utilizados en las caras y que divisiones serían las más adecuadas. Por otro lado, utilizando la información del vector gradiente, que indicase si la ventana de detección podría ser una cara rotada, con lo que también se podrían detectar caras con rotaciones.

5. CONCLUSIONES Y FUTUROS TRABAJOS DE INVESTIGACIÓN

Apéndice A

Optimizaciones al algoritmo AdaBoost

A.1. Introducción

El algoritmo AdaBoost permite generar clasificadores complejos a partir de una combinación de clasificadores simples y fue utilizado por Viola y Jones como base para entrenar su clasificador de caras en [27]. El algoritmo original AdaBoost está explicado en la figura 3.5. Sin embargo, la versión que utilizan Viola y Jones está adaptada para generar un clasificador de múltiples capas. Dicha adaptación se puede consultar en la figura A.1.

Dada la importancia de ese algoritmo y el tiempo necesario para entrenar un clasificador, cualquier mejora en tiempo que se consiga será muy útil. Sobre todo, como en nuestro caso, si se pretenden generar varios clasificadores para compararlos entre sí.

En esta investigación se han encontrado tres puntos clave que afectan no sólo al rendimiento del clasificador, sino también a la escalabilidad del mismo. Esos tres puntos clave son: la reducción del umbral, el uso de las muestras negativas y el entrenamiento de los clasificadores débiles. Muchos artículos plantean solucionar estos problemas, pero la mayoría modifica el algoritmo de entrenamiento sacrificando algo de eficacia por una mayor velocidad de entrenamiento. En nuestro caso hemos comprobado que se puede obtener un aumento de la velocidad de entrenamiento sin sacrificar eficacia utilizando una serie de técnicas que se van a explicar a continuación. Además de dichas técnicas, también se explica la fórmula elegida para reducir el umbral de capa del clasificador,

- $EDR = DR_{capa}$, $EFR = FR_{capa}$, $FR_0 = DR_0 = 1$
- Mientras $FR_t > FR_{target}$, realizar los siguientes pasos:
  1. Si $DR_t >= EDR$ y $FR_t <= EFR$, la capa actual está terminada y creamos una nueva:
     a) Generar l muestras negativas que sean detectadas como positivas en la capa actual.
     b) Establecer el peso inicial de cada muestra a $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_1 = 0, 1$, donde m es el número de muestras positivas y l el número de muestras negativas.
     c) $t = t + 1$, $EDR = DR_{t-1} * DR_{capa}$, $EFR = FR_{t-1} * FR_{capa}$
  2. Normalizar los pesos: $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$.
  3. Por cada característica, entrenar un clasificador débil con el error acumulado mínimo utilizando los pesos normalizados y según la siguiente ecuación: $\epsilon_{t,j} = \sum_i w_{t,i} \left| h_j(x_i) - y_i \right|$
  4. Elegir el clasificador débil, h_t, con el menor error ϵ_t y añadirlo a la capa actual.
  5. Establecer el umbral de la capa actual al máximo, 1.
  6. De forma iterativa:
     a) Utilizar el clasificador con el set de pruebas y calcular DR_t y FR_t.
     b) Si $DR_t >= EDR$ detener la iteración.
     c) Reducir el umbral de la capa actual
  7. Actualizar el peso de cada muestra: $w_{t+1,i} = w_{t,i}\beta_t^{1-\epsilon_i}$. siendo $e_i = 0$ si x_i se ha clasificado correctamente, $e_i = 1$ en caso contrario y $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

Siendo DR la tasa de detección, FR la tasa de falsos positivos y EDR y EFR las tasas objetivo para la capa en curso.

Figura A.1: Algoritmo AdaBoost Multicapa.

otro de los puntos clave del algoritmo que no está explicado en ninguno de los artículos analizados.

A.2. Reducción del umbral de detección

Un detalle del entrenamiento muy importante es el que hace referencia a "reducir" el umbral de la capa siendo entrenada hasta alcanzar la tasa de detección deseada. El problema de este detalle es que los autores no lo explican en ninguna parte. Otros detalles, como puede ser el entrenamiento individual de los clasificadores débiles, en

vez de aparecer en el artículo original, aparece en otros artículos de los mismos autores. Sin embargo, la explicación de comor reducir el umbral no se ha encontrado en ningún sitio, por lo que se ha decidido analizar el problema para obtener la solución adecuada.

Para empezar, calcular el umbral sirve para conseguir que el clasificador, hasta la capa actual, tenga la relación deseada entre detecciones y falsos positivos. Es decir, un umbral demasiado alto no permitiría alcanzar la detección objetivo y un umbral muy bajo provocaría una tasa de falsos positivos demasiado alta.

Obtener el umbral adecuado se consigue partiendo del umbral más alto posible, lo que equivaldría a sensibilidad nula, y reducirlo poco a poco hasta alcanzar el punto deseado. El problema radica en seleccionar adecuadamente el tamaño de los pasos. Si dichos pasos son pequeños, permiten ajustar bien el umbral a costa de un mayor número de iteraciones. Sin embargo, si son muy grandes permite realizar un menor número de iteraciones a costa de un ajuste más basto del umbral. Si excluimos el problema de el número de iteraciones, gracias a un técnica que se explica en la sección A.3, queda simplemente averiguar que paso es el más pequeño que tendría sentido utilizar.

Por recordar, el clasificador AdaBoost devuelve positivo según la siguiente fórmula:

$$h(x) = \begin{cases} 1 & \text{si } \sum_{n=1}^{N} \alpha_n h_n(x) \geq t \sum_{n=1}^{N} \alpha_n \\ 0 & \text{en caso contrario} \end{cases} \tag{A.1}$$

siendo N el número de clasificadores y t el umbral del clasificador. Para calcular el tamaño paso para reducir el umbral vamos a partir de un supuesto, en el que un clasificador AdaBoost está compuesto por tres clasificadores débiles diferentes, cada uno con α igual a 2, 3 y 5, respectivamente. Por lo tanto, los diferentes pasos de umbral necesitan por lo menos cubrir las diferentes conbinaciones posibles de clasificadores. Dichas combinaciones las podemos ver en la tabla A.1 y, tal como se puede ver, el tamaño mínimo de paso de umbral debería ser 0.1.

Ese paso de umbral corresponde al calculado con la siguiente fórmula:

$$step = \frac{M.C.M.(\alpha_1, \alpha_2, ..., \alpha_n)}{\sum_{n=1}^{N} \alpha_n}$$

Sin embargo, la realidad es que los clasificadores tienen como valor α números reales como 0,334 o 0,112. En la tabla A.2 se pueden ver los distintos umbrales necesarios si

Clasificadores activos	$\sum_{n=1}^{N} \alpha_n h_n(x)$	Umbral
A	2	0,2
B	3	0,3
C	5	0,5
A, B	5	0,5
A, C	7	0,7
B, C	8	0,8
A, B, C	10	1

Tabla A.1: Diferentes posibilidades de activación de un clasificador AdaBoost con α enteros.

los α fueran igual a 0,22, 0,33 y 0,55 respectivamente.

Clasificadores activos	$\sum_{n=1}^{N} \alpha_n h_n(x)$	Umbral
A	0,22	0,2
B	0,33	0,3
C	0,55	0,5
A, B	0,55	0,5
A, C	0,77	0,7
B, C	0,88	0,8
A, B, C	1,1	1

Tabla A.2: Diferentes posibilidades de activación de un clasificador AdaBoost con los α reales.

Para esta situación no podemos recurrir a la misma fórmula, ya que no se puede calcular el Mínimo Común Múltiplo de números reales. Por lo tanto, para aproximar el valor, se ha recurrido a la siguiente fórmula:

$$step = \frac{\frac{1}{N} MIN(\alpha_1, \alpha_2, ..., \alpha_n)}{\sum_{n=1}^{N} \alpha_n}$$

Para el caso planteado, por ejemplo, el salto de umbral óptimo sería de 0,10 y con esta fórmula se obtiene 0,05. Esta diferencia provoca el doble de comprobaciones, sin embargo, y dado que gracias a la próxima técnica aumentar el número de iteraciones no supone un coste considerable, se puede garantizar que se analizarán todas los umbrales de activación posibles.

A.3. Comprobación de eficacia en cada capa

Esta técnica es la más sencilla, y no afecta de ningún modo a la eficacia del clasificador. El punto que optimiza es la comprobación de la tasa de detección y de falsos negativos, más concretamente el punto 6.b en el algoritmo de la figura A.1. Tal como se ha explicado anteriormente, el umbral se va reduciendo hasta lograr las tasas de detección y de falsos positivos objetivo.

En particular, en esta investigación se ha utilizado el paso de umbral planteado en la sección anterior. El problema radica en que el gran número de iteraciones que provocaría utilizar un paso tan pequeño, convierte en un cuello de botella calcular el rendimiento utilizando las millones de muestras de prueba necesarias.

Sin embargo, si se divide el número de muestras de validación en positivas y negativas se puede comprobar lo siguiente:

- El número de muestras de validación negativas es, como mínimo, la inversa de la tasa de falsos positivos deseada. Es decir, para una tasa de 10^{-6} se necesitarían un mínimo de 10^6 muestras negativas. Además, cabe la posibilidad de aumentar dinámicamente dicho número según va aumentando la complejidad del clasificador.

- El número de muestras de validación positivas suele ser igual al número de muestras de entrenamiento positivas. De entre todos los estudios el valor más alto, obtenido de [27], es de 15.000 muestras a dividir entre validación y entrenamiento. A diferencia del número de muestras negativas, este valor es constante ya que no se pueden generar nuevas muestras positivas de forma aleatoria.

Siguiendo con el planteamiento anterior, si para calcular el umbral sólo calculamos la tasa de detecciones, el número de comprobaciones sería, en todas las iteraciones, igual al número de muestras positivas. Únicamente en la última iteración, cuando el umbral haya descendido hasta alcanzar la tasa de detección deseada, se tendría que calcular la tasa de falsos positivos, para averiguar si se ha terminado de entrenar la capa actual del clasificador.

A.4. Consumo de las muestras negativas

Según el artículo original, al añadir una capa se realizan dos pasos importantes con respecto a las muestras negativas:

1. Eliminar todas aquellas muestras negativas de entrenamiento que la capa anterior ha clasificado correctamente.

2. Rellenar el conjunto de muestras negativas con muestras del conjunto de validación clasificadas erróneamente.

3. Buscar aleatoriamente nuevas muestras negativas para rellenar el conjunto de muestras negativas de validación

Para poder realizar esta tarea se requiere que las muestras negativas, tanto de entrenamiento como de validación, tienen que estar almacenadas en una estructura de memoria. Sin embargo, almacenar esa información implica un consumo de memoria excesivo.

Por ejemplo, según nuestro diseño de la aplicación de entrenamiento cada muestra contiene los siguientes datos:

- Puntero a la imagen analizada: 8 bytes.
- Posición vertical: 8 bytes.
- Posición horizontal: 8 bytes.
- Anchura de la muestra: 8 bytes.
- Altura de la muestra: 8 bytes.

En total, y teniendo en cuenta que se utiliza una máquina de 64 bits, se necesita un mínimo de 40 bytes por muestra. Con las 10^6 muestras planteadas anteriormente para alcanzar una tasa de falsos positivos de 10^-6, hacen falta 40 MB sólo para las muestras de validación. Si se pretende reducir la tasa de falsos positivos hasta 10^{-7} se necesitarían 400 MB, y si se quiere llegar hasta 10^{-8}, 4000 MB. Es decir, la escalabilidad de este entrenamiento está limitada por el tamaño máximo de la estructura que almacene las muestras negativas de validación.

Nuestra opción es separar las muestras de entrenamiento de las de validación. Es decir, con cada capa nueva, en vez de buscar las nuevas muestras de entrenamiento dentro del cojunto de muestras de validación, se buscan de forma aletoria falsos positivos utilizando la técnica presentada en la sección 2.2.2. Eliminando, de esta forma, la necesidad de almacenar las muestras negativas de validación.

La validación se haría de una forma similar pero con un estilo de"usar y tirar". Es decir, para cada prueba, se busca aleatoriamente una muestra, se comprueba y se deshecha. Con esta técnica se pueden realizar millones de comprobaciones sin comprometer la escalabilidad del entrenamiento, simplemente si se decide aumentar el número de muestras negativas se aumenta el tiempode validación y no la memoria necesaria.

Esta técnica tiene el inconveniente de afectar al tiempo necesario para realizar la validación. En vez de utilizar de forma repetida muestras obtenidas con anterioridad, hay que obtener una nueva muestra en cada comprobación. Sin embargo, se ha visto que para los rangos de 10^6 muestras, el aumento del tiempo dedicado a obtener los valores aletorios de posición y tamaño es despreciable, siendo de milisegundos. Por comparar, el tiempo necesario para comprobar todas las muestras negativas es del orden de varios segundos.

A.5. Entrenamiento de clasificadores débiles.

El entranamiento de los clasificadores débiles supone el principal cuello de botella del algoritmo de entrenamiento global. Dicho entrenamiento, según plantean Viola y Jones, consiste en utilizar el valor devuelto por la característica que utilice el clasificador para ordenar el listado de muestras de entrenamiento y, a continuación, en una sola pasada se obtiene el umbral que genera el menor error de clasificación.

Ordenar el listado completo de muestras, por cada clasificador débil, se repite en cada iteración. Sin embargo, ya que durante el entrenamiento de una capa no se modifican las muestras de entrenamiento, el resultado de la ordenación no varía de iteración en iteración.

La técnica propuesta consiste en almacenar una copia del grupo de muestras de entremiento ordenada por cada clasificador débil. El problema se traslada, entonces, a disponer de una gran cantidad de memoria como para poder aplicar la técnica . Sin embargo, el aumento de velocidad conseguido es de varios órdenes de magnitud, ya que

el ordenamiento realizado con cada clasificador se realizaría 1 vez por capa, en vez de N veces, siendo N el número de clasificadores en la capa.

Bibliografía

[1] Shumeet Baluja y col. "Efficient face orientation discrimination". En: *International Conference on Image Processing*. 2004, págs. 589-592.

[2] G. Bradski. "The OpenCV Library". En: *Dr. Dobb's Journal of Software Tools* (2000).

[3] Corinna Cortes y Vladimir Vapnik. "Support-vector networks". En: *Machine Learning* 20 (3 1995). 10.1007/BF00994018, págs. 273-297. ISSN: 0885-6125. URL: http://dx.doi.org/10.1007/BF00994018.

[4] Franklin C. Crow. "Summed-area tables for texture mapping". En: *SIGGRAPH Comput. Graph.* 18 (3 1984), págs. 207-212. ISSN: 0097-8930. DOI: http://doi.acm.org/10.1145/964965.808600. URL: http://doi.acm.org/10.1145/964965.808600.

[5] Navneet Dalal y Bill Triggs. "Histograms of Oriented Gradients for Human Detection". En: *In CVPR*. 2005, págs. 886-893.

[6] Yoav Freund y Robert E. Schapire. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1995.

[7] Yoav Freund y Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". En: *J. Comput. Syst. Sci.* 55 (1 1997), págs. 119-139. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504. URL: http://dl.acm.org/citation.cfm?id=261540.261549.

[8] Ralph Gross. "Face Databases". En: *Handbook of Face Recognition*. Ed. por A.Jain S.Li. New York: Springer, 2005.

[9] Bernd Heisele y col. "Hierarchical classification and feature reduction for fast face detection with support vector machines". En: *Pattern Recognition* 36 (2003), págs. 2007-2017.

[10] Erik Hjelmås. *Feature-Based Face Recognition*. 2000.

[11] Chang Huang y col. "Incremental Learning of Boosted Face Detector". En: *ICCV*. 2007, págs. 1-8.

[12] Michael Jones y col. "Detecting Pedestrians Using Patterns of Motion and Appearance". En: *In ICCV*. 2003, págs. 734-741.

[13] kobi Levi, Yair Weiss y Of Good Features. *Learning Object Detection from a Small Number of Examples: the Importance of Good Features*. 2004.

[14] Rainer Lienhart y Jochen Maydt. "An Extended Set of Haar-Like Features for Rapid Object Detection". En: *IEEE ICIP 2002*. 2002, págs. 900-903.

[15] Ce Liu y Heung-Yeung Shum. "Kullback-Leibler Boosting". En: *CVPR (1)*. 2003, págs. 587-594.

[16] Takeshi Mita, Toshimitsu Kaneko y Osamu Hori. "Joint Haar-like Features for Face Detection". En: *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*. ICCV '05. Washington, DC, USA: IEEE Computer Society, 2005, págs. 1619-1626. ISBN: 0-7695-2334-X-02. DOI: http://dx.doi.org/10.1109/ICCV.2005.129. URL: http://dx.doi.org/10.1109/ICCV.2005.129.

[17] B. Moghaddam y A. Pentland. "Probabilistic visual learning for object detection". En: *Computer Vision, IEEE International Conference on* 0 (1995), pág. 786. DOI: http://doi.ieeecomputersociety.org/10.1109/ICCV.1995.466858.

[18] *Fast training and selection of Haar features using statistics in boosting-based face detection*. 2007, págs. 1-7. DOI: 10.1109/ICCV.2007.4409038. URL: http://dx.doi.org/10.1109/ICCV.2007.4409038.

[19] Jonathon P. Phillips y col. "The FERET Evaluation Methodology for Face-Recognition Algorithms". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.10 (2000), págs. 1090-1104. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.44.9852.

[20] Henry Rowley, Shumeet Baluja y Takeo Kanade. "Neural Network-Based Face Detection". En: *Computer Vision and Pattern Recognition '96*. 1996.

[21] Henry Rowley, Shumeet Baluja y Takeo Kanade. *Rotation Invariant Neural Network-Based Face Detection*. Inf. téc. CMU-CS-97-201. Pittsburgh, PA: Computer Science Department, 1997.

[22] Robert E. Schapire y Yoram Singer. "Improved Boosting Algorithms Using Confidence-rated Predictions". En: *Machine Learning*. 1999, págs. 80-91.

[23] Henry Schneiderman. "A Statistical Approach to 3D Object Detection Applied to Faces and Cars". Tesis doct. Pittsburgh, PA: Robotics Institute, Carnegie Mellon University, 2000.

[24] L. Sirovich y M. Kirby. "Low-dimensional procedure for the characterization of human faces". En: *J. Opt. Soc. Am. A* 4.3 (1987), págs. 519-524. DOI: 10.1364/JOSAA.4.000519. URL: http://josaa.osa.org/abstract.cfm?URI=josaa-4-3-519.

[25] Kah kay Sung y Tomaso Poggio. "Example-based learning for view-based human face detection". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), págs. 39-51.

[26] Kah Kay Sung. "Learning and example selection for object and pattern detection". AAI0800657. Tesis doct. 1996.

[27] Paul Viola y Michael Jones. "Robust real-time face detection". En: *International Journal of Computer Vision* 57 (2004), págs. 137-154.

[28] Paul Viola y Michael J. Jones. "Robust Real-Time Face Detection". En: *International Journal of Computer Vision* 57 (2 2004). 10.1023/B:VISI.0000013087.49260.fb, págs. 137-154. ISSN: 0920-5691.

[29] Paul Viola, Michael J. Jones y Daniel Snow. "Detecting Pedestrians Using Patterns of Motion and Appearance". En: *International Journal of Computer Vision* 63 (2 2005). 10.1007/s11263-005-6644-8, págs. 153-161. ISSN: 0920-5691. URL: http://dx.doi.org/10.1007/s11263-005-6644-8.

[30] Bo Wu y col. "Fast rotation invariant multi-view face detection based on real AdaBoost". En: *In Sixth IEEE International Conference on Automatic Face and Gesture Recognition*. 2004, págs. 79-84.

[31] Jianxin Wu, James M. Rehg y Matthew D. Mullin. "Learning a Rare Event Detection Cascade by Direct Feature Selection". En: *In NIPS*. 2003. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.8770.

[32] Shengye Yan y col. "Locally Assembled Binary (LAB) feature with feature-centric cascade for fast and accurate face detection". En: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage, AK, USA: IEEE, jun. de 2008, págs. 1-7. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587802. URL: http://dx.doi.org/10.1109/CVPR.2008.4587802.

[33] Ming hsuan Yang, Dan Roth y Narendra Ahuja. "A SNoW-Based Face Detector". En: *Advances in Neural Information Processing Systems 12*. MIT Press, 2000, págs. 855-861.

[34] Ming-Hsuan Yang, David J. Kriegman y Narendra Ahuja. "Detecting faces in images: A survey". En: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 24.1 (2002), págs. 34-58.

[35] Cha Zhang y Zhengyou Zhang. *A Survey of Recent Advances in Face detection*. 2010.

[36] Lun Zhang y col. "Face Detection Based on Multi-Block LBP Representation". En: *ICB*. 2007, págs. 11-18.