# DETECTOR AUTOMÁTICA DE PERSONAS EN IMÁGENES

**Autores** Ayesa Moneo, Ander

Fernandez Ortega, Unai Javier

Soba Jimenez, Iñigo

upna Universidad Pública de Navarra

# ÍNDICE

# PREPROCESAMIENTO DE IMÁGENES

Primero de todo convertimos a grises tanto las imágenes de train como las de test, hemos decidido trabajar con imágenes en escala de grises por comodidad y reducir el tiempo de ejecución. Una vez convertidas a escala de grises, se aplica una ecualización del histograma para realzarlas.

Se ha comprobado que, al no aplicar ecualización del histograma a las imágenes, el porcentaje de acierto disminuye un poco.

Para aumentar el número de imágenes de train de la clase positiva, se han duplicado dándole la vuelta a la imagen, rotándola sobre el eje vertical.

Respecto a las imágenes de train de la clase negativa, al ser de diferente tamaño a las de train (134x70), se ha implementado una función para extraer 3 subimágenes de cada imagen.

# EXTRACCIONES DE CARACTERISTICAS

Se ha usado la función **HOG** para la extracción de características para cada imagen.

Los parámetros más destacables que le hemos pasado son:

Los parámetros están todos por defecto, por recomendación del profesor, el único que hemos modificado ha sido el pixel_per_cell a fin de reducir el número de características y mejorar el tiempo de ejecución.

orientations = 8

pixels_per_cell = (32, 32)

cells_per_block = (1, 1)

La función HOG devuelve una matriz de características.

Las clases de las matrices de características son: clase positiva (**0** - persona) y clase negativa (**1** – no persona).

La matriz de características de **train** esta compuesta por la matriz de características de train tanto de la clase positiva como de la negativa.

De esta matriz de características se han extraído 150 filas aleatorias que componen la matriz de **validación**.

Por último, y poder testear nuestros modelos, se han creado dos matrices de características para **test**, una para la clase positiva y otra para la negativa.

Dimensiones de las **matrices de características**:

Matriz de características de train (clase positiva): (1125, 64)

Matriz de características de train (clase negativa): (600, 64)

Matriz de características de test (clase positiva): (50, 64)

Matriz de características de test (clase negativa): (10, 64)

El **ratio** de imágenes de clase positiva y negativa es recomendable que sea 2:1.

Para entrenamiento hay:

- Numero imágenes clase positiva: 1125
- Numero imágenes clase negativa: 600

Para test:

- Numero imágenes clase positiva: 50
- Numero imágenes clase negativa: 10

# MODELOS DE ENTRENAMIENTO

Se han implementado dos modelos para entrenar y clasificar los datos.

Primeramente, se ha implementado un modelo basado en **regresión logística**, en este modelo no hemos indicado ningún tipo de parámetro al constructor.

Después se implemento el **SVM** (Support Vector Machine), con el que se ha conseguido mejorar un poco la clasificación de los datos.

Los parámetros usados son:

C: parámetro de regularización (10)

max_iter: para fijar un numero máximo de iteraciones (10000)

Tras varias pruebas, los **resultados** que nos da en train y test para cada modelo son:

Accuracy train (lr): 83.3%

Accuracy train (svm): 84%

Accuracy test para la clase positiva (lr): 98%

Accuracy test para la clase negativa (lr): 80%

Accuracy test para la clase positiva (svm): 90%

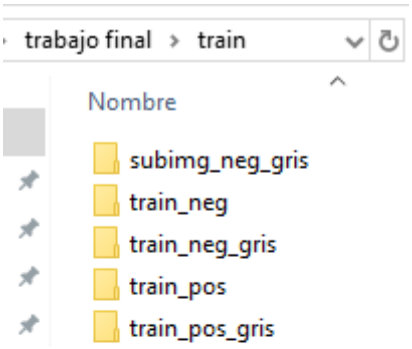
Accuracy test para la clase negativa (svm): 90%

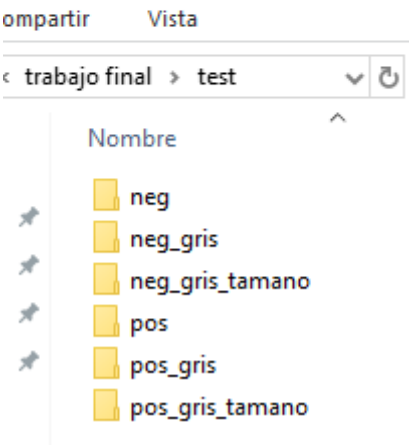
# DETECCIÓN

En las pruebas que se han llevado a cabo, se han utilizado imágenes de un tamaño superior a 134x70.

Para el correcto funcionamiento del detector, se deben de tener unas carpetas (vacías) ya creadas, para el almacenamiento de las imágenes en gris, redimensionadas, etc.

El **esquema** es:




La función principal es **analisis()**, esta llamará a **ventana_deslizante()** y **non_max_suppression()**.

La función ventana_deslizante() recorre la imagen de entrada utilizando una ventana deslizante creando un vector de ventanas y las devuelve.

Esta función, es llamada desde otra función llamada analisis().

En esta función aplicamos una **pirámide gaussiana** para ir reduciendo la imagen original en otras más pequeñas y aplicamos la ventana deslizante en cada una de estas.

De cada una de las ventanas resultantes de llamar a la función ventana_deslizante() obtenemos mediante HOG el vector de características. Posteriormente, con el modelo seleccionado (regresión logística o SVM) predecimos si pertenece a la clase positiva (persona) o a la clase negativa (no persona). Después con aquellas ventanas que han sido detectadas como clase positiva, creamos los rectángulos candidatos a contener una persona.

Por último, aplicamos la supresión de no máximos (utilizando una función ya creada) para evitar recuadros redundantes.

Como se puede comprobar, esta última función no termina de eliminar todos los recuadros redundantes.

El resultado de la ejecución del algoritmo, dará como resultado dos imágenes.

La primera imagen es el resultado sin aplicar la supresión de no máximos, la segunda aplicando la supresión.

El **tiempo de ejecución** del algoritmo es:

- Para regresión logística: 32s
- Para SVM: 31s

Algunos tiempos de ejecución destacables de otras funciones:

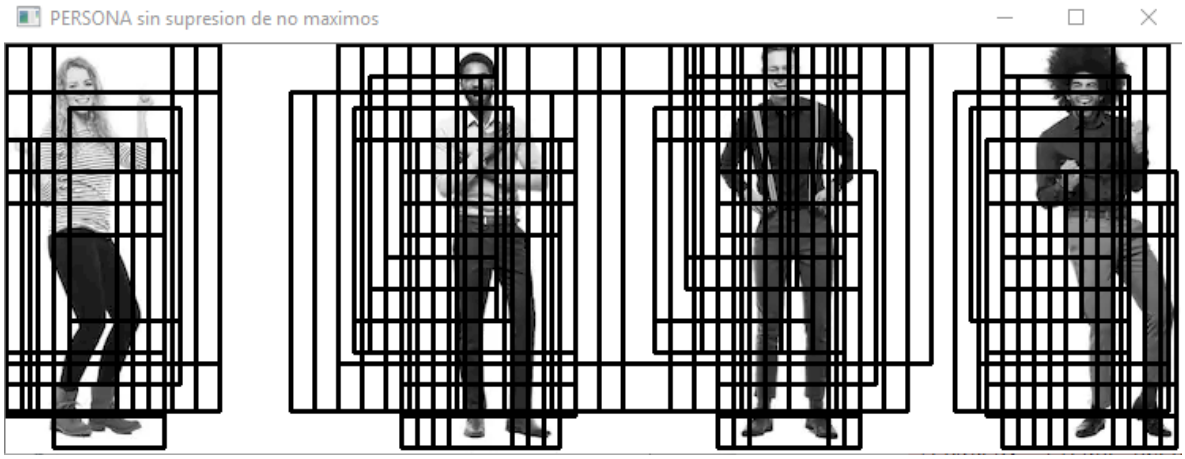
- convertir_a_gris(): 9.14s
- análisis(): 0.98s

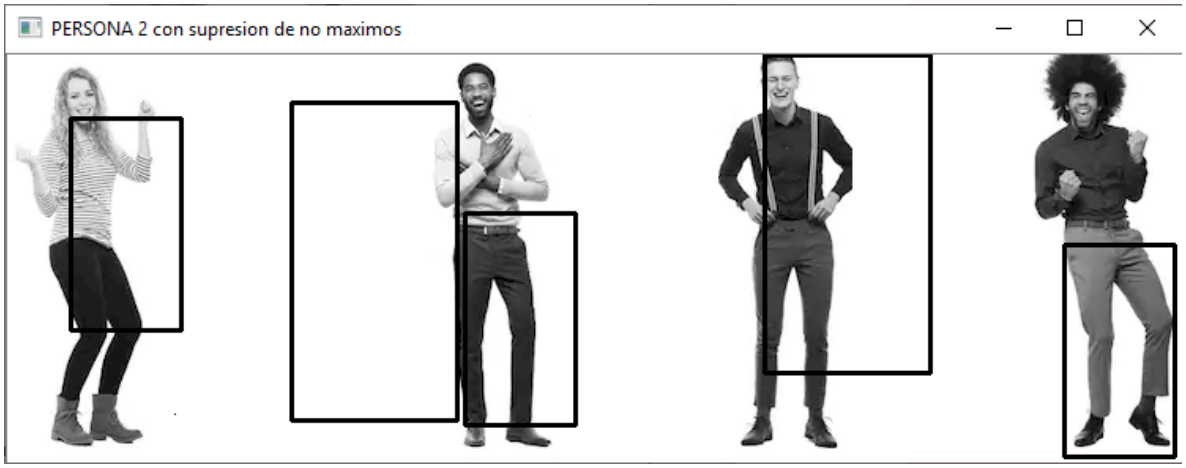
Para reducir notablemente el tiempo de ejecución, se podría trabajar con un menor numero de imágenes, o un menor número de características.

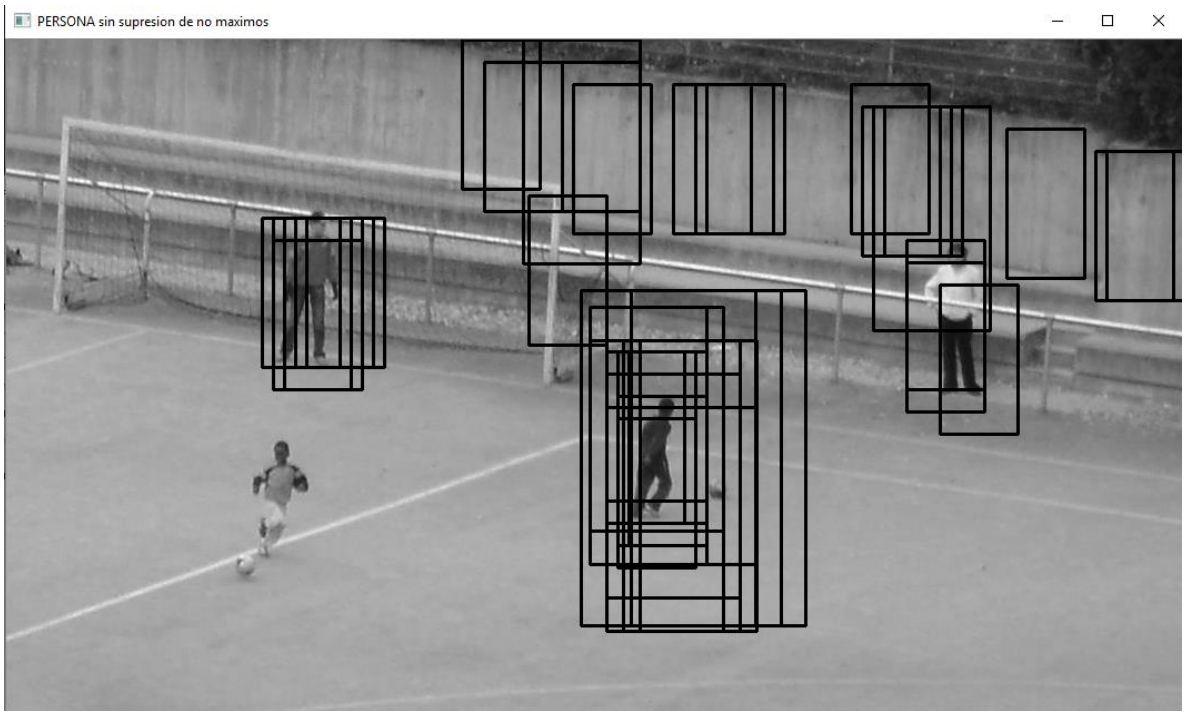
Algunas de las **imágenes resultantes** de la ejecución del detector:

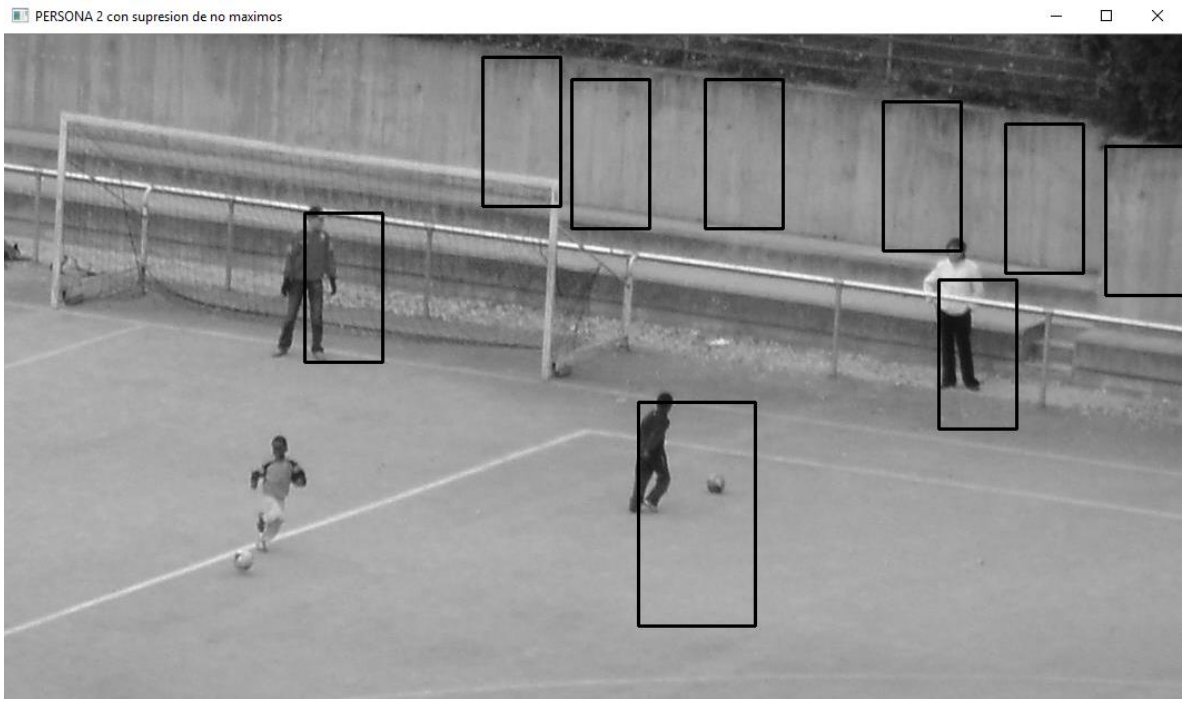
Imagen **sin** supresión de no máximos


Imagen **con** supresión de no máximos


PERSONA sin supresion de no maximos


PERSONA 2 con supresion de no maximos