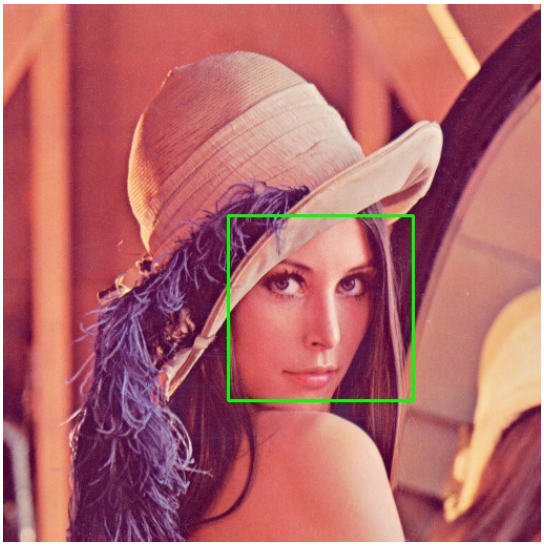
Procesado de imagen y visión por computador

DETECTOR DE CARAS

1. Introducción al detector

Como punto de arranque de la práctica, se facilita un código en Python que permite familiarizarse con el uso del detector de caras implementado en OpenCV. Este detector está basado en el modelo de *Viola and Jones,* cuyos fundamentos se describen en la documentación de la práctica. Se recomienda consultar la documentación de OpenCV relativa a este modelo de detección[1] para el desarrollo de la práctica.

Para comenzar, deberá ejecutar el script proporcionado `detect_demo.py` sobre la imagen de prueba `lena.jpg`. Como resultado de la ejecución, debería visualizarse la siguiente detección asociada a la misma.



Una vez ejecutado el código satisfactoriamente, es momento de realizar un análisis del mismo en profundidad. Responda a las siguientes preguntas:

1. ¿Qué función concreta es la encargada de realizar la detección?
2. ¿Qué parámetros recibe?
3. ¿Qué modelo de detector pre-entrenado se está utilizando?
4. ¿Por qué cree que se realiza una ecualización de la imagen y una conversión de la misma a escala de gris?

Una vez respondidas las preguntas planteadas se puede pasar a probar el detector con el resto de imágenes proporcionadas. ¿Funciona el detector correctamente con todas ellas? Razone sus respuestas.

Es interesante entender el significado de los parámetros que se le pasan al clasificador. Los parámetros que afectan más al funcionamiento son:

- *scaleFactor*
- *minNeighbors*
- *minSize*

Para comprender qué efecto tiene cada uno de los parámetros se deberá probar con distintos valores

1 https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html

en distintas imágenes y ver los efectos que se producen sobre precisión y velocidad (también es interesante buscar información sobre los mismos en la web para ver si los resultados obtenidos concuerdan con lo publicado). Para valorar la velocidad de ejecución del detector debe añadir en el código un temporizador en el script[2].

2. Detector de caras en tiempo real

Basándose en la implementación proporcionada en el script de demo, el alumno debe implementar ahora una solución más avanzada que permita:

1. Realizar la detección de caras sobre un flujo de vídeo tomado a partir de una webcam.
2. El sistema debe realizar una detección en cada frame del vídeo, y visualizará todas y cada una de las caras detectadas.
3. La solución propuesta debe permitir visualizar mas de una detección cuando haya varias caras en la escena. Las diferentes detecciones simultáneas se visualizarán con colores diferentes.
4. La aplicación debe guardar un vídeo de resultado, el cual será el material a entregar por el alumno.

Para analizar la robustez de la solución propuesta, el alumno debe probar la fiabilidad del detector cuando en el vídeo se dan las siguientes situaciones:

1. Cambios de iluminación bruscos.
2. Oclusiones parciales y severas: pruebe a tapar la cara gradualmente mientras el detector está funcionando.

3. Efecto de pixelado en caras (opcional)

El pixelado de rostros es muy utilizado para proteger la identidad de las personas, especialmente menores, en difusión de imágenes. Basándose en el código proporcionado en el primer apartado, el alumno debe implementar una solución para añadir de forma automática el efecto de pixelado sobre la detección de caras. El efecto se basa en dividir la zona del rostro a pixelar en una celda de bloques cuadrados del tamaño deseado (por ejemplo, 20x20 píxeles) y calcular el color promedio en cada celda. Cada uno de estos cuadrados es pintado con el color promedio.

Para ello, debe codificar en el script facilitado en la práctica una nueva función `def pixelado(img,rect,tam_bloque),` la cual recibirá como parámetros de entrada la imagen a procesar, las coordenadas del rectángulo de la detección de cara para pixelar y la dimensión del cuadrado del pixelado en píxeles.

2 https://docs.opencv.org/master/dc/d71/tutorial_py_optimization.html

4. Detección de ojos (opcional)

En el código base con el que ha trabajado en el apartado 2 incluya la detección de ojos utilizando el detector *haarcascade_eye.xml.* Será importante que solo se le pase al detector de ojos la ROI de la cara detectada previamente mediante el detector de cara. La representación del rectángulo alrededor de los ojos deberá ser del mismo color que la de la cara correspondiente.

Material a entregar

1. Una memoria con las respuestas a las preguntas planteadas en el enunciado de la práctica y las pruebas que el alumno haya realizado.
2. Un vídeo donde se vea el detector funcionando en tiempo real con imágenes tomadas desde la webcam y el código Python desarrollado. Respuesta a las preguntas sobre cambios de iluminación y oclusiones.
3. Apartado opcional (video/imágenes captado desde la webcam y el código desarrollado incluyendo la función para el pixelado)
4. Apartado opcional (video/imágenes captado desde la webcam y el código desarrollado).