Proyecto de investigación sobre detección de caras mediante OpenCV

Alejandro Rodríguez Rodríguez Pablo Cano Navajas
Salvador Caballero Macías

16 de abril de 2025

Índice

Resumen

Este documento detalla las diferentes implementaciones que se han llevado a cabo durante la implementación de las mismas, asi como los resultados obtenidos y las conclusiones a las que se ha llegado. El proyecto se ha realizado empleando el repositorio[1] que se detalla en el libro escogido[2]. De la documentacion escogida, se ha seleccionado el capítulo número 5, que trata sobre la detección de caras, tanto en imágenes como en captura en tiempo real, así como las implementaciones de mejoras de los algoritmos que se tratan en el libro para detectar un mayor número de caras o distintos objetos que cumplan unas restricciones que se impongan.

1. Introducción

2. Planteamiento teórico

En esta sección se detalla el planteamiento teórico del capítulo 5 del libro[2], que se centrará en los fundamentos de las cascadas de Haar mediante el uso de OpenCV para la detección de caras. Se dividirá en 3 subapartados, correspondientes a los subapartados del propio capítulo:

2.1. Conceptualizing Haar Cascade Data

subseccion1

2.2. Using OpenCV to Perform Face Detection

2.3. Improving the Haar Cascade Classifier

subseccion3

3. Implementación

3.1. Conceptualizing Haar Cascade Data

implementacion1

3.2. Using OpenCV to Perform Face Detection

En esta sección se explica cómo detectar caras tanto en imagenes como en una entrada de vídeo en tiempo real (e.g. una cámara de vídeo). OpenCV proporciona herramientas avanzadas para el procesamiento de imágenes y la detección de objetos mediante el uso de clasificadores en cascada de Haar.
La detección de caras se realiza utilizando archivos XML que contienen los datos preentrenados para identificar características faciales específicas. Estos clasificadores se cargan mediante la función `cv2.CascadeClassifier`, que permite aplicar los modelos a imágenes o fotogramas de vídeo. Para detectar caras en imágenes estáticas, se utiliza el script `0_stillImageFaceDetection.ipynb`. Este script carga una imagen, aplica un clasificador en cascada y detecta las caras presentes en la misma. El proceso incluye los siguientes pasos:

- Carga de la imagen mediante `cv2.imread`.
- Conversión de la imagen a escala de grises para optimizar el rendimiento del clasificador.
- Uso del clasificador `haarcascade_frontalface_default.xml` para detectar caras.

El script `1_cameraFaceDetection.ipynb` implementa la detección de caras en tiempo real utilizando la cámara del dispositivo. Este proceso incluye:

- Captura de vídeo en tiempo real mediante `cv2.VideoCapture`.
- Aplicación del clasificador en cascada a cada fotograma del vídeo.
- Detección de características adicionales, como ojos, utilizando el archivo `haarcascade_eye.xml`.

3.3. Improving the Haar Cascade Classifier

implementacion3

4. Experimentación

5. Manual de usuario

Para el correcto funcionamiento de los scripts, es necessario la instalación de los siguientes paquetes:

- **Windows 7, MacOS 10.7 o superior**.
- **Python 3.8 o superior**. Para instalar Python, se recomienda instalar la versión más reciente accediendo a la página web Python.org. Haremos click en el botón de descarga y se descargará automáticamente el ejecutable. Si fuese necesario otra versión de python, en la misma págna se puede descargar la versión que se necesite.
  Si se tiene instalado un sistema operativo distinto a windows, se puede acceder al enlace de versiones macOS y descargar la que sea necesaria siempre que cumppla con los requisitos de instalación.
- **OpenCV 4.0 o superior**. Para instalar la version 4.0 o superior de OpenCV, se recomienda usar el gestor de paquetes `pip`. Para ello, se abre una terminal y se ejecuta el comando `pip install opencv-python`. Si la máquina opera con macOS, se recomienda usar `brew`, escribiendo el comando `brew install opencv`.
- **NumPy 1.16 o superior**. Para instalarlo, en una ventana de comandos escribiremos `pip install numpy` (para un entorno Windows) o `brew install numpy` (para un entorno macOS).
- **Scipy 1.1 o superior**. Para instalarlo, en una ventana de comandos escribiremos `pip install scipy` (para un entorno Windows) o `brew install scipy` (para un entorno macOS).

6. Conclusiones

7. Autoevaluación de cada miembro

7.1. Autoevaluación de Alejandro

7.2. Autoevaluación de Pablo

7.3. Autoevaluación de Salvador

8. Tabla de tiempos

8.1. Tabla de tiempos del grupo

8.2. Tabla de tiempos de Alejandro

8.3. Tabla de tiempos de Pablo

8.4. Tabla de tiempos de Salvador

Referencias

[1] *Repositorio del proyecto*, disponible en GitHub

[2] Joseph Howse, Joe Minichino, *Learning OpenCV 4 Computer Vision with Python 3*", Third edition, Packt Publishing, pp. 1-372, 2020