Leonardo Serrentino, U0248350L

Documentación de Códigos

FaceBlurring.py

El código comienza importando la biblioteca *cv2* de OpenCV para manejar el procesamiento de imágenes. Luego, inicia la transmisión de video desde la cámara predeterminada con *cap = cv2.VideoCapture(0)*. Se inicializan las variables *x, y, w y h*, estableciendo las dimensiones y coordenadas para el desenfoque de los rostros.
El bucle principal (*while*) continúa siempre que la variable ret sea *True*, lo cual sucede cuando la captura de un fotograma se realiza con éxito. Dentro del bucle, el código lee el fotograma actual de la cámara a través de *ret, image = cap.read()*.
Luego, utiliza el *clasificador de Haar* para detectar los rostros presentes en el fotograma actual. La *región de interés (ROI)* se define basándose en las coordenadas de los rostros detectados, aplicando un *desenfoque gaussiano* a estas regiones. El resultado del desenfoque se superpone a la imagen original para crear el efecto de desenfoque de los rostros.
Finalmente, el código muestra el fotograma procesado con los rostros desenfocados a través de *cv2.imshow("Stream", image)* y verifica si se presiona la tecla "*Esc*" para detener el bucle.
Una vez que se interrumpe el bucle, el programa libera los recursos utilizados para la captura de video mediante *cap.release()* y cierra todas las ventanas abiertas mediante *cv2.destroyAllWindows()*.

FalseIdentity_Cascade.py

Este código en Python utiliza la biblioteca *OpenCV (cv2)* para segmentar rostros de una imagen proporcionada por el usuario o de una imagen capturada desde una cámara. Después de preguntar al usuario si desea tomar una foto o cargar una imagen, lleva a cabo varias operaciones:
- Si el usuario elige tomar una foto *(task == 1)*, el código activa la cámara, captura un solo fotograma y guarda la imagen en un archivo *'images/image.png'*.
- Si el usuario elige cargar una imagen *(task == 2)*, solicita al usuario la ruta de la imagen y la carga para su procesamiento.

Detección de rostros en la imagen o imagen capturada:
Utiliza el *clasificador de Haar* preentrenado para detectar rostros en la imagen a través de *cv2.CascadeClassifier*. Convierte la imagen a *escala de grises* para reducir el cálculo durante la detección de rostros (*cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)*).
Segmentación del rostro: Una vez que se detectan los rostros, extrae y guarda la región del rostro en un archivo *'images/segmented_face.png*'.
Sustitución del rostro con una identidad falsa:
Inicia un *flujo de video* desde la cámara (*cap.read()*) y detecta rostros en el video mediante el *clasificador de Haar*. Extrae la región del rostro detectado y *redimensiona* una imagen previamente guardada como "*identidad falsa*".
Sustituye la región del rostro en el *fotograma de video* con la imagen de la "identidad falsa".
Visualización del video en tiempo real con rostros sustituidos: Muestra el video en tiempo real con los rostros sustituidos utilizando cv2.imshow.
El ciclo continúa hasta que se presiona la tecla "*Esc*", momento en el cual el programa libera los recursos de la cámara y cierra todas las ventanas.

FalseIdentity_DeepLearning.py

Este código en Python utiliza las bibliotecas *mediapipe*, *cv2*, *pandas*, *numpy y math* para procesar imágenes de rostros y superponer segmentos de rostros en tiempo real desde una cámara web. Primero, se importan las bibliotecas necesarias, se establecen *variables* para *los landmarks faciales* y se definen métodos útiles, como la *superposición de imágenes* y *cálculos de ángulos y distancias*.

A continuación, el código carga una imagen y detecta y segmenta la cara en ella. Luego, se procesan los *landmarks faciales* usando el modelo de *Mediapipe FaceMesh*, y se extraen *coordenadas* y se almacenan las *distancias entre landmarks* para su uso posterior.

Después de obtener los landmarks y crear una *máscara* con ellos, se genera una *imagen segmentada* con la *cara recortada* y se guarda.

La parte principal del código inicia la captura de la cámara web, detecta los landmarks faciales en tiempo real y superpone la imagen segmentada en los rostros detectados en el video en vivo. El proceso implica ajustar la escala y la rotación de la imagen segmentada para que se ajuste y se superponga correctamente sobre los rostros en el video en tiempo real.

El ciclo continúa hasta que se presiona la tecla "*Esc*", momento en el que se cierra la captura de la cámara web y se destruyen todas las ventanas