Penjelasan Code

1. Import Library

import numpy as np

import cv2

**NumPy** adalah untuk mengolah array dan operasi matematika. Dalam konteks ini, NumPy digunakan bersama OpenCV untuk memanipulasi array gambar.

**OpenCV** (cv2) untuk pemrosesan gambar dan video real-time.

1. Import dan load klasifikasi Haar Cascade

face\_cascade = cv2.CascadeClassifier('Cascades/haarcascade\_frontalface\_default.xml')

eye\_cascade = cv2.CascadeClassifier('Cascades/haarcascade\_eye.xml')

smile\_cascade = cv2.CascadeClassifier('Cascades/haarcascade\_smile.xml')

**CascadeClassifier**: OpenCV menggunakan objek CascadeClassifier untuk mendeteksi objek dalam gambar. Objek ini memerlukan file XML yang berisi data yang telah dilatih untuk mendeteksi fitur tertentu seperti wajah, mata, atau senyuman. File XML ini dikenal sebagai classifier atau detektor.

* **haarcascade\_frontalface\_default.xml**: File XML yang digunakan untuk mendeteksi wajah frontal (menghadap kamera).
* **haarcascade\_eye.xml**: File XML yang digunakan untuk mendeteksi mata.
* **haarcascade\_smile.xml**: File XML yang digunakan untuk mendeteksi senyuman.

1. Membuka Video Capture

cap = cv2.VideoCapture(0)

cap.set(3, 640)

cap.set(4, 480)

Langkah ini membuka koneksi ke kamera dan mengatur ukuran frame yang diambil dari kamera

1. Memulai Loop untuk Membaca Frame Video

while True:

    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

Pada langkah ini, setiap frame video diambil secara berurutan dari kamera, dan kemudian dikonversi ke grayscale untuk mempersiapkannya untuk proses deteksi fitur lebih lanjut.

1. Mendeteksi Wajah

faces = face\_cascade.detectMultiScale(

        gray,

        scaleFactor= 1.3,

        minNeighbors= 5,

        minSize= (30, 30)

    )

Langkah ini memindai gambar grayscale untuk menemukan wajah. Jika wajah ditemukan, detectMultiScale akan mengembalikan koordinat dan ukuran persegi panjang yang mengelilingi setiap wajah yang terdeteksi.

1. Mendeteksi mata dan senyum di setiap wajah yang ditemukan

for (x, y, w, h) in faces:

        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

        roi\_gray = gray[y:y + h, x:x + w]

        roi\_color = frame[y:y + h, x:x + w]

        eyes = eye\_cascade.detectMultiScale(

            roi\_gray,

            scaleFactor= 1.5,

            minNeighbors= 15,

            minSize= (25, 25)

            )

        for(ex, ey, ew, eh) in eyes:

            cv2.rectangle(roi\_color, (ex, ey), (ex + ew, ey + eh), (0, 255, 0), 2)

        smile = smile\_cascade.detectMultiScale(

            roi\_gray,

            scaleFactor= 1.5,

            minNeighbors= 15,

            minSize= (25, 25)

            )

        for(sx, sy, sw, sh) in smile:

            cv2.rectangle(roi\_color, (sx, sy), (sx + sw, sy + sh), (0, 255, 0), 2)

pada setiap wajah yang terdeteksi akan dilakukan iterasi untuk menggambar kotak berwarna biru berdasarkan informasi koordinat dan ukuran dari proses sebelumnya

Mendefinisikan Region of Interest (ROI) untuk area wajah yang terdeteksi, baik pada gambar grayscale (roi\_gray) maupun gambar berwarna (roi\_color). ROI ini digunakan untuk mendeteksi mata dan senyuman di dalam area wajah.

Dari ROI yang didapatkan, dapat dimanfaatkan untuk mendeteksi mata dan senyuman, dari setiap wajah dan senyuman yang terdeteksi akan digambarkan kotak berwarna hijau.

1. Menampilkan Hasil Deteksi ke Frame Video

cv2.imshow('video', frame)

Menampilkan hasil frame yang sudah dimanipulasi ke video

1. Menghentikan Program dengan Menekan Tombol 'Esc'

k = cv2.waitKey(30) & 0xFF

    if k == 27:

        break

Langkah ini mengatur jika user ingin menghentikan program maka bisa dilakukan dengan menekan tombol 'Esc' pada keyboard.

1. Membersihkan dan Menutup Semua Window

cap.release()

cv2.destroyAllWindows()

Langkah ini untuk memastikan semua resource sudah dilepas dan memastikan semua window tertutup ketika aplikasi berhenti.