LAPORAN CODE PROGRAM

DETEKSI/PENGENALAN OBJECT WAJAH MENGGUNAKAN PYTHON LIBRARY OPENCV

STMIK AL MUSLIM

Create Creative, Confident & Caring Generations

S1 – Ilmu Komputer

Dosen = Dikky Suryadi S.Kom., M.Kom

Kelompok 1 :

- Abrahamsyah (4122002)
- Aisyah (4122003)
- Karenina Octarisfa (4122008)

Sekolah Tinggi Menengah Informatika Dan Komputer Al Muslim

2025

DAFTAR ISI

BAB 1

1.1 PENDAHULUAN

Dalam dunia pengenalan wajah, pengumpulan dataset yang berkualitas sangat penting untuk meningkatkan akurasi model. Salah satu langkah awal dalam pembuatan sistem pengenalan wajah adalah pengambilan citra wajah yang baik dan melakukan pra-pemrosesan agar lebih siap untuk analisis lebih lanjut.

Deteksi wajah merupakan salah satu teknologi penting dalam pengolahan citra digital yang banyak digunakan dalam berbagai bidang, seperti:

- Keamanan (pengenalan wajah untuk akses kontrol)
- Media sosial (penandaan otomatis dalam foto)
- Analisis ekspresi (misalnya dalam interaksi manusia-komputer)

Proyek ini bertujuan untuk mengembangkan aplikasi yang dapat mengenali wajah dalam video menggunakan teknik deteksi tepi dan pemrosesan gambar untuk meningkatkan akurasi.

1.2 TUJUAN

Paper ini bertujuan untuk menjelaskan implementasi deteksi wajah dalam video real-time menggunakan metode Haarcascade dan Canny Edge Detection serta meningkatkan kualitas gambar wajah dengan berbagai teknik pemrosesan citra.

BAB 2

2.1 METODOLOGI

A. Teknologi yang Digunakan

- Python 3
- OpenCV (cv2) untuk pemrosesan citra
- NumPy untuk operasi matriks
- OS dan Time untuk pengelolaan file dan penjadwalan waktu pengambilan gambar

B. Langkah-Langkah Implementasi

1. Pengumpulan Dataset: Mengambil gambar wajah dengan berbagai kondisi.
2. Pra-pemrosesan Gambar: Konversi grayscale, pengurangan noise, histogram equalization, dan sharpening.
3. Deteksi Wajah: Menggunakan Haarcascade untuk mendeteksi wajah dalam video real-time.
4. Deteksi Tepi: Menggunakan algoritma Canny Edge Detection untuk meningkatkan identifikasi fitur wajah.
5. Uji Aplikasi: Menampilkan hasil deteksi wajah dan deteksi tepi secara real-time.

2.2 IMPLEMENTASI KODE

Berikut adalah cuplikan kode utama yang digunakan dalam program:

```
import cv2
import os
import numpy as np
import time

# Buat folder untuk menyimpan dataset
dataset_folder = "dataset_wajah"
if not os.path.exists(dataset_folder):
    os.makedirs(dataset_folder)

# Inisialisasi webcam
cap = cv2.VideoCapture(0)
```

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

jumlah_gambar = 20  # Jumlah gambar yang ingin dikumpulkan
hitung = 0
waktu_terakhir = time.time()  # Catat waktu awal

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Ubah ke grayscale untuk deteksi wajah lebih akurat
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Deteksi wajah
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5, minSize=(50, 50))

    for (x, y, w, h) in faces:
        wajah = frame[y:y+h, x:x+w]
        wajah = cv2.resize(wajah, (200, 200))  # Ubah ukuran gambar wajah

        # **Peningkatan Kualitas Gambar Berwarna**
        b, g, r = cv2.split(wajah)
        b_eq = cv2.equalizeHist(b)
        g_eq = cv2.equalizeHist(g)
        r_eq = cv2.equalizeHist(r)
        wajah_eq = cv2.merge((b_eq, g_eq, r_eq))  # Gabungkan kembali ke gambar berwarna

        # Buat sharpening filter
        kernel = np.array([[0, -1, 0],
                           [-1, 5, -1],
                           [0, -1, 0]])
```

```
        wajah_sharp = cv2.filter2D(wajah_eq, -1, kernel)  # Terapkan filter sharpening

        # Cek apakah sudah lewat 2 detik sejak foto terakhir
        if time.time() - waktu_terakhir >= 2:
            file_name = os.path.join(dataset_folder, f"wajah_{hitung}.jpg")
            cv2.imwrite(file_name, wajah_sharp)  # Simpan gambar wajah yang sudah diperbaiki
            hitung += 1
            waktu_terakhir = time.time()
            print(f'Gambar {hitung} diambil...")

        # Gambar kotak di sekitar wajah pada frame utama
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # **Deteksi garis di seluruh frame dengan Canny Edge Detection**
    edges = cv2.Canny(gray, 100, 200)

    # **Gabungkan dua tampilan: kamera asli & deteksi tepi**
    frame_resized = cv2.resize(frame, (400, 300))  # Resize agar seimbang
    edges_resized = cv2.resize(edges, (400, 300))  # Resize agar seimbang
    combined_view = np.hstack((frame_resized, cv2.cvtColor(edges_resized,
cv2.COLOR_GRAY2BGR)))  # Gabungkan

    # Tampilkan dalam satu jendela
    cv2.imshow("Pengambilan Wajah & Deteksi Garis", combined_view)

    # Hentikan jika sudah cukup gambar atau tekan 'q'
    if hitung >= jumlah_gambar or cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Bersihkan
cap.release()
cv2.destroyAllWindows()
```

```
print(f"Pengambilan selesai! {hitung} gambar wajah disimpan di folder '{dataset_folder}'.")
```

2.3 HASIL DAN ANALISIS

A. Hasil Pengambilan Gambar

Setelah program dijalankan, sebanyak 20 gambar wajah berhasil disimpan dalam folder dataset_wajah.

B. Analisis Hasil

1. Deteksi Wajah:
    - Haar Cascade berhasil mendeteksi wajah dalam berbagai kondisi pencahayaan.
    - Namun, metode ini masih sensitif terhadap sudut dan ekspresi wajah.
2. Peningkatan Kualitas:
    - Histogram Equalization membantu meningkatkan kontras gambar.
    - Filter sharpening membuat gambar lebih tajam, namun bisa menimbulkan noise jika digunakan berlebihan.
3. Deteksi Tepi:
    - Algoritma Canny Edge Detection berhasil menampilkan kontur wajah dengan baik.
    - Penerapan parameter threshold pada cv2.Canny() dapat disesuaikan untuk mendapatkan hasil yang lebih optimal.

C. Hambatan dan Solusi

- Hambatan: Kadang wajah tidak terdeteksi dengan baik karena pencahayaan buruk.
- Solusi: Menggunakan metode deteksi wajah tambahan seperti Deep Learning (CNN) untuk hasil lebih akurat.

BAB 3

3.1 KESIMPULAN

Proyek ini berhasil mengembangkan sistem deteksi wajah secara real-time menggunakan OpenCV. Dengan memanfaatkan metode Haarcascade untuk deteksi wajah dan Canny Edge Detection untuk identifikasi kontur, sistem dapat mengenali wajah dengan cukup akurat.

3.2 SARAN

- Menggunakan metode deteksi wajah yang lebih canggih seperti Deep Learning (dlib atau MTCNN) untuk akurasi lebih baik.
- Menyimpan gambar dalam format grayscale untuk mengurangi ukuran dataset dan meningkatkan kecepatan pemrosesan.
- Menggunakan augmentasi data untuk memperkaya variasi gambar dalam dataset.