Pengenalan Wajah Berbasis *Eigenvalue* dan *Eigenvector* pada Sistem Keamanan CCTV

Ranashahira Reztaputri 13523007[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523007@std.stei.itb.ac.id, [2]ranashahira.rez@gmail.com

***Abstrak*—CCTV merupakan alat pengawasan yang populer dan dapat meminimalisir kriminalitas. Seiring perkembangan zaman, teknologi dari CCTV terus dikembangkan untuk menghasilkan CCTV modern yang canggih dan semakin memudahkan proses pengawasan. Salah satu inovasi terbaru dari CCTV yang masih dikembangkan hingga saat ini adalah CCTV dengan pengenalan wajah. Dengan adanya pengenalan wajah, akan lebih mudah mencari seseorang yang ingin dicari lewat CCTV karena tidak memerlukan pengecekan secara manual. Maka dari itu, makalah ini ingin mengkaji salah satu pendekatan dalam pengenalan wajah, yakni menggunakan nilai eigen dan vektor eigen. Makalah ini membahas salah satu cara implementasi dari nilai eigen dan vektor eigen bersama dengan kombinasi algoritma lain dalam pengenalan wajah. Tujuan utama dari makalah ini adalah untuk mengevaluasi potensi dari penggunaan nilai eigen dan vektor eigen untuk teknologi yang lebih kompleks. Makalah ini menemukan bahwa nilai eigen dan vektor eigen masih bisa digunakan untuk teknologi yang lebih kompleks, tetapi membutuhkan bantuan dari algoritma dan metode lainnya.**

***Kata Kunci*—CCTV, Nilai Eigen, Pengenalan Wajah, Vektor Eigen**

I. PENDAHULUAN

CCTV (Closed Circuit Television) adalah sistem pengawasan yang telah lama digunakan untuk meningkatkan keamanan di berbagai tempat. Sistem ini mengandalkan kamera yang terhubung ke layar monitor untuk memantau dan merekam kejadian yang terjadi di sekitar area pengawasan. Dengan berkembangnya teknologi, kini CCTV tidak hanya berfungsi sebagai alat pemantau, tetapi juga dilengkapi dengan berbagai fitur canggih, salah satunya adalah pengenalan wajah.

Pengenalan wajah dalam CCTV merupakan salah satu inovasi terkini yang sangat meningkatkan efektivitas sistem keamanan. Teknologi ini memungkinkan kamera CCTV untuk memindai dan menganalisis wajah seseorang, kemudian mencocokkannya dengan data wajah yang tersimpan dalam basis data. Melalui teknologi biometrik ini, setiap individu dapat diidentifikasi dengan menggunakan pola wajah yang unik, yang melibatkan berbagai elemen seperti ekspresi, pola, dan fitur wajah lainnya. Teknologi pengenalan wajah bekerja dengan mengumpulkan data biometrik wajah, kemudian memprosesnya menggunakan algoritma canggih untuk mencocokkan dan memverifikasi identitas seseorang.

CCTV melakukan tracking dan pengenalan wajah menggunakan AI modern. AI tersebut akan mengambil banyak gambar dan mengumpulkan data tersebut untuk dibandingkan dengan database yang sudah dibuat oleh AI dengan menghubungkan dan mencocokkan kumpulan data yang diambil. AI tersebut bekerja dengan menggunakan algoritma seperti Haar Cascades, HOG (Histogram of Oriented Gradients), atau model berbasis Deep Learning (contohnya MTCNN). Selain itu, AI juga menggunakan model seperti Convolutional Neural Networks (CNNs) untuk menghasilkan vektor fitur unik untuk setiap wajah. Walaupun pendekatan AI lebih akurat dan canggih, pendekatan menggunakan Eigenfaces yang berbasis vektor eigen dan nilai eigen juga dapat digunakan untuk pengenalan wajah yang sederhana. Pendekatan menggunakan Eigenfaces akan menjadi pembahasan pada makalah ini.

II. LANDASAN TEORI

A. CCTV (Closed Circuit Television)

Pada sekitaran abad ke-20. Teknologi berbasis video banyak digunakan pada bidang militer dan hiburan. Lalu, muncullah konsep pengawasan berbasis video, yaitu CCTV. Asal-usul adanya CCTV dimulai dari Perang Dunia II. Walter Bruch, seorang insinyur Jerman, mengembangkan CCTV (Closed Circuit Television) pertama untuk memantau peluncuran roket V-2. Pemanfaatan teknologi video untuk pengawasan militer ini menjadi dasar bagi perkembangan teknologi keamanan di masa depan.

Gambar 2.1. Layar CCTV di masa lalu
Sumber: Flickr

CCTV terus mengalami perkembangan setiap tahunnya. Perkembangan CCTV pada tahun 1970-an menandai kemajuan signifikan dalam kamera keamanan dengan diperkenalkannya VTR (Video Tape Recorder) dan VCR (Video Cassette Recorder). Kedua teknologi tersebut memungkinkan perekaman dan pemutaran hasil rekaman sehingga meningkatkan keefektivitasan sistem keamanan. Kemudian, pada tahun 1990-an, CCTV menghadirkan DVR (Digital Video Recorders). DVR berfungsi untuk menyimpan video di hard drive, memperpanjang waktu perekaman, dan menjaga kualitas rekaman yang konsisten. Lalu, pada tahun 2000-an, muncul NVR (Networked Video Recorders) dan server terpusat yang memungkinkan akses jarak jauh dan perluasan sistem yang lebih mudah. Era ini juga menyaksikan munculnya kamera IP, yang terhubung langsung ke jaringan dan menawarkan resolusi lebih tinggi, zoom digital, dan fitur-fitur canggih seperti deteksi gerakan dan penglihatan jarak jauh.

Perkembangan-perkembangan CCTV di masa lampau berhasil membawa kita kepada kecanggihan dari CCTV masa kini. Kamera keamanan saat ini sangat canggih, menampilkan video resolusi tinggi, penglihatan malam, dan deteksi gerakan. Sistem modern sering kali berintegrasi dengan teknologi rumah pintar, memungkinkan pemantauan dan kontrol jarak jauh melalui ponsel cerdas dan perangkat lainnya. Konektivitas hibrida, menggabungkan sistem kabel dan nirkabel, memastikan keandalan dan fleksibilitas. Kemajuan ini menjadikan kamera keamanan penting untuk sistem keamanan rumah dan bisnis yang komprehensif, sehingga memperluas kemampuan dan tujuannya.

Di masa depan, tentunya, akan ada perkembangan lainnya dari CCTV masa sekarang. Beberapa perkembangan CCTV yang akan dilanjutkan di masa depan adalah AI, *deep learning*, NVR, dan analisis video. Teknologi-teknologi tersebut hanyalah beberapa teknologi baru yang muncul untuk membantu mengoptimasi pengawasan video. Seiring dengan kemajuan teknologi, tentunya akan ada kemungkinan munculnya inovasi baru terhadap CCTV.

B. Nilai Eigen dan Vektor Eigen

Dalam aljabar linear, nilai eigen berkaitan dengan vektor eigen dan sering digunakan untuk menganalisis transformasi linier. Nilai eigen merupakan kumpulan nilai skalar tertentu yang berhubungan dengan sistem persamaan linier, biasanya ditemukan dalam konteks persamaan matriks. Sementara itu, vektor eigen, yang juga dikenal sebagai akar karakteristik, adalah vektor bukan nol yang hanya mengalami perubahan skala setelah diterapkan transformasi linier. Faktor yang menskalakan vektor eigen tersebut disebut nilai eigen.

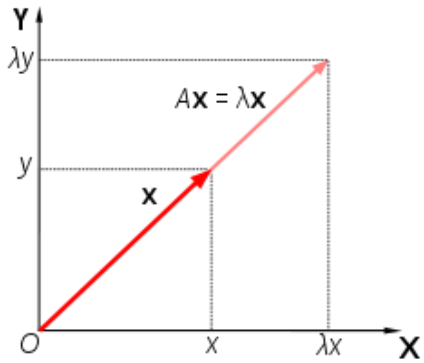
Nilai eigen adalah himpunan skalar khusus yang berhubungan dengan sistem persamaan linier. Kata “eigen” berasal dari Bahasa Jerman yang artinya “asli” atau “karakteristik”. Oleh karena itu, nilai eigen disebut juga sebagai nilai karakteristik atau akar karakteristik. Nilai eigen seringkali digunakan untuk perhitungan matriks. Dalam perhitungan, nilai eigen adalah skalar yang digunakan untuk mengubah vektor eigen. Persamaan dasar dari nilai eigen adalah:

$$Ax = \lambda x \tag{1}$$

Skalar λ disebut nilai eigen dari A, dan x dinamakan vektor eigen yang berkoresponden dengan λ.

Dalam Matematika, vektor eigen terkait dengan nilai eigen nyata bukan nol yang menunjukkan arah regangan akibat transformasi, sedangkan nilai eigen dianggap sebagai faktor yang menentukan seberapa besar regangan tersebut. Jika nilai eigen bernilai negatif, arah transformasi akan berbalik.

Vektor eigen adalah vektor (bukan nol) yang tidak berubah arah ketika diterapkan transformasi linier. Vektor tersebut berubah hanya dengan faktor skalar. Secara singkat dapat dikatakan, jika A adalah transformasi linier dari ruang vektor V dan x adalah vektor di V, yang bukan merupakan vektor nol, maka v adalah vektor eigen dari A jika A(X) adalah kelipatan skalar dari x.



Gambar 2.2. Hasil perkalian vektor eigen
Sumber: Wikipedia

Untuk menghitung nilai eigen dan vektor eigen, kita dapat memanfaatkan persamaan (1) untuk menghasilkan persamaan baru. Berikut penurunan rumusnya:

$$\begin{aligned} Ax &= \lambda x \\ IAx &= \lambda Ix \\ Ax &= \lambda Ix \\ (\lambda I - A)x &= 0 \end{aligned} \tag{2}$$

Persamaan tersebut memiliki solusi x = 0 yang merupakan solusi trivial. Untuk mencari solusi lain selain x = 0, maka harus terpenuhi persamaan di bawah ini:

$$\det(\lambda I - A) = 0 \tag{3}$$

Persamaan (3) disebut sebagai persamaan karakteristik. Persamaan tersebut memiliki akar-akar karakteristik berupa nilai eigen.

C. Eigenfaces dan PCA

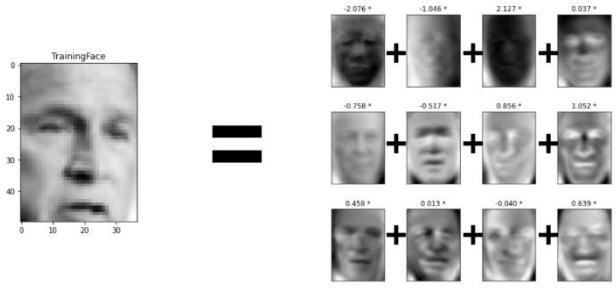
Metode Eigenfaces adalah metode yang memanfaatkan nilai eigen dan vektor eigen untuk pengenalan wajah. Metode ini mengandalkan PCA untuk reduksi dimensi. Reduksi dimensi menggunakan PCA bertujuan untuk merepresentasikan gambar wajah dalam dimensi yang lebih rendah (disebut ruang eigen), di mana setiap wajah direpresentasikan sebagai kombinasi dari vektor eigen yang menangkap variasi dalam pengumpulan data.

PCA (Principal Component Analysis) adalah teknik statistik yang digunakan untuk menyederhanakan data yang kompleks. Ide utamanya adalah mengubah data ke dalam sistem koordinat baru, di mana perbedaan terbesar (varians) dalam data ditempatkan pada beberapa sumbu pertama. Sumbu-sumbu ini disebut komponen utama. Komponen utama ini adalah arah-arah yang paling mencerminkan perbedaan dalam data.

Dalam konteks matematika, komponen utama ini berasal dari vektor eigen yang dihitung dari matriks kovarians data. Vektor-vektor eigen ini menunjukkan arah mana yang paling menggambarkan perbedaan dalam data, dan nilai eigen yang terkait dengan vektor-vektor tersebut menggambarkan seberapa besar perbedaan yang bisa ditangkap oleh setiap arah tersebut.

Singkatnya, PCA membantu kita menemukan pola yang paling penting dalam data dan mengurangi kompleksitas dengan hanya menggunakan sebagian kecil komponen utama yang menangkap perbedaan terbesar.

Saat diterapkan pada gambar wajah, PCA membantu mengubah sekumpulan besar gambar wajah menjadi sejumlah "eigenface" yang merupakan arah atau pola utama di mana wajah-wajah tersebut paling berbeda satu sama lain. Setiap gambar wajah kemudian bisa digambarkan sebagai gabungan dari eigenface ini dengan bobot tertentu. Dengan cara ini, PCA menyederhanakan gambar wajah, sehingga lebih mudah untuk mengenali wajah dalam data yang lebih kecil dan lebih sederhana.



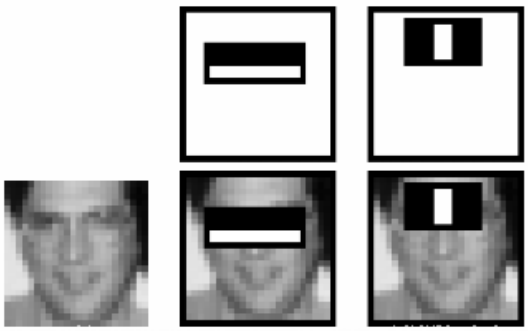
Gambar 2.3. Metode Eigenfaces
Sumber: Geeksforgeeks

Metode Eigenfaces melibatkan perhitungan matematika dalam berbagai prosesnya. Pada representasi wajah, setiap gambar wajah dibentuk ulang menjadi vektor, dan seluruh kumpulan data direpresentasikan sebagai matriks yang setiap barisnya berhubungan dengan gambar yang divektorkan. Lalu, pada metode Eigenfaces, rata-rata wajah (vektor rata-rata) dihitung dan dikurangkan dari setiap gambar untuk menghilangkan informasi umum di semua wajah, yang memungkinkan model untuk fokus pada fitur unik. Eigenfaces juga melibatkan matriks kovarians dalam prosesnya. Matriks kovarians dihitung dari gambar yang dikurangi rata-rata. Matriks ini menangkap bagaimana berbagai fitur (piksel) wajah berbeda-beda satu sama lain. Dengan menyelesaikan masalah nilai eigen untuk matriks kovarians, wajah eigen diperoleh sebagai vektor eigen dari matriks kovarians. Vektor eigen ini kemudian digunakan untuk memproyeksikan wajah ke ruang berdimensi lebih rendah (ruang eigen).

D. Haar Cascade

Haar Cascade merupakan salah satu metode deteksi objek yang diperkenalkan oleh Paul Viola dan Michael Jones pada tahun 2001 dalam paper mereka yang berjudul "*Rapid Object Detection using a Boosted Cascade of Simple Features*". Metode ini dikenal karena kecepatan dan akurasi yang cukup baik dalam mendeteksi objek, termasuk wajah.

Haar Cascade merupakan algoritma pendeteksi objek berbasis fitur untuk mendeteksi objek dari gambar, salah satu contoh objek yang dapat dideteksi adalah wajah. Fungsi *cascade* memerlukan pelatihan pada banyak gambar positif dan negatif untuk dideteksi.



Gambar 2.4. Algoritma Haar Cascade
Sumber: OpenCV

Keunggulan dari algoritma Haar Cascade ini adalah algoritmanya tidak memerlukan komputasi yang ekstensif dan dapat berjalan secara real-time. Kita juga dapat melatih fungsi *cascade* kita sendiri untuk objek-objek khusus, seperti hewan, mobil, sepeda, dll.

Haar Cascade bisa mendeteksi wajah, tetapi tidak dapat melakukan pengenalan wajah karena hanya mengidentifikasi bentuk dan ukuran yang cocok. Maka dari itu, jika ingin melakukan pengenalan wajah dengan algoritma Haar Cascade, dibutuhkan alat lain untuk bisa mewujudkannya.

E. SVD (Singular Value Decomposition)

SVD merupakan salah satu metode dekomposisi matriks yang banyak diaplikasikan dalam berbagai bidang, seperti pemrosesan citra, analisis data, dan pengenalan pola. SVD memecah sebuah matriks A berukuran mxn menjadi perkalian dari tiga buah matriks:

$$A = U\Sigma V^T \quad (4)$$

U = matriks ortogonal yang kolom-kolomnya adalah vektor eigen dari AA^T

Σ = matriks diagonal yang memiliki nilai singular dari matriks A^TA

V^T = matriks ortogonal yang baris-barisnya adalah vektor eigen dari A^TA

Dapat dilihat bahwa SVD memiliki keterkaitan erat dengan konsep vektor eigen dan nilai eigen dalam analisis matriks.

III. IMPLEMENTASI

Implementasi pengenalan wajah ini memanfaatkan Eigenfaces, PCA, dan Haar Cascade. Algoritma pengenalan wajah ini menggunakan kombinasi pendekatan berbasis PCA (Principal Component Analysis), representasi wajah dalam bentuk Eigenfaces, serta algoritma deteksi wajah Haar Cascade. Kombinasi ini memiliki beberapa keunggulan, di antaranya adalah Haar Cascade yang memudahkan deteksi wajah secara real-time, serta PCA dan Eigenfaces yang memberikan cara yang efisien untuk melakukan pengenalan wajah dengan mengurangi dimensi data tanpa menghilangkan informasi yang penting sehingga kombinasi ini mengambil keunggulan dari masing-masing metode.

A. Program Utama

```python
import cv2
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from math import ceil, sqrt

def showImgs(imgs, n_imgs, i_imgs):
    n = sqrt(n_imgs)
    m = ceil(n)
    n = int(n)
    fig = plt.figure(figsize=(12, 8))
    p = 1
    for i in i_imgs:
        ax = fig.add_subplot(n, m, p)
        ax.imshow(imgs[i], cmap='gray')
        ax.axis('off')
        p += 1
    plt.tight_layout()
    plt.show()

from sklearn.datasets import fetch_olivetti_faces
faces = fetch_olivetti_faces()
images = faces.images
features = faces.data
targets = faces.target

Γ = np.array([I.reshape(-1) for I in images])
Ψ = np.mean(Γ, axis=0)
Φ = Γ - Ψ

pca = PCA(svd_solver='full')
pca.fit(features)
best_eigenfaces = pca.components_[:40]
weights = pca.transform(Φ)
```

Gambar 3.1. Kode program utama bagian pertama

```python

def recognize_face(face_resized, pca, weights, targets, Ψ):
    ϕ_webcam = face_resized - Ψ
    weight_webcam = pca.transform([ϕ_webcam])
    distances = np.linalg.norm(weights - weight_webcam, axis=1)
    matched_index = np.argmin(distances)
    return targets[matched_index], distances[matched_index], matched_index

cap = cv2.VideoCapture(1)
if not cap.isOpened():
    print("OBS Virtual Camera tidak ditemukan.")
    exit()
print("Tekan 'q' untuk keluar.")
while True:
    ret, frame = cap.read()
    if not ret:
        print("Tidak dapat membaca frame dari OBS Virtual Camera.")
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces_detected = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    faces = faces_detected.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    for (x, y, w, h) in faces:
        face_roi = gray[y:y+h, x:x+w]
        face_resized = cv2.resize(face_roi, (64, 64)).flatten()

        label, distance, matched_index = recognize_face(face_resized, pca, weights, targets, Ψ)

        text = f"ID: {label}, Dist: {distance:.2f}"
        cv2.putText(frame, text, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        cv2.imshow(f"ID {label}", images[matched_index])
    cv2.imshow("OBS Virtual Camera - Face Recognition", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Gambar 3.2. Kode program utama bagian kedua

Program utama ini mengandung kode yang mencakup PCA, representasi wajah dalam bentuk Eigenfaces, serta algoritma deteksi wajah menggunakan Haar Cascade. Program ini memungkinkan kita untuk melakukan pengenalan wajah menggunakan *virtual camera* (pada implementasi ini menggunakan OBS Studio). Lalu, menampilkan gambar dari dataset Olivetti yang memiliki kemiripan terbesar dengan wajah yang terdeteksi dari *virtual camera*.

B. Pengambilan Dataset Gambar Wajah Olivetti

```python
from sklearn.datasets import fetch_olivetti_faces
faces = fetch_olivetti_faces()
images = faces.images
features = faces.data
targets = faces.target
```

Gambar 3.3. Kode untuk mengambil gambar wajah

Pada potongan kode tersebut, kita mengimpor dataset Olivetti dari sklearn.datasets. Lalu, dataset tersebut dimuat dan disimpan di dalam variabel faces. Dataset tersebut memiliki informasi images, yaitu gambar wajah dalam format array 2D, data, yaitu gambar yang sudah diratakan menjadi array 1D, target, yaitu label numerik untuk setiap wajah (serupa dengan ID tiap orang), dan DESCR, yaitu deskripsi dari dataset. Kemudian, kita mengambil bagian images dari dataset yang sudah disimpan pada variabel faces dan disimpan di dalam iamges. Begitu pula untuk mengambil bagian data dari dataset, kita menyimpannya di variabel features, dan bagian target dari dataset disimpan di dalam targets.

C. *Mean Face* dan PCA

```python
Γ = np.array([I.reshape(-1) for I in images])
Ψ = np.mean(Γ, axis=0)
Φ = Γ - Ψ
```

Gambar 3.4. Kode untuk menghitung *mean face*

Pada potongan kode tersebut, pertama, gambar wajah I dalam array images yang awalnya adalah array 3D diratakan menjadi 1D menggunakan fungsi reshape(-1).

Proses tersebut membuat images menjadi array 2D, di mana setiap baris merupakan representasi vektor 1D dari gambar wajah. Setelah itu, rata-rata dari array 2D tersebut dihitung menggunakan fungsi np.mean() yang menghasilkan vektor 1D berisi nilai rata-rata dari setiap piksel pada seluruh gambar. Maka, variabel Ψ pada kode disebut sebagai *mean face*. Proses mencari *mean face* ini dilakukan untuk menghilangkan bias dari data. Setelah mendapat *mean face*, dilakukan substraksi wajah dari setiap gambar pada Γ (vektor wajah) dengan Ψ (*mean face*) untuk menghasilkan gambar wajah yang sudah terpusat di sekitar nol. Proses ini mempersiapkan data untuk analisis lebih lanjut menggunakan PCA (Principal Component Analysis), dengan memastikan bahwa setiap wajah yang dianalisis sudah terpusat pada rata-rata dataset.

```
pca = PCA(svd_solver='full')
pca.fit(features)
best_eigenfaces = pca.components_[:40]
weights = pca.transform(Φ)
```

Gambar 3.5. Kode untuk melakukan PCA

Kode di atas merupakan implementasi dari PCA (Principal Component Analysis) yang digunakan untuk mereduksi dimensi data dan mengekstrak fitur utama, yaitu komponen utama yang disebut eigeinfaces, dari data. PCA diimpor dari sklearn.decomposition dan diberi paramenter svd_solver = ‘full’. Parameter tersebut menunjukkan bahwa pendekatan yang digunakan untuk menghitung komponen utama adalah pendekatan SVD penuh. Lalu, PCA dilatih menggunakan dataset wajah yang sudah disimpan pada variabel features. Dalam proses tersebut, PCA menghitung komponen utama dari features. Hal tersebut memungkinkan pereduksian dimensi dataset dan penemuan fitur-fitur yang paling signifikan. Kemudian, 40 komponen utama pertama diambil dari hasil PCA yang telah disimpan dalam atribut components_. Hasilnya disimpan dalam array best_eigenfaces.

Dari proses sebelumnya, diketahui bahwa Φ merupakan hasil substraksi. Hasil substraksi itu diproyeksikan ke dalam ruang komponen utama (eigenspace) yang telah dihitung dengan menggunakan PCA. Proses tersebut menghasilkan array weights yang berisi hasil proyeksi setiap wajah ke dalam 40 komponen utama yang dipilih sebelumnya.

D. Pencocokan Wajah Baru dengan Dataset

```
def recognize_face(face_resized, pca, weights, targets, Ψ):
    ϕ_webcam = face_resized - Ψ
    weight_webcam = pca.transform([ϕ_webcam])
    distances = np.linalg.norm(weights - weight_webcam, axis=1)
    matched_index = np.argmin(distances)
    return targets[matched_index], distances[matched_index], matched_index
```

Gambar 3.6. Fungsi pencocokan wajah

Kode di atas merupakan kode dari fungsi recognize_face() yang berfungsi untuk mencocokkan wajah yang dideteksi dari kamera dengan dataset wajah Olivetti. Fungsi tersebut memiliki beberapa parameter, di antaranya adalah face_resized (berisi gambar wajah yang telah diproses menjadi vektor 1D), pca (objek PCA yang telah dilatih menggunakan dataset wajah), weights (bobot yang dihasilkan dari transformasi PCA pada dataset wajah yang merepresentasikan posisi wajah pada ruang eigen), targets (label wajah dari setiap wajah pada dataset), dan Ψ (*mean face*).

Sama halnya seperti proses sebelumnya, wajah yang terdeteksi dari kamera disubstraksi dengan *mean face*. Lalu, wajah dari kamera diproyeksikan ke ruang eigen yang dibentuk oleh eigenfaces. Hasilnya disimpan dalam weight_webcam, yang berisi bobot wajah tersebut di ruang eigen yang dibentuk oleh eigenfaces. Kemudian, untuk mencari gambar dengan kemiripan terbesar, kita perlu menghitung jarak Euclidean antara gambar dari kamera dengan wajah yang ada dalam dataset (yang disimpan dalam weights). Terakhir, fungsi akan mengembalikan label wajah yang dikenali (berupa semacam ID), jarak Euclidean antara wajah yang dikenali dengan wajah yang terdeteksi, dan indeks wajah yang paling mirip dalam dataset.

E. Pendeteksian dan Pengenalan Wajah

```
cap = cv2.VideoCapture(1)
if not cap.isOpened():
    print("OBS Virtual Camera tidak ditemukan.")
    exit()
print("Tekan 'q' untuk keluar.")
while True:
    ret, frame = cap.read()
    if not ret:
        print("Tidak dapat membaca frame dari OBS Virtual Camera.")
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces_detected = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    faces = faces_detected.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    for (x, y, w, h) in faces:
        face_roi = gray[y:y+h, x:x+w]
        face_resized = cv2.resize(face_roi, (64, 64)).flatten()

        label, distance, matched_index = recognize_face(face_resized, pca, weights, targets, Ψ)

        text = f"ID: {label}, Dist: {distance:.2f}"
        cv2.putText(frame, text, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        cv2.imshow(f"ID {label}", images[matched_index])
    cv2.imshow("OBS Virtual Camera - Face Recognition", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Gambar 3.7. Deteksi dan pengenalan wajah dengan OBS

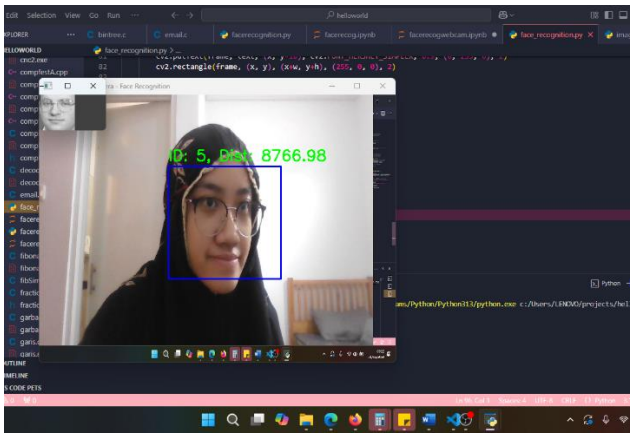
Program tersebut berisi proses pendeteksian wajah yang baru dan pengenalan wajah pada dataset yang paling mirip dengan wajah baru tersebut. Pertama-tama, program membuka kamera (dalam implementasi ini menggunakan *virtual camera* pada OBS Studio). Dengan bantuan OpenCV, program menangkap video dari *virtual camera*. Setelah melewati semua proses validasi dan kamera berhasil di buka, akan dijalankan loop selama frame kamera masih berhasil dibaca. Lalu, gambar yang diterima dari kamera/*frame* dikonversi menjadi *grayscale*. Hal ini dilakukan karena akan mempercepat proses deteksi dan meningkatkan keakuratan.

Program menggunakan algoritma Haar Cascade untuk mendeteksi wajah. Prosesnya dimulai dengan memuat *classifier* Haar Cascade yang sudah dilatih untuk mendeteksi wajah. Kemudian, dengan fungsi detectMultiScale(), program akan mencari wajah dalam gambar dengan ukuran dan proporsi yang berbeda. Parameter dari fungsi tersebut terdiri dari scaleFactor (faktor skala untuk meningkatkan ketelitian deteksi), minNeighbors (memberikan batas tetangga yang harus dimiliki agar dianggap sebagai wajah yang valid), dan minSize (ukuran minimum objek yang harus dideteksi

sebagai wajah).

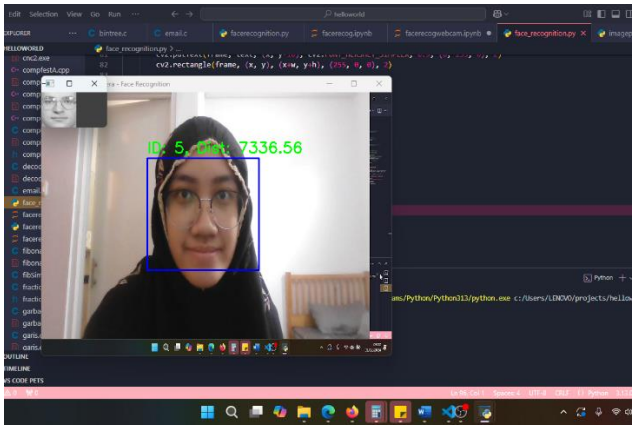
Proses dilanjut dengan dilakukannya iterasi untuk setiap wajah yang terdeteksi dalam gambar. Wajah yang terdeteksi dipotong dari gambar, lalu diubah ukurannya menjadi 64x64 piksel dan diubah menjadi vektor 1D. Kemudian, wajah terdeteksi yang sudah dipotong dimasukkan sebagai parameter dalam pemanggilan fungsi recognize_faces(). Fungsi tersebut akan mengenali wajah, lalu mengembalikan label wajah, jarak, dan indeksnya. Hasil dari pemanggilan recognize_faces() langsung ditampilkan di kamera dengan disertai gambar wajah yang paling mirip dengan wajah yang dideteksi kamera.

F. Hasil Pengujian



Gambar 3.8. Hasil pengujian pertama

Dilakukan pengujian pertama dengan sisi wajah yang sedikit miring. Program mengeluarkan hasil bahwa gambar dengan ID 5 memiliki jarak terdekat yang berarti memiliki kemiripan terbesar dengan wajah yang dideteksi. Pada kamera, diberikan kotak pada wajah yang terdeteksi dan ditampilkan ID beserta jaraknya. Dapat dilihat bahwa jarak gambar orang dengan ID 5 dengan wajah yang terdeteksi adalah sebesar 8766.98.



Gambar 3.9. Hasil pengujian kedua

Pada pengujian kedua, wajah sudah diluruskan dan program mendeteksi hal tersebut sehingga menampilkan gambar yang lebih mirip lagi (sama-sama menghadap ke depan). Sekarang, jarak Euclidean mengecil menjadi 7336.56. Sedangkan, wajah yang dideteksi masih memiliki kemiripan terdekat dengan ID wajah orang yang sama, yaitu ID 5.

IV. KESIMPULAN

Pengenalan wajah adalah salah satu bentuk inovasi terbaru yang dapat memperluas fungsi CCTV dari yang awalnya hanya sekadar pengawasan visual menjadi alat identifikasi biometrik yang canggih. Teknologi modern ini memungkinkan sistem untuk memindai, menganalisis, dan mencocokkan wajah individu dengan data dalam basis data. Hal tersebut bisa menjadi efisien dalam upaya peningkatan keamanan. Dalam implementasi pengenalan wajah, salah satu cara yang dapat digunakan adalah menggunakan nilai eigen dan vektor eigen (eigenfaces). Tentunya, cara ini masih cukup sederhana sehingga perlu digabungkan dengan PCA dan algoritma Haar Cascade.

Pendekatan ini menggunakan konsep dekomposisi nilai eigen dan vektor eigen untuk mereduksi dimensi data gambar wajah yang awalnya berukuran tinggi menjadi representasi fitur utama yang lebih sederhana. Fitur-fitur ini disebut eigenfaces, yang menggambarkan karakteristik wajah secara matematis dan digunakan untuk membangun ruang eigenspace. Proses pengenalan wajah melibatkan proyeksi wajah input ke ruang eigen untuk mendapatkan bobot fitur, yang kemudian dibandingkan dengan bobot fitur wajah dalam dataset menggunakan jarak Euclidean untuk mengenali individu. Pendekatan ini juga memanfaatkan Haar Cascade Classifier untuk mendeteksi lokasi wajah pada gambar yang ditangkap oleh kamera sebelum melanjutkan ke proses pengenalan wajah. Input wajah berasal dari kamera dan dibandingkan kemiripannya dengan dataset wajah Olivetti. Program akan menampilkan gambar yang memiliki jarak Euclidean terdekat dengan wajah yang dideteksi oleh kamera.

V. UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan makalah yang berjudul “Pengenalan Wajah Berbasis Eigenvalue dan Eigenvector pada Sistem Keamanan CCTV” dengan baik. Penulis juga mengucapkan terima kasih kepada pihak-pihak yang telah mendukung dalam penulisan makalah ini, yaitu:

1. Bapak Dr. Rila Mandala, Bapak Dr. Rinaldi Munir, Bapak Dr. Judhi Santoso, serta Bapak Arrival Dwi Sentosa, M.T selaku dosen-dosen pengajar IF2123 Aljabar Linier dan Geometri atas ilmu, pengajaran, dan bimbingan yang telah diberikan kepada penulis selama perkuliahan,
2. Orang tua penulis yang telah memberikan dukungan dan semangat hingga saat ini,
3. Teman-teman penulis yang telah berjuang bersama-sama dan memberikan dukungan dalam menyelesaikan makalah ini,

Akhir kata, penulis berharap makalah ini akan memberikan manfaat dan wawasan kepada pembacanya.

REFERENSI

[1] Analytics Vidhya, "Face Detection Using Haar Cascade Using Python." https://www.analyticsvidhya.com/blog/2022/10/face-detection-using-haar-cascade-using-python/. (diakses 28 Desember 2024)

[2] R. Munir, "Nilai Eigen dan Vektor Eigen (Bagian 1)," https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf. (diakses 29 Desember 2024).

[3] R. Munir, "Singular Value Decomposition (Bagian 1)." https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-21-Singular-value-decomposition-Bagian1-2023.pdf. (diakses 29 Desember 2024)

[4] 3Sixty Integrated, "The History of Video Surveillance." https://www.3sixtyintegrated.com/blog/2023/07/26/history-video-surveillance/. (diakses 29 Desember 2024)

[5] TechCube, "Facial Recognition CCTV Systems Explained," https://www.techcube.co.uk/blog/facial-recognition-cctv-systems-explained/. (diakses 29 Desember 2024).

[6] H. Id Mansour, "Face Recognition Using Eigenfaces (Python)," https://hassan-id-mansour.medium.com/face-recognition-using-eigenfaces-python-b857b2599ed0. (diakses 30 Desember 2024).

[7] M. Turk and A. Pentland, "Eigenfaces for recognition," J. Cognitive Neurosci., vol. 3, no. 1, pp. 71–86, 1991.

[8] C. M. Bishop, Pattern Recognition and Machine Learning. New York: Springer, 2006, ch. 5.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

Ranashahira Reztaputri
13523007