PENGENALAN WAJAH MENGGUNAKAN *PRINCIPAL COMPONENT ANALYSIS* (PCA)

SKRIPSI

oleh :

NITA DEWI EKOWATI

0610960050-96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2011

UNIVERSITAS BRAWIJAYA

PENGENALAN WAJAH MENGGUNAKAN *PRINCIPAL COMPONENT ANALYSIS* (PCA)

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar sarjana dalam bidang ilmu komputer

oleh :

NITA DEWI EKOWATI

0610960050-96

PROGRAM STUDI ILMU KOMPUTER

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS BRAWIJAYA

MALANG

2011


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN

PENGENALAN WAJAH MENGGUNAKAN *PRINCIPAL COMPONENT ANALYSIS* (PCA)

oleh :
NITA DEWI EKOWATI
0610960050-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 29 Desember 2010
Dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang ilmu komputer

Pembimbing I	Pembimbing II
Drs. Muh. Arif Rahman, M. Kom	**Drs. Marji, MT**
NIP. 196604231991111001	**NIP. 196708011992031001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya,

Dr. Abdul Rouf Algofari, M.Sc.
NIP. 196709071992203001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Nita Dewi Ekowati
NIM : 0610960050
Jurusan : Matematika / Ilmu Komputer
Penulis Skripsi berjudul : Pengenalan Wajah Menggunakan *Principal Component Analysis* (PCA)

Dengan ini menyatakan bahwa:

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila di kemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 29 Desember 2011
Yang menyatakan,

(Nita Dewi Ekowati)
NIM. 0610960050

PENGENALAN WAJAH MENGGUNAKAN *PRINCIPAL COMPONENT ANALYSIS* (PCA)

ABSTRAK

Principal Component Analysis (PCA) merupakan sebuah metode statistika yang sering digunakan dalam pengenalan wajah. Dengan PCA, gambar-gambar wajah direduksi dimensinya kemudian dicari hubungan antar gambar tersebut agar dapat diklasifikasikan dalam dimensi yang lebih kecil. Di Indonesia, telah banyak penelitian mengenai pengenalan wajah. Sebagian penelitian-penelitian yang telah ada tersebut belum menjelaskan penerapan algoritma PCA secara bertahap. Oleh karena itu dalam tulisan ini dituliskan cara penerapan algoritma PCA untuk pengenalan wajah secara bertahap dan rinci. Pada penelitian ini pengujian dilakukan dengan mengubah banyaknya data latihan serta banyaknya wajah eigen yang digunakan untuk mengetahui pengaruhnya terhadap kemampuan sistem dalam mengenali wajah. Kemampuan sistem dalam mengenali wajah diukur dengan menggunakan *precision* dan *recall*. Dari hasil pengujian di mana wajah input sama dengan wajah latihan, nilai *precision* dan *recall* dapat mencapai 1. Hal ini menunjukkan bahwa PCA dapat digunakan untuk mengenali kembali wajah yang dilatihkan dengan baik. Sedangkan hasil pengujian di mana wajah yang tidak dilatihkan diinputkan dalam jumlah yang sama untuk tiap-tiap pelatihan dengan jumlah data latihan berbeda, menghasilkan nilai *precision* dan *recall* yang berkisar antara 0.6 hingga 0.9 untuk data latihan berjumlah 30 hingga 50. Hal tersebut menunjukkan bahwa PCA mampu mengenali wajah yang tidak dilatihkan dengan cukup baik.

Kata kunci*: PCA, pengenalan wajah, precision dan recall, tahapan penerapan PCA, wajah eigen.*

FACE RECOGNITION USING PRINCIPAL COMPONENT ANALYSIS (PCA)

ABSTRACT

Principal Component Analysis (PCA) is a statistical method often used in face recognition. With PCA, facial images dimensions are reduced so that the relationship of those facial images can be classified in smaller dimensions. There were a lot of research about face recognition using PCA in Indonesia. Most of those research did not explain the implementation of PCA algorithm step-by-step. Therefore in this research was written the way of implementing PCA algorithm for face recognition in detail. In this research testing was carried out by changing the amount of training data and the amount of eigenfaces used in recognition process, to determine the effect of those changing to the system's ability to recognizing faces. The system's ability to recognize faces is measured using precision and recall. From the test results where the input face same as the training face, the value of precision and recall can reach 1. This suggest that the PCA can be used to recognize training face well. While the results of testing using untrained face, same amount entered with different amount of training data, resulting precision and recall values ranging from 0.6 to 0.9, to training data amounted to 30 to 50. It shows that PCA is able to recognize untrained faces well enough.

Keywords*: PCA, face recognition, precision and recall, step-by-step PCA implementation, eigenface.*

KATA PENGANTAR

Alhamdulillahi rabbil ‘alamin. Puji syukur Penulis haturkan kehadirat Allah SWT yang telah memberikan rahmatNya sehingga Penulis dapat menyelesaikan penulisan skripsi yang berjudul “*Pengenalan Wajah Menggunakan* Principal Component Analysis *(PCA)* ”. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Skripsi ini bertujuan untuk menerapkan teori Principal Component Analysis dalam memodelkan dan mengenali wajah.

Dalam penyusunan dan penyelesaian skripsi ini, Penulis mengucapkan terima kasih kepada :

1. Drs. Muh. Arif Rahman, M.Kom selaku pembimbing utama dalam penulisan skripsi ini
2. Drs. Mardji, MT selaku pembimbing pendamping dan Ketua Program Studi Ilmu Komputer, Jurusan Matematika, FMIPA Universitas Brawijaya.
3. Dewi Yanti Liliana, S.Kom., M.Kom selaku dosen pembimbing akademis
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Studi Ilmu Komputer
5. Segenap staf dan karyawan Jurusan Matematika Universitas Brawijaya yang telah membantu penyusunan skripsi ini.
6. Kepada Orang Tua dan juga adikku yang tak pernah berhenti memberikan doa dan dukungannya kepada Penulis.
7. Sahabatku Lutfi, Tyas, Dinda, Septi dan Asri yang telah meluangkan waktunya untuk membantu proses belajar dan penyusunan skripsi.
8. Rekan-rekan Ilkomers 2006 yang telah memberikan dukungan dan semangatnya.
9. Rekan-rekan PAKO (Kak Fanny, Kak Putri, Tante Lily, Kak Jessi, Kak Erly, dan Kak Aning) yang selalu memberi semangat dan motivasi.
10. Sahabatku Yusixka yang tak pernah berhenti mendoakan dan mengingatkan untuk selalu berusaha.

11. Dan semua pihak yang telah membantu dalam penyusunan skripsi ini yang tidak dapat disebutkan satu per satu.

Penulis sadari bahwa masih banyak kekurangan dalam penulisan laporan ini yang disebabkan oleh keterbatasan kemampuan dan pengalaman. Semoga penulisan skripsi ini bermanfaat bagi pembaca

Malang, Desember 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN


UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Pengenalan wajah merupakan sebuah kegiatan untuk mengetahui identitas seseorang berdasarkan ciri-cirinya. Pengenalan wajah berkembang seiring dengan perkembangan sistem komputer yang semakin memudahkan pemrosesan citra digital. Pengenalan wajah sebagai salah satu teknologi biometrik semakin penting digunakan dalam pengembangan teknologi dalam aplikasi praktis seperti kamera digital, dan aplikasi pengamanan.

Penelitian terhadap pengenalan wajah terus dilakukan karena semakin banyaknya aplikasi praktis yang memerlukan identifikasi manusia. Telah banyak ilmuwan yang mengemukakan pendekatan-pendekatan dalam pengenalan wajah. Pendekatan awal pengenalan wajah dilakukan dengan mendeteksi ciri-ciri individu seperti bentuk luar kepala, letak mata, hidung dan mulut. Ciri-ciri ini kemudian digunakan untuk mendefinisikan sebuah model wajah berdasarkan posisi, ukuran, serta hubungan antara ciri-ciri tersebut. Pendekatan dengan menggunakan ciri-ciri individu dan hubungannya menurut Carey dan Diamond, dalam tulisannya *From Piecemeal to Configurational Representation of Faces* (1977), tidak cukup menyediakan sebuah representasi untuk menjelaskan kinerja identifikasi wajah manusia dewasa (Turk, 1991). Hal tersebut disebabkan karena pendekatan dengan menggunakan ciri-ciri individu kurang baik dalam menghadapi perbedaan posisi wajah dan pencahayaan.

Penelitian-penelitian selanjutnya diupayakan untuk memperoleh sebuah sistem pengenalan wajah yang lebih cepat, dan semakin tidak tergantung pada ciri-ciri individu. Bledsoe (1966), mengajukan pengenalan wajah semiotomatis dengan menggunakan sistem *hybrid human-computer*, yang mengklasifikasikan wajah berdasarkan tanda yang diberikan pada foto. Parameter yang digunakan untuk klasifikasi adalah *normalized distance*, dan rasio antar titik seperti sudut mata, sudut bibir, ujung hidung, dan titik dagu. Pendekatan lain diajukan oleh Fischler dan Elschlager (1973) dengan menghitung ciri-ciri yang mirip secara otomatis dengan menggunakan *template matching*.

Pada tahun 1991, Turk dan Pentland dalam jurnalnya *Eigenface for Recognition*, mengajukan pendekatan pengenalan wajah berdasarkan *Principal Component Analysis* (PCA). Pendekatan tersebut diupayakan

untuk membuat sebuah sistem pengenalan wajah yang tidak bergantung pada posisi ciri-ciri fisik wajah. Ide dasar dari sistem pengenalan ini adalah dengan mendekomposisi gambar wajah ke dalam serangkaian gambar ciri karakteristik, yang disebut *eigenface*. *Eigenface* dapat dianggap sebagai *principal component* dari sekumpulan gambar pelatihan awal wajah. Proses pengenalan dilakukan dengan memproyeksikan sebuah gambar baru ke dalam sebuah ruang bagian yang terdiri dari sekumpulan *eigenface*, dan kemudian mengelompokkan wajah dengan membandingkan posisinya dalam ruang wajah (dilakukan dengan mentransformasikan gambar pada komponen *eigenface*) dengan posisi individu yang dikenali.

PCA merupakan sebuah cara untuk menemukan pola dalam sebuah data dan menampilkan data sedemikian rupa untuk menampilkan perbedaan dan persamaannya. Dengan PCA, gambar-gambar wajah direduksi dimensinya, kemudian dicari hubungan antar gambar tersebut agar dapat diklasifikasikan.

Dengan menggunakan ide dasar dari PCA tersebut, pendekatan pengenalan wajah yang diajukan oleh Turk dan Pentland ini tidak bergantung pada posisi ciri-ciri wajah secara fisik, sehingga membuatnya lebih baik dalam menghadapi kondisi wajah dengan pencahayaan atau posisi yang berbeda. Pengenalan wajah dilakukan dengan membandingkan persamaan dan perbedaan pola yang terdapat dalam gambar.

Di Indonesia sendiri, telah banyak penelitian mengenai pengenalan wajah menggunakan *Principal Component Analysis*. Di antaranya adalah "Perancangan Program Aplikasi Pengenalan Wajah berbasiskan JST dengan Menerapkan Metode PCA" (Wikaria, 2003); "Ekstraksi Fitur Berbasis 2D-*Discrete Consine Transform* dan *Principal Component Analysis* untuk Pengenalan Citra Wajah" (Muntasa, 2009); dll. Sebagian besar dari penelitan-penelitian yang telah ada tersebut belum menjelaskan penerapan algoritma PCA secara bertahap. Oleh karena itu penulis berusaha untuk menuliskan cara penerapan algoritma PCA untuk pengenalan wajah secara bertahap dan rinci. Berdasarkan latar belakang yang telah diuraikan, maka penulis mengambil judul "**Pengenalan Wajah dengan Menggunakan *Principal Component Analysis***"

1.2 Rumusan Masalah

Dalam penelitian ini rumusan masalah adalah sebagai berikut:

1. Bagaimana algoritma PCA diterapkan untuk memodelkan sekumpulan gambar wajah.
2. Bagaimana model wajah yang didapatkan melalui PCA digunakan untuk mengenali wajah.
3. Bagaimana kemampuan sistem pengenalan wajah menggunakan PCA dalam mengenali wajah.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Data latih yang digunakan harus memiliki ukuran dan tipe yang sama.
2. Baik data latih maupun data gambar yang digunakan sudah merupakan gambar wajah.
3. Data uji yang digunakan terdiri dari dua macam data wajah, yaitu data wajah yang digunakan dalam pelatihan, serta data wajah baru.

1.4 Tujuan Penelitian

1. Mengimplementasikan algoritma PCA untuk mengenali wajah.
2. Mengetahui hubungan antara jumlah data latihan dengan kemampuan sistem dalam mengenali wajah.

1.5 Manfaat Penelitian

Hasil penelitian pada skripsi ini dapat digunakan untuk mengenali sebuah gambar wajah, apakah wajah tersebut dikenali atau tidak, sehingga dapat digunakan untuk mengembangkan sebuah sistem yang memerlukan pengenalan wajah sebagai identifikasi.

1.6 Metode Penelitian

Metode penelitian yang dilakukan pada penelitian ini adalah :

1. Studi Literatur

Mempelajari dan mengkaji beberapa literatur (jurnal, buku, dan artikel dari *website*) mengenai pengenalan wajah, algoritma *principal component analysis* (PCA), serta algoritma pengenalan wajah menggunakan PCA.

2. Perancangan dan implementasi sistem

Mengimplementasikan algoritma PCA untuk melakukan pengenalan wajah pada sebuah gambar wajah masukan.

Pengimplementasian dilakukan dengan merancang dan membangun sebuah perangkat lunak untuk melakukan pelatihan pada gambar wajah yang ingin dikenali sistem, kemudian menggunakan data latihan tersebut untuk mengenali wajah baru.

3. Uji coba dan analisis hasil implementasi

Mengenali sebuah wajah yang termasuk dalam data latihan, untuk mengetahui keberhasilan penerapan algoritma, serta mengenali sebuah wajah baru untuk mengetahui keakuratan sistem.

1.7 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut:

1. **BAB I PENDAHULUAN**
   Berisi latar belakang penelitian, perumusan masalah, batasan masalah, tujuan penelitian, manfaat, metode penelitian, dan sistematika penulisan.
2. **BAB II TINJAUAN PUSTAKA**
   Bab ini berisi teori-teori dari berbagai pustaka yang menunjang penelitian dalam penulisan skripsi. Adapun teori yang tercakup dalam bab ini yaitu mengenai pengenalan wajah, *principal component analysis* (PCA), serta pengenalan wajah menggunakan PCA.
3. **BAB III METODOLOGI DAN PERANCANGAN SISTEM**
   Bab ini berisi mengenai perancangan perangkat lunak yang dibangun, meliputi analisis sistem, perancangan sistem, perancangan tabel dan perancangan uji coba.
4. **BAB IV IMPLEMENTASI DAN PEMBAHASAN**
   Bab ini berisi hasil dari implementasi perangkat lunak yang digunakan untuk melakukan pelatihan pada gambar wajah yang dimasukkan, pengenalan wajah baru berdasarkan data training, serta pembahasan analisis hasil uji coba dan evaluasi hasil uji coba.
5. **BAB V KESIMPULAN DAN SARAN**
   Bab ini memuat kesimpulan dari hasil penelitian dan saran-saran untuk pengembangan penelitian selanjutnya.

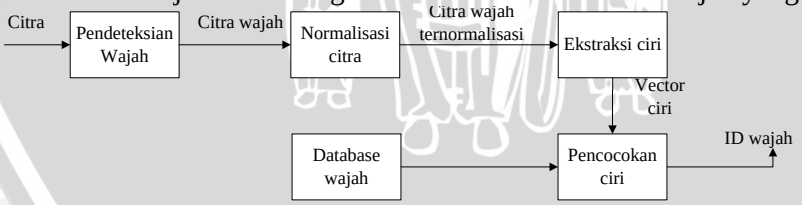
BAB II
TINJAUAN PUSTAKA

Bab ini berisi teori-teori dari berbagai pustaka yang menunjang penelitian dalam penulisan skripsi. Adapun teori yang tercakup dalam bab ini yaitu mengenai pengenalan wajah, *Principal Component Analysis* (PCA), serta penerapan PCA dalam pengenalan wajah.

2.1 Pengenalan Wajah

Pengenalan wajah merupakan salah satu masalah pengenalan pola. Wajah sebagai objek tiga dimensi yang memiliki keragaman pencahayaan, pose, dan ekspresi, diidentifikasi sebagai gambar dua dimensi. Sebuah sistem pengenalan wajah umumnya terdiri dari empat modul, yaitu: pendeteksian, pra pengolahan citra (*pre-processing image*) dengan melakukan normalisasi citra wajah, ekstraksi ciri wajah, dan pengenalan / pencocokan wajah (Li, 2005).

Deteksi wajah pada citra bertujuan untuk memisahkan wajah dari latar belakang. Sedangkan normalisasi dilakukan untuk mencari lokasi wajah dengan lebih akurat, serta bertujuan untuk mendapatkan komponen wajah secara garis besar, meliputi ukuran wajah dan komponen-komponen wajah. Setelah citra wajah dinormalisasi, proses selanjutnya adalah ekstraksi ciri-ciri wajah pada citra wajah tersebut. Ekstraksi ciri dilakukan untuk mendapatkan informasi-informasi penting yang dapat digunakan untuk membedakan wajah masing-masing individu. Langkah terakhir yaitu pencocokan citra wajah dilakukan dengan cara mencocokkan hasil ekstraksi ciri wajah baru terhadap data wajah. Bila ditemukan adanya kesesuaian citra wajah dengan data wajah, maka pencocokan wajah akan menghasilkan keluaran identitas wajah yang tepat.



Gambar 2.1 Aliran Proses Pengenalan Wajah

Gambar 2.1 menggambarkan aliran proses pengenalan wajah secara umum. Pada masukan yang berupa citra, dilakukan proses

pendeteksian untuk memisahkan wajah dengan latar belakang. Kemudian citra wajah yang telah didapatkan dari proses pendeteksian wajah dinormalisasi. Citra wajah yang telah ternormalisasi kemudian diekstrak ciri-cirinya. Proses ekstraksi ciri menghasilkan sebuah vektor ciri. Vektor ciri akan dicocokkan dengan data yang ada dalam database, jika ada yang sesuai, maka akan didapatkan keluaran berupa ID wajah.

Dalam suatu citra wajah masukan, wajah yang akan dikenali biasanya merupakan bagian dari citra masukan tersebut. Dapat dikatakan bahwa wajah terletak pada subruang citra masukan. Hal ini menunjukkan bahwa citra masukan memiliki informasi yang tumpang tindih serta dimensi yang cukup besar. Untuk membuat representasi citra masukan tersebut memiliki dimensi yang lebih kecil, pola wajah pada citra masukan perlu ditemukan terlebih dahulu.

Penentuan pola wajah dapat dilakukan dengan pendekatan *eigenface* atau yang disebut juga *Principal Component Analysis* (PCA). Dengan menggunakan PCA, sekumpulan kecil *eigenface* diturunkan dari *training set* citra wajah dengan menggunakan *Karhunen-Loeve transform* atau disebut juga PCA. Sebuah citra wajah direpresentasikan sebagai vektor ciri (vektor bobot dengan dimensi rendah). Ciri yang telah berupa vektor tersebut memiliki informasi yang lebih kaya serta lebih mudah diamati daripada citra mentahnya (Li, 2005).

2.1.1 Representasi Wajah

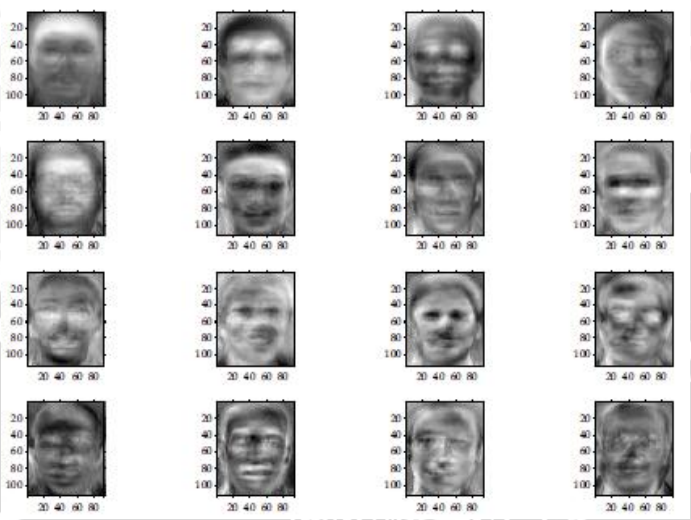
Sebelum dilakukan proses pengenalan, wajah terlebih dulu dibuat representasinya. Terdapat dua kategori dalam merepresentasikan wajah, yaitu: *global approach* atau *appearance-based*, yang menggunakan tekstur ciri dan diaplikasikan pada keseluruhan wajah atau daerah spesifik wajah, dan *feature-based* atau *component-based*, yang menggunakan relasi geometris antar ciri wajah seperti mulut, mata, dan hidung.

Principal Component Analysis(PCA) dan *Linear Discriminant Analysis* (LDA) adalah contoh representasi wajah yang termasuk dalam kategori *appearance-based*. Sedangkan contoh dari representasi kategori *component-based* adalah pendekatan dengan merepresentasikan wajah ke dalam model geometris dengan grafik elastis 2-D yang dilakukan oleh Wiskott (1997), serta pendekatan yang dilakukan oleh Brunelli dan Poggio (1993) dengan melakukan pencocokan tiga wilayah wajah (mata, mulut, dan hidung) secara independen dan konfigurasi ciri tersebut tidak dibatasi karena sistem tidak menyertakan model geometri(Delac, 2007).

Salah satu representasi wajah yang termasuk dalam kategori *appearance-based*, didasari oleh penerapan PCA dalam representasi wajah yang diajukan oleh Sirovich &Kirby pada tahun 1987. Metode ini diawali dengan penghitungan sistem koordinat terbaik untuk kompresi citra, dimana setiap koordinat sebenarnya merupakan sebuah citra yang disebut *eigenpicture*. Sirovich dan Kirby berpendapat bahwa, setiap kumpulan citra wajah dapat direkonstruksi dengan menyimpan sekumpulan kecil bobot untuk tiap wajah dan sekumpulan kecil *eigenpicture*(Turk,1991).

Berdasarkan metode PCA yang ditemukan tersebut, Turk dan Pentland (1991) mengembangkan sebuah metode pengenalan wajah menggunakan PCA, dengan representasi wajah yang disebut *eigenface*. *Eigenface* didapatkan melalui perhitungan PCA pada wajah pelatihan. Metode PCA banyak digunakan karena dapat mereduksi dimensi serta dapat merekam ciri-ciri utama yang sangat penting dalam proses pengenalan wajah (Delac, 2007). Representasi *Eigenface* untuk data latihan pada Gambar 2.2 ditunjukkan oleh Gambar 2.3.

Gambar 2.2 Contoh Wajah sebagai Data Latihan dari ORL Database (Sumber: Delac, 2007)



Gambar 2.3 *Eigenface* dari Data Latihan
(Sumber: Delac, 2007)

Proses pengenalan wajah menggunakan PCA tersebut dilakukan dengan mendekomposisi wajah menjadi sekumpulan gambar karakteristik ciri yang disebut *eigenface*. *Eigenface* dapat dianggap sebagai *principal component* dari *training set* awal citra wajah(Turk, 1991). *Principal component* merupakan sebuah variabel turunan yang memaksimalkan varian yang diperhitungkan dalam variabel asli (Dunteman, 1989).

Pengenalan dilakukan dengan memproyeksikan sebuah wajah baru ke dalam sebuah subruang yang disediakan oleh *eigenface* dan mengelompokkan wajah dengan membandingkan posisinya dalam ruang wajah terhadap individu-individu yang diketahui (Turk, 1991).

2.2 Konsep Dasar Principal Component Analysis (PCA)

Sebelum membahas cara kerja dan aplikasi PCA dalam pengenalan wajah, dalam subbab ini akan dibahas beberapa teori dasar matematika yang mendukung pemahaman terhadap cara kerja PCA. Teori-teori dasar tersebut meliputi: matriks, vektor dan ruang vektor, varian dan kovarian pada matriks, nilai eigen dan vektor eigen. Sedangkan pembahasan mengenai PCA akan dijelaskan pada subbab selanjutnya.

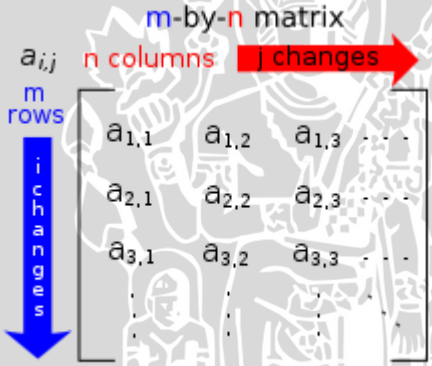
2.2.1 Matriks

Matriks adalah sekumpulan bilangan yang tersusun dalam baris dan kolom sehingga bentuknya menyerupai persegi panjang. Bilangan-

bilangan yang menyusun sebuah matriks disebut elemen atau anggota matriks. Matriks dapat dimanipulasi, seperti dijumlah, dikurangi, dikalikan dan didekomposisi. Contoh matriks dapat dilihat pada Gambar 2.4.

Matriks merupakan alat bantu yang sangat penting dalam berbagai bidang. Salah satu manfaat matriks adalah untuk merepresentasikan transformasi linear, dengan demikian persamaan $f(x) = cx$ dapat dinyatakan dalam dimensi yang lebih tinggi dengan menggunakan matriks, di mana c adalah sebuah konstanta.

Perkalian matriks bertalian erat dengan pembentukan transformasi linear. Matriks juga dapat melacak sebuah koefisien dalam sebuah sistem persamaan linear. Untuk sebuah matriks persegi (matriks yang memiliki jumlah baris dan kolom yang sama banyak), determinan dan invers matriks sangat membantu penemuan solusi dari sistem persamaan linear, sedangkan nilai eigen dan vektor eigen menyediakan wawasan geometris dari transformasi linear.



Gambar 2.4 Contoh Matriks

2.2.2 Operasi Dasar Matriks

Berikut merupakan beberapa operasi yang dapat diaplikasikan pada matriks, antara lain: penjumlahan matriks, perkalian skalar, dan transposisi (Brown, 1991).

1. Penjumlahan matriks

   Jumlah $A + B$ dari matriks A dan B berukuran mxn dihitung sesuai dengan posisinya: $(A + B)_{i,j} = A_{i,j} + B_{i,j}$ di mana $1 \leq i \leq m$ dan $1 \leq j \leq n$

2. Perkalian Skalar

   Perkalian skalar cA dari sebuah matriks A dan sebuah bilangan c didapatkan dengan mengalikan setiap elemen A dengan c

$$(cA)_{i,j} = c.A_{i,j} \tag{2.1}$$

3. Transpos Matriks

   Transpos matriks A yang berukuran mxn , dibentuk dengan mengubah baris dan kolom dan kolom menjadi baris, seperti yang ditunjukkan pada persamaan 2.2.

$$(A^T)_{i,j} = A_{j,i} \tag{2.2}$$

2.2.3 Perkalian matriks, persamaan linear, dan transformasi linear

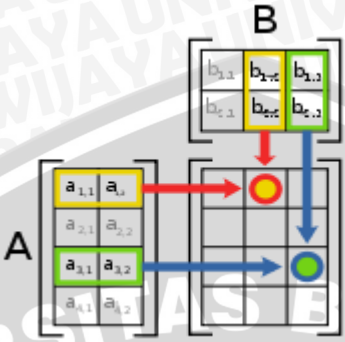
2.2.3.1 Perkalian Matriks

Perkalian dua matriks dapat dilakukan jika jumlah kolom matriks sebelah kiri sama dengan jumlah baris matriks sebelah kanan. Jika A adalah sebuah matriks berukuran mxn dan B adalah matriks berukuran nxp , maka AB adalah matriks berukuran mxp di mana elemennya merupakan hasil perkalian titik dari kolom A yang bersesuaian dengan kolom B. Hasil perkalian matriks AB dapat diperoleh melalui persamaan 2.3.

$$B]_{i,j} = A_{i,1}B_{1,i} + A_{i,2}B_{2,j} + \ ... + A_{i,n}B_{n,j} = \sum_{r=1}^{n} A_{i,r}B_{1,r} \tag{2.3}$$

Di mana $1 \leq i \leq m$ dan $1 \leq j \leq p$.

Perkalian tidak bersifat komutatif, $AB \neq BA$, namun demikian bersifat asosiatif $(A + B)C = AC + BC$ sama dengan $C(A + B) = CA + CB$. Langkah-langkah perkalian matriks digambarkan pada Gambar 2.5.



Gambar 2.5 Perkalian Matriks

2.2.3.2 Persamaan Linear

Perkalian matriks berkaitan erat dengan persamaan linear. jika x merupakan vektor kolom dari n variabel $x_1, x_2, \ldots, x_n$ dan A merupakan matriks mxn, maka persamaan matriks 2.4 sama dengan persamaan linear 2.5, di mana b adalah sebuah vektor kolom berukuran x x 1.

$$A_{1,1}x_1 + A_{1,2}x_2 + \cdots + A_{1,n}x_n = b_1 \quad (2.4)$$

...

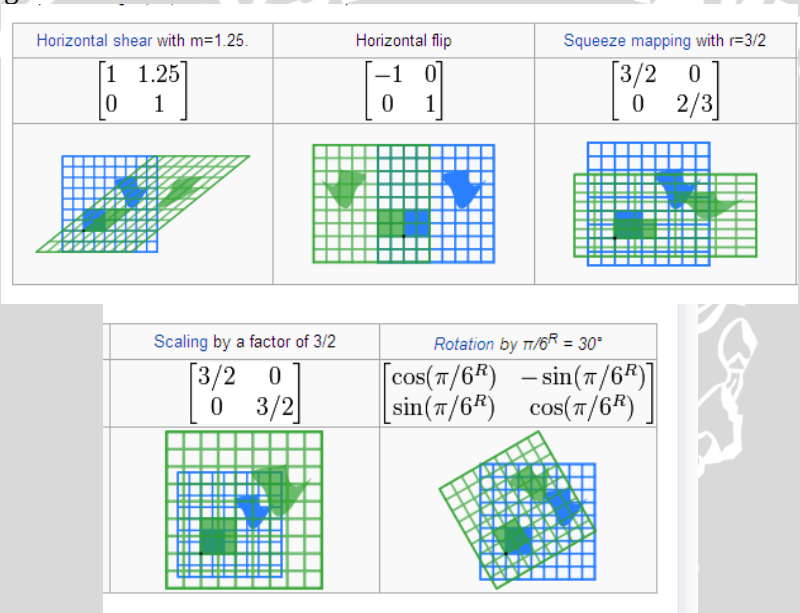
$$A_{m,1}x_1 + A_{m,2}x_2 + \cdots + A_{m,n}x_n = b_m \quad (2.5)$$

Dengan demikian matriks dapat digunakan untuk menyederhanakan persamaan linear majemuk.

2.2.3.3 Transformasi Linear

Matriks dan perkalian matriks sangat berperan dalam transformasi linear, atau yang disebut pemetaan linear. Sebuah matriks bilangan real, A, berukuran mx n yang ditransformasi oleh transformasi linear $R^n \rightarrow R^m$, akan memetakan setiap vektor x dalam R^n ke dalam hasil Ax, yang merupakan vektor dalam R^m. Sebaliknya, setiap transformasi linear f: $R^n \rightarrow R^m$ dibangkitkan dari sebuah matriks unik A berukuran mx n. Elemen ke (i, j) dari A adalah koordinat ke-i dari $f(e_j)$, di mana $e_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ adalah vektor satuan. Matriks A berperan untuk merepresentasikan pemetaan linear f, dan A adalah transformasi dari f.

Sebagai contoh, matriks A 2x2, $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$, dapat dilihat sebagai transformasi sebuah satuan persegi ke sebuah parallelogram dengan titik (0,0), (a, b), (a+a, b+d) dengan vektor kolom masing-masing $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, dan $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Vektor-vektor tersebut mendefinisikan titik pada bidang satuan persegi. Gambar 2.6 merupakan gambar yang menunjukkan sejumlah matrks 2x2 dengan pemetaan linear R^2. Gambar biru yang merupakan gambar asli dipetakan ke dalam gambar hijau. Titik asal (0,0) ditandai dengan titik hitam.



Gambar 2.6 Transformasi Matriks

2.3 Varian dan Kovarian matriks

Varian merupakan salah satu alat ukur persebaran data dalam dataset. Varian dapat ditemukan menggunakan persamaan 2.6

$$s^2 = \left(\sum_{i=1}^{n} (X_i - \bar{X})^2 \right) / (n-1) \qquad (2.6)$$

Varian adalah pengukur persebaran data dalam 1 dimensi, untuk mengetahui variasi antar dimensi yang saling berhubungan terhadap rata-ratanya, maka digunakanlah kovarian. Kovarian selalu diukur antara 2 dimensi. Jika kovarian dihitung antara satu dimensi itu sendiri, maka didapatkanlah varian. Persamaan kovarian hampir mirip dengan persamaan varian. Persamaan varian dapat juga dituliskan dalam persamaan 2.7.

$$var(X) = \left(\sum_{i=1}^{n} (X_i - \bar{X})(X_i - \bar{X}) \right) \Big/ (n-1) \quad (2.7)$$

Sedangkan persamaan kovarian adalah

$$cov(X,Y) = \left(\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y}) \right) \Big/ (n-1) \quad (2.8)$$

Persamaan 2.8 dapat berarti "Untuk setiap data, kalikan selisih antara nilai x dan rata-rata dari x dengan selisih antara nilai y dan rata-rata y. Jumlah seluruhnya, kemudian dibagi dengan $(n-1)$" (Smith, 2002).

Besar nilai kovarian tidak sepenting tandanya. Jika nilainya positif, maka hal tersebut mengindikasikan bahwa kedua dimensi memiliki hubungan yang berbanding lurus. Jika nilainya negatif, maka hubungan kedua dimensi tersebut berbanding terbalik. Jika kovariannya nol, maka kedua dimensi tersebut independen satu sama lain.

Kovarian matriks selalu diukur antara 2 dimensi. Jika terdapat sebuah dataset yang terdiri lebih dari 2 dimensi, maka terdapat lebih dari 2 kovarian yang ditemukan. Misalnya untuk dataset 3 dimensi (dimensi x, y, z) dapat dihitung $cov(x,y)$, $cov(x,z)$, dan $cov(y,z)$. Kovarian dalam data n-dimensi akan berjumlah $\frac{n!}{(n-2)!*2}$ yang berbeda.

Untuk memudahkan perhitungan kovarian di antara seluruh dimensi yang ada dalam dataset, maka digunakanlah matriks. kovarian matriks untuk sekumpulan data dengan dimensi *n* ditunjukkan oleh persamaan 2.9

$$C^{nxn} = \left(c_{i,j}, c_{i,j} = cov\left(Dim_i, Dim_j\right) \right) \quad (2.9)$$

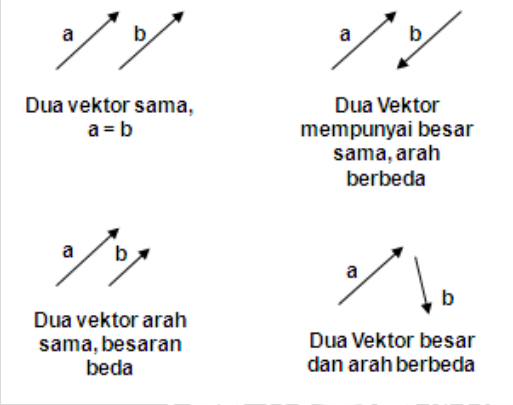
di mana C^{nxn} adalah sebuah matriks dengan baris dan kolom berukuran *n*, dan Dim_x adalah dimensi ke-x. Persamaan di atas menyatakan bahwa

jika terdapat dataset dengan dimensi *n*, maka matriks memiliki kolom dan baris sejumlah *n* dan setiap elemen dalam matriks adalah ahasil dari perhitungan kovarian di antara dua dimensi yang terpisah. Misalnya, elemen pada baris ke-2 kolom ke-3 adalah nilai kovarian antara dimensi ke-2 dan dimensi ke-3 (Smith, 2002).

2.4 Vektor

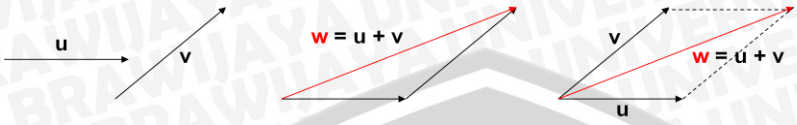
Vektor adalah sebuah obyek geometri yang memiliki besar dan arah. Vektor digambarkan menggunakan tanda panah (→). Panjang panah menggambarkan besar vektor, sedangkan arah panah menggambarkan arah vektor. Vektor berperan penting dalam menggambarkan posisi, kecepatan dan percepatan obyek, dan gaya.

Panjang sebuah vektor $u = \binom{a}{b}$ adalah $|u| = \sqrt{a^2 + b^2}$. Vektor dinyatakan sama jika memiliki arah dan besar yang sama. Gambar 2.7 menunjukkan pasangan-pasangan vektor yang memiliki arah dan besar yang berbeda-beda.



Gambar 2.7 Kesamaan Vektor

Vektor-vektor dapat dijumlahkan dengan mengikuti aturan segitiga dan aturan jajar genjang seperti pada Gambar 2.8.



Gambar 2.8 Penjumlahan Vektor

Jika dimisalkan dengan pasangan bilangan, $u = \binom{a}{b}$ dan $v = \binom{c}{d}$ maka penjumlahan vektor adalah $u + v = \binom{a}{b} + \binom{c}{d} = \binom{a+c}{b+d}$.

Vektor merupakan matriks yang hanya memiliki satu baris atau satu kolom saja. Vektor menjadi perhatian khusus karena digunakan untuk menyatakan penyelesaian dari sistem-sistem linear. Vektor-vektor kolom merupakan pernyataan penyelesaian-penyelesaian dari persamaan-persamaan matriks. Himpunan matriks-matriks *n*x1 dari bilangan-bilangan real disebut *Euclidean n-space*, yang biasanya dituliskan dengan R^n (Leon, 2001).

2.5 Vektor Eigen dan Nilai Eigen

Sebuah matriks persegi dapat memiliki vektor eigen dan nilai eigen yang dapat membantu menggambarkan sisi geometris dari sistem transformasi linear. Vektor eigen dari sebuah matriks persegi adalah sebuah vektor bukan nol yang jika dikalikan dengan matriks, menghasilkan vektor yang proporsional dengan matriks asal (yang berubah adalah besar vektor, bukan arah vektor). Setiap vektor eigen memiliki nilai eigen yang menunjukkan perubahan vektor eigen saat dikalikan dengan matriks.

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} x \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \end{bmatrix} \quad (2.9)$$

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} x \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} = 4x \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad (2.10)$$

$$2x \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \quad (2.11)$$

$$\begin{bmatrix} 2 & 3 \\ 2 & 1 \end{bmatrix} x \begin{bmatrix} 6 \\ 4 \end{bmatrix} = \begin{bmatrix} 24 \\ 16 \end{bmatrix} = 4x \begin{bmatrix} 6 \\ 4 \end{bmatrix} \quad (2.12)$$

Vektor eigen merupakan kasus khusus dalam perkalian dua matriks. Pada persamaan 2.9, hasil dari perkalian matriks bukan merupakan

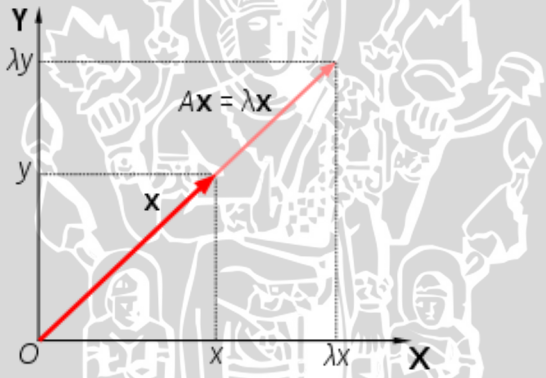
kelipatan dari vektor asli, sedangkan hasil dari perkalian matriks pada persamaan 2.10 tepat 4 kali dari vektor asli.

Vektor $\begin{bmatrix}3\\2\end{bmatrix}$ pada persamaan 2.11 dan 2.12 merepresentasikan sebuah perpindahan titik dari titik asal (0,0) ke titik (3,2). Matriks $\begin{bmatrix}2 & 3\\2 & 1\end{bmatrix}$ merupakan matriks yang simetris, dapat dianggap sebagai matriks transformasi.

Sebuah vektor dapat dianggap sebuah vektor eigen jika hasil kalinya dengan sebuah matriks persegi menghasilkan sebuah vektor yang merupakan kelipatan dari vektor asalnya. Secara matematis, vektor eigen dan nilai eigen dituliskan dalam persamaan 2.13.

$$A\mathbf{x} = \lambda\mathbf{x} \qquad (2.13)$$

A merupakan sebuah matriks persegi, $\mathbf{x}$ adalah sebuah vektor bukan nol, dan λ sebuah bilangan skalar.



Gambar 2.9 Ilustrasi vektor eigen

Dalam bidang kartesius, persamaan 2.13 dapat diumpamakan sebagai Gambar 2.9. Matriks A memperbesar vektor $\mathbf{x}$ dan tidak mengubah arahnya, maka $\mathbf{x}$ adalah vektor eigen dari A. Vektor eigen dan nilai eigen bergantung pada konsep vektor dan transformasi linear. Sebuah transformasi linear dapat digambarkan dengan sebuah matriks persegi.

Biasanya perkalian sebuah matriks dengan sebuah matriks persegi A mengubah arah dan juga besar dari vektor, namun dalam kondisi

tertentu jika vektor hanya berubah besarnya sedangkan arahnya tetap, atau mengubah vektor ke arah yang berlawan, maka vektor tersebut disebut vektor eigen dari matriks tersebut. Saat dikalikan dengan sebuah matriks, setiap vektor eigen dari matriks mengalami perubahan besar oleh sebuah faktor yang disebut nilai eigen yang bersesuaian dengan vektor eigen tersebut.

Karakteristik vektor eigen adalah sebagai berikut (Smith, 2002):

a) Vektor eigen hanya dapat ditemukan dalam matriks persegi. Dan tidak semua matriks persegi memiliki vektor eigen. Jika terdapat matriks berukuran *n* x *n* yang memiliki vektor eigen, maka matriks tersebut memiliki vektor eigen sebanyak *n.*
b) Jika sebuah vektor eigen diskala dengan sebuah nilai sebelum mengalikannya dengan sebuah vektor, maka hasilnya adalah adalah kelipatan dari hasil sebenarnya. Hal ini dikarenakan melakukan skala hanya menyebabkan vektor semakin panjang, dan bukan mengubah arahnya.
c) Seluruh vektor eigen dari matriks adalah ortogonal—saling tegak lurus antara satu sama lain berapapun dimensinya.

2.5.1 Nilai eigen dan vektor eigen pada matriks

Nilai eigen dari A adalah pendekatan solusi λ pada persamaan 2.14

$$\det(A - \lambda I) = 0 \tag{2.14}$$

Persamaan 2.14 disebut *characteristic equation* di mana det adalah determinan dari matriks, dan I adalah matriks identitas. Misalkan jika A adalah matriks diagonal, seperti pada persamaan 2.15, maka *characteristic equation* dapat dibaca seperti persamaan 2.16.

$$A = \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & a_{n,n} \end{bmatrix} \tag{2.15}$$

$$\det(A - \lambda I) = \det \begin{bmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & a_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & a_{n,n} \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} a_{1,1} - \lambda & 0 & \dots & 0 \\ 0 & a_{2,2} - \lambda & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & a_{n,n} - \lambda \end{bmatrix}$$

$$= (a_{1,1} - \lambda)(a_{2,2} - \lambda) \dots (a_{n,n} - \lambda) = 0 \tag{2.16}$$

Solusi persamaan 2.16 adalah berupa nilai-nilai eigen $\lambda_i = a_{i,i} (i = 1, \dots, n)$.

2.6 Proses Pencarian Nilai Eigen dan Vektor Eigen

Nilai eigen dan vektor eigen dapat dicari dengan pendekatan numerik. Secara garis besar, proses pencarian nilai eigen dan vektor eigen pada matriks persegi dan simetris, adalah sebagai berikut:

1. Mereduksi matriks menjadi bentuk tridiagonal menggunakan reduksi Householder untuk menyederhanakan perhitungan.
2. Mencari nilai eigen (λ) pada matriks tridiagonal
   a. Mencari nilai eigen dengan menyelesaikan persamaan determinan (mencari akar-akarnya). Proses pencarian nilai eigen melalui tahap-tahap sebagai berikut:
      i. Mencari interval-interval yang diduga memiliki nilai eigen.
         - Mencari dugaan nilai eigen minimal dan maksimal
         - Mencari banyaknya nilai eigen yang mungkin ada
      ii. Mencari akar persamaan dengan metode *bisection*, akar persamaan tersebut merupakan nilai eigen yang diingingkan.
   b. Mencari vektor eigen nilai-nilai eigen yang ditemukan
      i. Menyelesaikan persamaan $[A - \lambda I]t_i = 0$ dengan dekomposisi LU

ii. Mengalikan hasil penyelesaian persamaan (i) dengan matriks transformasi Householder untuk mendapatkan vektor eigen matriks yang sebenarnya.

2.6.1 Reduksi Householder

Perhitungan nilai eigen dan vektor eigen memiliki kompleksitas semakin besar jika dimensinya semakin besar. Untuk mengurangi kompleksitas perhitungan, matriks dapat ditransformasi terlebih dahulu ke dalam bentuk matriks tridiagonal. Transformasi matriks menjadi bentuk tridiagonal dapat dilakukan dengan menggunakan metode Householder.

Metode Householder yang diajukan oleh Alton Householder membantu mereduksi matriks simetris menjadi matriks tridiagonal. Transformasi Householder mereduksi matriks simetris menjadi bentuk tridiagonal dengan transformasi ortogonal sebanyak *n-2*.

Transformasi Householder menggunakan matriks Householder

$$\mathbf{Q} = \mathbf{I} - \frac{\mathbf{u}\mathbf{u}^T}{H} \tag{2.17}$$

di mana *H* adalah vektor dan didapatkan melalui persamaan 2.18

$$H = \frac{1}{2}\mathbf{u}^T\mathbf{u} = \frac{1}{2}|\mathbf{u}|^2 \tag{2.18}$$

Matriks **Q** adalah matriks simetris ($\mathbf{Q}^T = \mathbf{Q}$) dan juga ortogonal. Jika *x* merupakan vektor sebarang, maka untuk menghitung transformasi **Qx** dipilih nilai **u** dengan menggunakan persamaan 2.19

$$\mathbf{u} = \mathbf{x} + k\mathbf{e}_1 \tag{2.19}$$

di mana $k = \pm|\mathbf{x}|$ dan $\mathbf{e}_1 = [1\ 0\ 0\ \ldots\ 0]^T$.

Dengan demikian didapatkan persamaan 2.20

$$\mathbf{Q}\mathbf{x} = \left(\mathbf{I} - \frac{\mathbf{u}\mathbf{u}^T}{H}\right)\mathbf{x} = \left[\mathbf{I} - \frac{\mathbf{u}(\mathbf{x}+k\mathbf{e}_1)^T}{H}\right]\mathbf{x} = \mathbf{x} - \frac{\mathbf{u}(\mathbf{x}+k\mathbf{x}_1)}{H} \tag{2.20}$$

Dengan menggunakan persamaan 2.18 dan 2.19, maka nilai $2H = 2(k^2 + kx_1)$, sehingga

$$\mathbf{Q}\mathbf{x} = \mathbf{x} - \mathbf{u} = -k\mathbf{e}_1 = [-k \quad 0 \quad 0 \quad \ldots \quad 0]^T \tag{2.21}$$

dengan demikian transfomasi **Qx** mengeliminasi elemen dari x kecuali elemen pertama.

Pada matriks simetris, reduksi Householder mengikuti pola transformasi 2.22

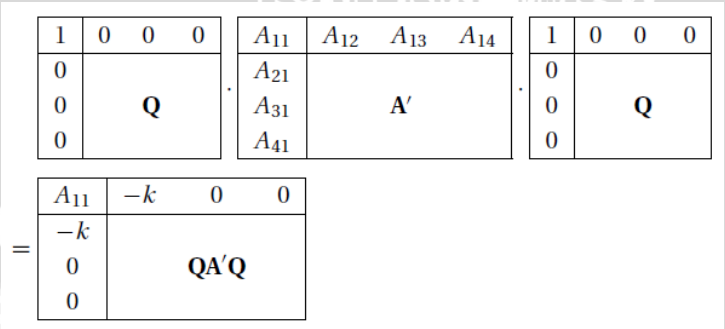
$$\mathbf{P}_1\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0}^T \\ \mathbf{0} & \mathbf{Q} \end{bmatrix} \begin{bmatrix} \mathbf{A_{11}} & \mathbf{x}^T \\ \mathbf{x} & \mathbf{A}' \end{bmatrix} = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{x}^T \\ \mathbf{Qx} & \mathbf{QA}' \end{bmatrix} \tag{2.22}$$

x merepresentasikan kolom pertama dari **A** dengan menghilangkan elemen pertama, dan **A′** adalah **A** dengan kolom dan baris pertama dihilangkan. Matriks **Q** dengan dimensi $(n-1)x(n-1)$ disusun menggunkan persamaan 2.17 dan 2.18. mengacu pada persamaan 2.20, maka dapat dilihat bahwa transformasi mereduksi kolom pertama matriks A menjadi persamaan 2.23

$$\begin{bmatrix} A_{11} \\ \mathbf{Qx} \end{bmatrix} = \begin{bmatrix} A_{11} \\ -k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.23}$$

$$A \leftarrow P_1 A P_1 = \begin{bmatrix} \mathbf{A_{11}} & \mathbf{(Qx)}^T \\ \mathbf{Qx} & \mathbf{QA'Q} \end{bmatrix} \tag{2.24}$$

Transformasi 2.23 mendiagonalsiasi baris dan sekaligus kolom pertama matriks A. Diagram transformasi matriks 4x4 ditunjukkan pada Gambar 2.10



Gambar 2.10 Diagram transformasi pada matriks 4x4
(Sumber: Kiusalaas, 2009)

```
function [c,d,P] = householder(A)
% Housholder reduction of A to tridiagonal form A -> [c\d\c].
% USAGE: [c,d] = householder(A)  computes c and d only.
%      [c,d,P] = householder(A)  also computes the transformation
%                                matrix P.

% Householder reduction
   n = size(A,1);
   for k = 1:n-2
       u = A(k+1:n,k);
       uMag = sqrt(dot(u,u));
       if u(1) < 0; uMag = -uMag; end
       u(1) = u(1) + uMag;
       A(k+1:n,k) = u;             % Save u in lower part of A
       H = dot(u,u)/2;
       v = A(k+1:n,k+1:n)*u/H;
       g = dot(u,v)/(2*H);
       v = v - g*u;
       A(k+1:n,k+1:n) = A(k+1:n,k+1:n) - v*u' - u*v';
       A(k,k+1) = -uMag;
   end
   d = diag(A); c = diag(A,1);
   if nargout == 3; P = transMatrix; end

   % Computation of the transformation matrix
   function P = transMatrix
   P = eye(n);
   for k = 1:n-2
       u = A(k+1:n,k);
       H = dot(u,u)/2;
       v = P(1:n,k+1:n)*u/H;
       P(1:n,k+1:n) = P(1:n,k+1:n) - v*u';
   end
   end
end
```

Gambar 2.11 Prosedur reduksi Householder pada MATLAB (Sumber: Kiusalaas, 2009)

Langkah langkah reduksi Householder adalah sebagai berikut (Kiusalaas, 2009):

1. $\mathbf{A}'$ berisi bagian bawah kanan matriks $\mathbf{A}$, berukuran $(n-i)x(n-i)$
2. $\mathbf{x} = [\mathbf{A}_{i+1,i} \quad \mathbf{A}_{i+2,i} \quad \cdots \quad \mathbf{A}_{n,i}]^T$ yang merupakan kolom dengan panjang $n-i$ yang terletak di sebelah kiri A'

3. Menghitung $|\mathbf{x}|$. Jika $x_1 > 0$, maka $k = |\mathbf{x}|$, dan $k = -|\mathbf{x}|$ jika $x_1 < 0$. Pemilihan tanda untuk k, bertujuan untuk membantu meminimalkan kesalahan.
4. $\mathbf{u} = [k + x_1 \quad x_2 \quad x_3 \quad \dots \quad x_{n-i}]^T$
5. Menghitung $H = \frac{|\mathbf{u}|}{2}$
6. Menghitung $\mathbf{v} = \mathbf{A}'\mathbf{u}/H$
7. Menghitung $g = \frac{\mathbf{u}^T\mathbf{v}}{(2H)}$
8. Menghitung $\mathbf{w} = \mathbf{v} - g\mathbf{u}$
9. Menghitung transformasi $\mathbf{A}' \rightarrow \mathbf{A}' - \mathbf{w}^{\mathbf{T}}\mathbf{u} - \mathbf{u}^{\mathbf{T}}\mathbf{w}$
10. Mengubah nilai $\mathbf{A}_{i,i+1} = \mathbf{A}_{i+1,i} = -k$

Contoh aplikasi metode Householder pada MATLAB ditunjukkan oleh Gambar 2.11

Transformasi household er merupakan transformasi kemiripan matriks, maka nilai eigen pada matriks hasil transformasi dan matriks asli memiliki nilai yang sama. Namun demikian untuk mendapatkan vektor eigen matriks asli, maka vektor eigen dari matriks tridiagonal harus dikalikan dengan matriks transformasi.

Untuk menghitung matriks transformasi, diperlukan langkah-langkah sebagai berikut:

1. Mengambil $\mathbf{u}$ dari proses tridiagonalisasi
2. Menghitung $H = \frac{|\mathbf{u}|}{2}$
3. Menghitung $\mathbf{y} = \mathbf{P}'\mathbf{u}/H$
4. Menghitung transformasi $\mathbf{P}' \leftarrow \mathbf{P}' - \mathbf{y}\mathbf{u}^T$

Hasil transformasi berupa matriks $\mathbf{P}$ akan dikalikan dengan vektor eigen matriks tridiagonal untuk mendapatkan vektor eigen matriks asal. Perhitungan tersebut dapat digambarkan oleh persamaan 2.25.

$$X = \mathbf{P}X_{tridiag} \tag{2.25}$$

2.6.2 Mencari nilai eigen dan vektor eigen matriks tridiagonal

Pencarian nilai eigen dan vektor eigen dapat dilakukan dengan menggunakan metode pangkat(*power method*) atau metode pangkat terbalik (*inverse power method*). Metode tersebut merupakan metode pendekatan secara iteratif, sehingga memerlukan tebakan awal untuk mendekati hasil dari nilai eigen yang sebenarnya. Untuk menemukan tebakan awal nilai eigen, dilakukan sebuah penyelesaian iteratif untuk menyelesaikan persamaan determinan.

2.6.2.1 Menyelesaikan persamaan determinan

Pada dasarnya, nilai eigen dari matriks A dapat ditemukan dengan menemukan akar dari persamaan karakteristik $|\mathbf{A}-\lambda\mathbf{I}|=0$. Metode tersebut tidak praktis untuk matriks yang memiliki dimensi besar, karena determinan memerlukan perkalian sebanyak $\frac{n^3}{3}$. Namun demikian, jika matriks berbentuk tridiagonal dan simetris, karakteristik polinomial 2.26 dapat dihitung dalam $3(n-1)$ perhitungan dengan menggunakan deret operasi 2.27.

$$P_n(\lambda)=|\mathbf{A}-\lambda\mathbf{I}|=\begin{vmatrix} d_1-\lambda & c_1 & 0 & 0 & \cdots & 0 \\ c_1 & d_2-\lambda & c_1 & 0 & \cdots & 0 \\ 0 & c_1 & d_3-\lambda & c_1 & \cdots & 0 \\ 0 & 0 & c_1 & d_4-\lambda & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & c_1 & d_n-\lambda \end{vmatrix} \tag{2.26}$$

$$\begin{aligned} P_0(\lambda) &= 1 \\ P_1(\lambda) &= d_1-\lambda \\ P_i(\lambda) &= (d_i-\lambda)P_{i-1}(\lambda)-c_{i-1}^2P_{1-2}(\lambda), \quad i=2,3,\dots,n \end{aligned} \tag{2.27}$$

Polinomial $P_0(\lambda), P_1(\lambda), \dots, P_n(\lambda)$ membentuk deret Sturm yang memiliki ciri sebagai berikut: Jumlah dari perubahan tanda pada deret $P_0(a), P_1(a), \dots, P_n(a)$ adalah sama dengan jumlah akar dari $P_n(\lambda)$ yang lebih kecil dari a. Jika sebuah anggota $P_{i-1}(a)$ dari deret adalah 0, maka tandanya adalah berlawanan dengan $P_{i-1}(a)$.

Deret Sturm digunakan untuk menemukan batasan/range nilai-nilai eigen pada matriks tridiagonal. Deret akhir pada deret Sturm merupakan bentuk solusi persamaan determinan, jika hasil dari persamaan tersebut adalah nol. Aplikasi Deret Sturm pada MATLAB ditunjukkan pada Gambar 2.12.

```
function p = sturmSeq(c,d,lambda)
% Returns Sturm sequence p associated with
% the tridiagonal matrix A = [c\d\c] and lambda.
% USAGE: p = sturmSeq(c,d,lambda).
% Note that |A - lambda*I| = p(n).

n = length(d) + 1;
p = ones(n,1);
p(2) = d(1) - lambda;
for i = 2:n-1
    p(i+1) = (d(i) - lambda)*p(i) - (c(i-1)^2 )*p(i-1);
end
```

Gambar 2.12 Prosedur deret Sturm pada MATLAB
(Sumber: Kiusalaas, 2009)

Deret Sturm dapat menunjukkan banyaknya solusi nilai eigen (ditunjukkan dengan banyaknya perubahan tanda), dan dapat membantu menemukan solusi dari persamaan determinan $|\mathbf{A} - \lambda\mathbf{I}|$ secara iteratif, terhadap sebuah nilai λ tertentu. Dengan demikian nilai λ harus dapat ditebak sedemikan rupa hingga nilai tersebut menjadikan baris akhir deret Sturm untuk matriks A bernilai 0. Untuk menebak batasan nilai eigen maksimal dan minimal, dapat digunakan teorema Greschgorin. Sedangkan untuk menebak nilai λ yang paling sesuai, dapat digunakan metode *bisection*.

2.6.2.1.1 Teorema Greschgorin

Teorema Greschgorin bermanfaat untuk menentukan batasan global dari nilai eigen dalam matriks A berukuran nxn. Istilah global berarti bahwa batas tersebut mencakup seluruh nilai eigen. Teorema sederhana yang berlaku untuk matriks simetris adalah sebagai berikut:

Jika λ adalah sembarang nilai eigen dari A, maka $a_i - r_i \leq \lambda \leq a_i + r_i, \quad i = 1, 2, \dots, n$ di mana didapatkan melalui persamaan 2.27.

$$a_i = A_{ii} \qquad r_i = \sum_{\substack{j=1 \\ j \neq 1}}^{n} |A_{ij}| \tag{2.27}$$

Berdasarkan persamaan 2.27, batas global nilai-nilai eigen ditunjukkan oleh persamaan 2.28.

$$\lambda_{min} \geq \min_i(a_i - r_i) \qquad \lambda_{max} \leq \max_i(a_i + r_i) \tag{2.28}$$

2.6.2.1.2 Metode Bisection

Metode *bisection* merupakan salah satu metode untuk menghitung akar persamaan secara iteratif. Metode *bisection* menyelesaikan permasalahan dengan membagi dua interval hingga sekecil mungkin. Oleh karenanya, teknik ini disebut sebagai metode pembagian interval (*interval halving method*). *Bisection* bukan merupakan metode tercepat, tetapi cukup mendekati hasil yang sebenarnya. Langkah-langkah metode ini adalah sebagai berikut (Kiusalaas, 2009) :

1. x_1 adalah batas atas dan x_2 adalah batas bawah dari interval yang diduga memiliki akar. Dengan demikian, nilai dari $f(x_1).f(x_2) < 0$.
2. Membagi dua interval yang telah ada, dengan mencari nilai $x_3 = \frac{1}{2}(x_1 + x_2)$
3. Melakukan pengecekan, jika nilai $f(x_2).f(x_3) < 0$, maka akar terdapat pada interval (x_2, x_3), jika tidak, maka akar akan terdapat pada interval (x_1, x_3).
4. Mengulangi kembali langkah 2-3 hingga interval menjadi sangat kecil (mendekati nilai ε), sehingga $|x_2 - x_1| \leq \varepsilon$

2.6.2.2 Mencari vektor eigen dari setiap nilai eigen yang ditemukan

Pada matriks *NxN* penyelesaian determinan pada persamaan 2.12 memerlukan perhitungan yang cukup kompleks karena determinan matriks *NxN* menghasilkan persamaan determinan berderajat *N*. Untuk mendapatkan nilai eigen dan vektor eigen dari persamaan 2.13 dan 2.14 diperlukan sebuah metode tertentu. Metode pangkat merupakan salah satu metode iteratif yang cukup baik dalam nyelesaikan pencarian nilai dan vektor eigen pada matriks *NxN*, jika nilai eigen telah diduga sebelumnya (Kiusalaas, 2009).

Untuk mempercepat konvergensi pada metode pangkat, digunakan teknik nilai eigen *shifting*. Nilai eigen *shifing* "menggeser" nilai eigen dengan persamaan 2.29.

$$\lambda = \lambda^* + t \tag{2.29}$$

t merupakan nilai pergeseran yang ditentukan sebelumnya. Dengan demikian, persamaan 2.13 dapat dituliskan menjadi persamaan 2.30.

$$\boldsymbol{Av} = (\lambda^* + t)\, v \tag{2.30}$$

atau

$$\boldsymbol{A}^* v = \lambda^* v \tag{2.31}$$

di mana

$$\boldsymbol{A}^* = \boldsymbol{A} - t\mathbf{I} \tag{2.32}$$

Jika persamaan 2.30 diselesaikan dengan metode pangkat akan menghasilkan λ_1^* dan v_1, dimana λ_1^* adalah nilai eigen terbesar dari $\boldsymbol{A}^*$. Nilai eigen yang sesuai untuk permasalahan asal $\lambda = \lambda^* + t$ adalah nilai eigen yang terdekat dengan t.

Eigenvalue shifting memiliki dua kegunaan. Kegunaan pertama adalah dapat menemukan nilai eigen yang terdekat dengan nilai t. Kedua, *eigenvalue shifting* juga digunakan untuk mempercepat konvergensi. Jika dimisalkan dicari nilai eigen terkecil dari matriks $\boldsymbol{A}$ adalah λ_1, maka untuk menemukannya adalah dengan menemukan sebuah nilai t yang membuat λ_1^*/λ_2^* sekecil mungkin. Karena $\lambda_1^* = \lambda_1 - t$, maka harus dipilih $t \approx \lambda_1$. Dengan demikian dapat disimpulkan metode ini akan dapat bekerja dengan baik jika nilai λ_1 telah diperkirakan sebelumnya (Kiusalaas, 2009).

Setelah teknik *eigenvalue shifting* diterapkan, selanjutnya persamaan 2.28 dicari nilai eigen dan vektor eigennya dengan metode pangkat atau metode pangkat terbalik. Metode pangkat mencari nilai eigen yang terbesar terlebih dahulu, sedangkah metode pangkat terbalik mencari nilai eigen terkecil lebih dahulu. Langkah-langkah metode pangkat adalah:

1. x merupakan pendekatan terhadap v_n (didekati dengan sebuah vektor random)
2. Menyelesaikan $z = \boldsymbol{A}x$ untuk menemukan vektor z
3. Menghitung $|z|$
4. Menghitung $x = \frac{z}{|z|}$ dan mengulangi langkah 2-4 hingga selisih perubahan x bernilai sangat kecil (hingga dapat diabaikan)

Sedangkan metode pangkat terbalik memiliki langkah-langkah sebagai berikut:

1. x merupakan pendekatan terhadap v_n (didekati dengan sebuah vektor random)
2. Menyelesaikan $\boldsymbol{A}z = x$ untuk menemukan vektor z
3. Menghitung $|z|$
4. Menghitung $x = \frac{z}{|z|}$ dan mengulangi langkah 2-4 hingga x bernilai sangat kecil (hingga dapat diabaikan)

Seperti halnya teknik iteratif yang lain, metode pangkat menggunakan tebakan awal untuk memulai pencarian hasil. Tebakan awal berupa vektor random yang memiliki nilai antara 0 dan 1. Hasil akhir dari langkah-langkah di atas adalah $|z| = \pm\frac{1}{\lambda_1}$ dan$x = v_1$. Tanda dari λ_1 dapat didapatkan sebagai berikut: jika z mengalami perubahan tanda selama iterasi, makaλ_1 adalah negatif, jika sebaliknya maka tanda dari λ_1 adalah positif (Kiusalaas, 2009).

Contoh prosedur metode pangkat terbalik pada MATLAB ditunjukkan pada Gambar 2.13

```
function [eVal,eVec] = invPower(A,s,maxIter,tol)
% Inverse power method for finding the eigenvalue of A
% closest to s & the corresponding eigenvector.
% USAGE: [eVal,eVec] = invPower(A,s,maxIter,tol)
% maxIter = limit on number of iterations (default is 50).
% tol = error tolerance (default is 1.0e-6).

if nargin < 4; tol = 1.0e-6; end
if nargin < 3; maxIter = 50; end
n = size(A,1);
A = A - eye(n)*s;    % Form A* = A - sI
A = LUdec(A);        % Decompose A*
x = rand(n,1);       % Seed eigenvecs. with random numbers
xMag = sqrt(dot(x,x)); x = x/xMag;      % Normalize x
for i = 1:maxIter
    xOld = x;        % Save current eigenvecs.
    x = LUsol(A,x); % Solve A*x = xOld
    xMag = sqrt(dot(x,x)); x = x/xMag;  % Normalize x
    xSign = sign(dot(xOld,x)); % Detect sign change of x
    x = x*xSign;
    % Check for convergence
    if sqrt(dot(xOld - x,xOld - x)) < tol
        eVal = s + xSign/xMag; eVec = x;
        return
    end
end
error('Too many iterations')
```

Gambar 2.13 Prosedur metode pangkat terbalik pada MATLAB (Sumber: Kiusalaas, 2009)

2.6.2.3 Dekomposisi Matriks

Dekomposisi matriks merupakan sebuah usaha mentransformasikan matriks ke dalam bentuk tertentu agar mempermudah proses komputasi dan menyederhanakan analisis. Perhitungan matriks yang cukup rumit misalnya pada perhitungan determinan matriks, memerlukan sebuah penguraian matriks agar matriks yang akan dihitung memiliki dimensi yang lebih rendah.

Salah satu bentuk dekomposisi matriks adalah dekomposisi LU. Dekomposisi LU menyusun ulang matriks menjadi sebuah matriks segitiga bawah L dan sebuah matriks segitiga atas U yang ditunjukkan pada persamaan 2.33

$$\mathbf{A = LU} \tag{2.33}$$

Persamaan 2.34 merupakan sebuah contoh dekomposisi LU pada matriks 3x3.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \tag{2.34}$$

Dekomposisi LU dapat diperoleh dengan menggunakan algoritma Doolittle. Algoritma Doolittle secara garis besar dapat dibagi ke dalam dua tahap, yaitu tahap dekomposisi dan tahap penyelesaian.

Penggunaan Algoritma Doolittle untuk menyelesaikan kasus matriks tridiagonal adalah sebagai berikut (Kiusalaas, 2009):

1. Tahap dekomposisi
   a. Menggunakan eliminasi Gauss untuk menghasilkan matriks segitiga atas
   b. Elemen L yang terletak di bawah diagonal adalah pengali dari persamaan pivot yang digunakan selama eliminasi Gauss, di mana l_{ij} merupakan pengali yang mengeliminasi a_{ij}

Gambar 2.14 menunjukkan pengaplikasian dekomposisi LU pada MATLAB

```
function [c,d,e] = LUdec3(c,d,e)
% LU decomposition of tridiagonal matrix A = [c\d\e].
% USAGE: [c,d,e] = LUdec3(c,d,e)

n = length(d);
for k = 2:n
    lambda = c(k-1)/d(k-1);
    d(k) = d(k) - lambda*e(k-1);
    c(k-1) = lambda;
end
```

Gambar 2.14 Dekomposisi LU dalam MATLAB
(Sumber: Kiusalaas, 2009)

2. Tahap penyelesaian

Tahap penyelesaian terdiri dari *forward substitution* dan *back substitution*. Contoh aplikasi penyelesaian dekomposisi LU pada MATLAB ditunjukkan pada Gambar 2.15.

```
function x = LUsol3(c,d,e,b)
% Solves A*x = b where A = [c\d\e] is the LU
% decomposition of the original tridiagonal A.
% USAGE: x = LUsol3(c,d,e,b)

n = length(d);
for k = 2:n             % Forward substitution
    b(k) = b(k) - c(k-1)*b(k-1);
end
b(n) = b(n)/d(n);       % Back substitution
for k = n-1:-1:1
    b(k) = (b(k) -e(k)*b(k+1))/d(k);
end
x = b;
```

Gambar 2.15 Penyelesaian Dekomposisi LU dalam MATLAB (Sumber: Kiusalaas, 2009)

Setelah nilai eigen dan vektor eigen ditemukan, maka untuk mendapatkan vektor eigen dari matriks yang sebenarnya, maka vektor eigen matriks tridiagonal harus dikalikan dengan matriks transformasi Householder.

2.7 Principal Component Analysis

PCA merupakan sebuah metode statistika yang berfungsi untuk menemukan pola pada sekelompok data dan mengekspresikan data-data tersebut sedemikan rupa sehingga perbedaan dan persamaan antara data-data tersebut dapat terlihat dengan jelas (Smith, 2002).

Ide dasar PCA adalah untuk mereduksi dimensionalitas dari dataset yang mengandung sejumlah besar variabel yang berhubungan, namun tetap mempertahankan sebanyak mungkin variasi yang terdapat dalam dataset. Hal ini dilakukan dengan mentransformasikannya ke dalam sekumpulan variabel baru yang disebut *principal component*. *Principal component* bersifat tak terkorelasi dan terurut sedemikian hingga *principal component* pertama mencakup sebanyak mungkin variasi yang terdapat dalam seluruh variabel asli (Jolliffe, 2002).

Tujuan dari PCA adalah untuk mereduksi dimensi dari sekumpulan data asli. Sejumlah kecil data variabel tak terkorelasi, lebih mudah dimengerti dan digunakan dalam analisis lebih lanjut daripada sejumlah besar variabel terkorelasi. Ide tersebut diusulkan oleh Pearson (1901) dan kemudian dikembangkan oleh Hotelling (1933).

Walaupun PCA tidak mengabaikan kovarian dan korelasi, PCA lebih berkonsentrasi pada varian. Untuk menyusutkan dimensi peubah asal X dapat dilakukan dengan membentuk peubah baru $Y = \alpha_1\chi_1 + \alpha_2\chi_2 + \cdots + \alpha_p\chi_p$ atau $Y = \alpha'$X, di mana α adalah matriks transformasi yang mengubah peubah asal X menjadi peubah baru Y yang disebut *principal component*. Syarat untuk membentuk *principal component* yang merupakan kombinasi linear dari peubah X agar mempunyai varian yang besar adalah dengan memilih α' sedemikian hingga $var\,(Y) = \alpha'\Sigma\alpha$ dengan $\alpha'\alpha = 1$ (Joliffe, 2002).

Secara umum *principal component* ke-i adalah kombinasi linear terbobot peubah asal yang mampu menerangkan keragaman data ke-i. *Principal component* tersebut dapat dituliskan ke dalam persamaan 2.35.

$$Y_i = \alpha_{i1}\chi_1 + \alpha_{i2}\chi_2 + \cdots + \alpha_{ip}\chi_p \qquad i = 1,2,\ldots,p \tag{2.35}$$

α_i merupakan vektor eigen dari matriks kovarian X yang bersesuaian dengan nilai eigen λ_p. α_i dipilih sedemikian hingga memiliki panjang satuan ($\alpha_i'\,\alpha_i = 1$), dan $var(Y_i) = \lambda_i$.

Pada pengolahan citra, PCA dapat digunakan untuk mereduksi dimensi gambar sehingga memudahkan observasi. Dalam pengenalan wajah, PCA bertujuan untuk menangkap variasi total di dalam kumpulan wajah yang dilatihkan dan mendeskripsikan variasi tersebut dengan variabel yang sedikit (Gunadi, 2001).

2.8 Perhitungan PCA

Secara umum, algoritma PCA adalah sebagai berikut (Smith, 2002):

1. Mendapatkan data pelatihan
2. Normalisasi data

   Agar PCA bekerja dengan baik, perlu dilakukan pengurangan setiap data dengan rata-ratanya. Setiap data dikurangi dengan rata-rata setiap dimensi. Proses ini akan menghasilkan sebuah dataset yang memiliki rata-rata total 0.
3. Menghitung matriks kovarian dari matriks yang telah dinormalisasi.

4. Menghitung vektor eigen dan nilai eigen dari matriks kovarian.

   Karena matriks kovarian merupakan matriks persegi, maka vektor eigen dan nilai eigen dapat diketahui. Panjang dari setiap unit vektor eigen adalah satu.
5. Memilih komponen dan membentuk sebuah vektor ciri.

   Pada tahap inilah letak pengurangan dimensi dan kompresi. Vektor eigen dengan nilai eigen tertinggi merupakan *principal component* dari dataset. Oleh karena itu, langkah yang biasa dilakukan setelah vektor eigen ditemukan dari matriks kovarian, adalah mengurutkannya berdasar nilai eigen dari besar ke kecil. Setelah itu, dapat diputuskan apakah vektor eigen dengan nilai eigen kecil akan diabaikan atau tidak. Jika diabaikan, maka hal tersebut tidak akan menjadi masalah karena nilai eigen-nya kecil. Hal ini berarti bahwa vektor eigen tersebut kurang signifikan. Dengan hanya menggunakan komponen utama data yang sangat signifikan (nilai eigen besar), maka dataset akhir akan terkurangi dimensinya. Kumpulan vektor eigen terpilih disatukan dalam sebuah matriks yang biasa disebut dengan vektor ciri (*feature vector*).
6. Memperoleh data set baru

   Setelah vektor eigen (komponen) telah dipilih dan telah dibentuk dalam sebuah vektor ciri, kemudian melakukan transpose dari vektor dan mengalikannya dengan data set asli dari sebelah kiri, yang ditranspose.

$$Final\ Data = RowFeatureVector \times RowData\ Adjust \tag{2.36}$$

   Di mana *RowFeatureVector* adalah matriks dengan vektor eigen dalam kolom yang telah ditranspose sehingga vektor eigen dalam posisi baris, dengan vektor eigen paling signifikan terletak paling atas. *RowDataAdjust* adalah data yang telah dinormalisasi kemudian ditranspose.

2.9 Aplikasi PCA pada pengenalan wajah

Secara garis besar, pengenalan wajah menggunakan pendekatan PCA memiliki dua fase, yaitu fase pelatihan dan fase klasifikasi. Pada fase pelatihan, *eigenface* didapatkan dari sampel pelatihan menggunakan PCA dan citra wajah yang dilatih dipetakan ke dalam ruang eigen untuk kemudian diklasifikasikan. Pada fase klasifikasi, sebuah wajah masukan

diproyeksikan ke dalam ruang eigen yang sama dan dikelompokkan sesuai dengan klasifikasi yang sesuai(Delac, 2007).

2.9.1 Fase Pelatihan

Hasil akhir dari fase pelatihan ini adalah representasi wajah yang berupa citra *eigenface*. Citra *eigenface* didapatkan dari vektor eigen L, dimana L merupakan matrix yang berisi nilai *traning set* dasar. Dalam konteks matematika, proses pelatihan ditujukan untuk menemukan *principal component* dari distribusi wajah, atau vektor eigen dari matriks kovarian L. Pada proses ini, gambar diperlakukan sebagai vektor dalam dimensi ruang yang sangat besar. Setiap vektor eigen memuat nilai variasi citra yang berbeda-beda. Vektor eigen-vektor eigen tersebut dapat dianggap sebagai sekumpulan ciri yang secara bersama-sama mencirikan variasi antar gambar wajah. *Eigenface* terbentuk dari vektor eigen-vektor eigen tersebut.

Setiap wajah tunggal dapat direpresentasikan dengan tepat ke dalam sebuah kombinasi linear *eigenface*. Setiap wajah juga dapat juga didekati dengan hanya menggunakan *eigenface* terbaik, yaitu yang memiliki nilai eigen terbesar, yang mana merupakan nilai variasi terbesar dalam sekumpulan gambar. M *eigenface* terbaik memerlukan dimensi ruang bagian sebesar M.

Untuk menemukan *eigenface*, maka perlu diketahui nilai eigen dan vektor eigen yang dimiliki oleh citra wajah pelatihan. Setiap citra wajah yang berukuran NxM dijadikan sebuah vektor berukuran 1x(*NxM*).

Sebelum dilakukan analisis PCA sekumpulan citra wajah pelatihan Γ_1, Γ_2, …, Γ_M, terlebih dahulu dicari rata-rata wajahnya menggunakan persamaan 2.37. Rata-rata wajah tersebut digunakan untuk mencari selisih setiap wajah dengan rata-rata dengan persamaan 2.38. Selisih tiap wajah membentuk sebuah vektor Φ.

$$\psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n \tag{2.37}$$

$$\Phi_i = \Gamma_i - \Psi \tag{2.38}$$

Pada vektor-vektor wajah Φ ini kemudian dilakukan analisis PCA yang bertujuan untuk mencari sekumpulan vektor ortonormal u_n sebanyak *M* yang paling baik mendeskripsikan distribusi data. vektor *u*

ke-*k* dipilih sehingga persamaan 2.39 memiliki nilai maksimum terhadap kondisi 2.40.

$$\lambda_k = \frac{1}{M}\sum_{n-1}^{M}\left(u_k^T \Phi_n\right)^2 \tag{2.39}$$

$$u_l^T u_k = \delta_{lk} = \begin{cases} 1, & jika\ l = k \\ 0, & selainnya \end{cases} \tag{2.40}$$

Vektor u_k dan skalar λ_k masing-masing adalah vektor eigen dan nilai eigen dari matriks kovarian C yang didapatkan melalui persamaan 2.41.

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T \tag{2.41}$$

Matriks A=[$\Phi_1 \Phi_2 \ldots \Phi_M$]. Matriks *C* berukuran N^2xN^2 dan dengan demikian memiliki vektor eigen dan nilai eigen sebanyak N^2. Menghitung N^2 vektor eigen dan nilai eigen merupakan tugas yang tidak mudah. Oleh karena itu diperlukan metode perhitungan yang lebih mudah.

Jika jumlah data dalam ruang gambar lebih kecil daripada dimensi ruangnya, ($M < N^2$), maka akan terdapat hanya M-1 vektor eigen yang berarti. (Nilai dari vektor eigen yang lain memiliki nilai eigen 0). Dengan demikian vektor eigen berdimensi N^2 dapat diselesaikan dengan terlebih dahulu menyelesaikan vektor eigen untuk matriks berukuran *M*x*M* dan kemudian menambil kombinasi linear yang sesuai dari citra wajah Φ_i . Jika vektor eigen adalah v_i dari A^TA sehingga didapatkan persamaan 2.42.

$$A^T A v_i = \mu_i v_i \tag{2.42}$$

Jika persamaan 2.42 dikalikan dengan A dari sebelah kiri, maka akan didapatkan persamaan 2.43.

$$AA^T A v_i = \mu_i A v_i \tag{2.43}$$

Dengan demikian dapat disimpulkan bahwa Av_i adalah vektor eigen dari $C = AA^T$.Berdasarkan analisis tersebut, dapat disusun matriks *L* berukuran *M*x*M*. Matriks L disusun dengan menggunakan persamaan 2.44. Matriks L merupakan matriks yang disusun dari hasil kali transpose matriks A dengan matriks A.

$$L = A^T A \tag{2.44}$$

Selanjutnya, dilakukan perhitungan untuk menemukan vektor eigen $v_i$sebanyak *M* dari *L*. Vektor tersebut merupakan kombinasi linear dari sekumpulan pelatihan gambar wajah untuk membentuk *eigenface* u_i yang dapat diperoleh menggunakan persamaan 2.45.

$$u_i = \sum_{k=1}^{M} v_{ik}\phi_k \qquad i = 1, \dots, M \tag{2.45}$$

Dengan analisis ini, perhitungan sangat tereduksi, dari derajat sejumlah piksel dalam citra (N^2) menjadi sejumlah wajah pelatihan (*M*). dalam prakteknya, wajah pelatihan akan relatif lebih kecil, dan perhitungan menjadi lebih mudah ditangani. Nilai eigen yang berasosiasi memungkinkan untuk mengurutkan vektor eigen berdasarkan kegunaannya dalam mengkarakterisasi variasi antar citra (Turk, 1991).

2.9.2 Fase Klasifikasi

Gambar *eigenface* dihitung dari vektor eigen dari L, yang berfungsi untuk mendeskripsikan gambar wajah. Sebuah gambar wajah baru (Γ) ditransformasikan ke dalam komponen *eigenface* (diproyeksikan ke dalam ruang wajah) dengan operasi 2.46.

$$\omega_k = u_k^T(\Gamma - \Psi) \tag{2.46}$$

Untuk *k=1…M'*.

Bobot dari sebuah vektor $\Omega^T = [\omega_1, \omega_2, \dots, \omega_{M'}]$ mendeskripsikan kontribusi dari setiap *eigenface* dalam merepresentasikan gambar wajah masukan, dengan memperlakukan *eigenface* sebagai sebuah kumpulan basis untuk gambar wajah. Vektor Ω kemudian dapat digunakan dalam sebuah algoritma pengenalan pola standar untuk menemukan pendeskripsian wajah paling sesuai di antara kelas wajah yang telah didefinisikan. Metode yang paling sederhana untuk menentukan kelas wajah yang paling tepat mendeskripsikan gambar wajah masukan adalah untuk menemukan sebuah kelas wajah *k* yang meminimalkan jarak Euclidean dari persamaan 2.47.

$$\epsilon_k^2 = \|\Omega - \Omega_k\|^2 \tag{2.47}$$

Ω_k adalah sebuah vektor yang mendeskripsikan kelas wajah ke-*k*. kelas wajah Ω_i dihitung dengan merata-rata hasil dari representasi *eigenface* dari sejumlah gambar wajah dari tiap individu. Sebuah wajah

diklasifikasikan sebagai bagian dari kelas ke-*k* saat ϵ_k minimal memiliki nilai lebih kecil dari threshold θ_ϵ. Sebaliknya, wajah dikategorikan sebagai "tidak dikenali".

Karena membuat vektor bobot berarti sama dengan memproyeksikan gambar wajah asli ke dalam ruang wajah berdimensi rendah, banyak gambar (kebanyakan dari gambar-gambar tersebut tidak menyerupai wajah) akan diproyeksikan ke dalam sebuah vektor pola. Hal ini bukan merupakan sebuah masalah untuk sistem, bagaimanapun karena jarak ϵ antara gambar dan ruang wajah adalah merupakan jarak kuadrat antara selisih gambar masukan dengan rata-rata, $\Phi = \Gamma - \Psi$ dan $\Phi_f = \sum_{i=1}^{M'} \omega_i u_i$ adalah proyeksinya ke dalam ruang wajah. Jarak ϵ dapat ditemukan dengan menggunakan persamaan 2.48.

$$\epsilon^2 = \left\| \Phi - \Phi_f \right\|^2 \qquad (2.48)$$

Oleh karena itu terdapat empat kemungkinan untuk sebuah gambar masukan dan vektor polanya, yaitu;

(1) Dekat dengan ruang wajah dan dekat dengan sebuah kelas wajah,
(2) dekat dengan ruang wajah tetapi tidak dekat dengan kelas wajah yang diketahui,
(3) jauh dari ruang wajah dan dekat dengan sebuah kelas wajah, dan,
(4) jauh dari ruang wajah dan kelas wajah yang diketahui.

Pada kasus pertama, sebuah individu dikenali dan diidentifikasi. Pada kasus kedua, sebuah individu tidak diketahui muncul. Kedua kasus terakhir mengindikasikan bahwa wajah bukan merupakan gambar wajah. Kasus ketiga biasanya menunjukkan sebuah kesalahan positif pada banyak sistem pengenalan kebanyakan; pada sistem ini, pengenalan yang salah dapat dideteksi karena terdapatnya jarak signifikan antara gambar dan *subruang* dari gambar wajah yang ditentukan (Turk, 1991).

Secara garis besar, prosedur pengenalan dengan menggunakan *eigenface* adalah sebagai berikut:

1. Mengumpulkan sekumpulan gambar wajah dari individu yang diketahui. Kumpulan ini harus mencakup gambar-gambar setiap orang dengan beberapa variasi ekspresi dan pencahayaan. Dimisalkan jumlah wajah sebanyak *N*.
2. Menghitung matriks *L* berukuran *NxN*, menemukan vektor eigen dan nilai eigen dan memilih M' vektor eigen yang memiliki nilai eigen paling besar.

3. Menggabungkan *training set* ternormalisasi berdasarkan persamaan 2.45 untuk menghasilkan *eigenface* u_k sebanyak *M'*
4. Untuk setiap individu yang diketahui, hitung kelas vektor Ω_k dengan merata-rata vektor pola vektor eigen Ω (dari persamaan 2.47) dihitung dari gambar asli tiap individu. Memilih *threshold* θ_ϵ yang menggambarkan jarak maksimum yang diperbolehkan dari semua kelas wajah dan *threshold* θ_ϵ yang mendefinisikan jarak maksimum yang diperbolehkan dari ruang wajah.
5. Untuk setiap wajah baru yang diidentifikasi, dihitung pola vektor Ω, jarak ϵ_i untuk tiap kelas yang diketahui, dan jarak ϵ untuk ruang wajah. Jika jarak minimal $\epsilon_k < \theta_\epsilon$ dan jarak $\epsilon < \theta_k$, maka wajah masukan diklasifikasikan sebagai individu yang berasosiasi dengan kelas vektor Ω_k. Jika jarak minimal $\epsilon_k > \theta_\epsilon$ tetapi jarak $\epsilon < \theta_k$, maka gambar mungkin dikategorikan sebagai "tidak dikenali' dan mungkin dapat digunakan untuk memulai sebuah kelas wajah baru.
   Jika gambar baru diklasifikasikan sebagai individu yang diketahui, maka gambar dapat ditambahkan ke dalam kumpulan asli citra wajah yang serupa, dan *eigenface* dapat dihitung kembali (langkah 1-4).

2.10 Penentuan Threshold (θ)

Pada pendekatan *eigenface*, penentuan *threshold* merupakan faktor penting yang mempengaruhi performa pengenalan wajah. Berdasarkan penelitian yang dilakukan Gupta (2010), *threshold* yang optimal adalah sebesar 0.8 kali nilai maksimum dari jarak-jarak euclidean minimal.

2.11 Precision dan Recall

Precision dan *recall* merupakan alat ukur untuk mengetahui keefektifan sebuah *information retrieval*. *Information retrieval* merupakan sebuah usaha untuk menemukan sebuah dokumen dari lingkungan tak terstruktur yang memenuhi informasi yang dibutuhkan dalam sebuah kumpulan berjumlah banyak (biasanya tersimpan dalam komputer) (Manning, 2008).

Pengenalan wajah bertujuan untuk menemukan wajah yang sesuai dari sekumpulan data yang ada, oleh karena itu pengenalan wajah dapat dianggap sebagai sebuah *information retrieval*. *Precision* dapat dianggap sebagai ukuran untuk ketepatan, sedangkan *recall* adalah ukuran untuk kelengkapan.

Precision (P) menyatakan sejumlah wajah relevan yang didapatkan. *Precision* dapat dinyatakan dalam persamaan 2.49.

$$Precision = \frac{\#(item\ relevan\ yang\ didapatkan)}{\#(item\ yang\ didapatkan)} = P(relevan|didapatkan) \quad (2.49)$$

Recall menyatakan bagian wajah relevan yang diambil. *Recall* dinyatakan dalam persamaan 2.50.

$$Recall = \frac{\#(item\ yang\ didapatkan)}{\#(item\ yang\ relevan)} = P(didapatkan|relevan) \quad (2.50)$$

Tabel 2.1 merupakan tabel yang menyatakan hubungan antara wajah yang relevan dan wajah yang terambil. Berdasarkan hubungan dalam Tabel 2.1 tersebut, maka *precision* dan *recall* dapat dinyatakan dalam persamaan 2.51 dan 2.52.

Tabel 2.1 Tabel hubungan wajah yang relevan dan wajah yang terambil

	Relevan	Tidak relevan
Terambil	*True positive (tp)*	*False positive (fp)*
Tidak terambil	*False negative (fn)*	*True negative (tn)*

$$P = \frac{tp}{tp + fp} \quad (2.51)$$

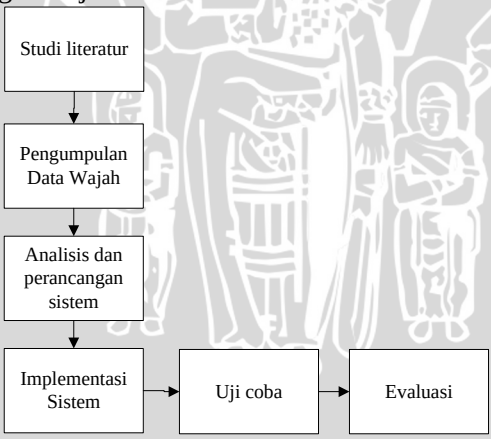
$$R = \frac{tp}{tp + fn} \quad (2.52)$$

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Dalam bab ini akan dijelaskan mengenai analisis sistem dan perancangan sistem untuk mengenali sebuah citra wajah dengan menggunakan *Principal Component Analysis*. Tahapan yang dilakukan dalam penelitian pengenalan wajah adalah sebagai berikut:

1. Mempelajari pustaka yang terkait dengan pengenalan wajah dan *Principal Component Analysis*.
2. Mengumpulkan data wajah yang akan digunakan sebagai data latihan dan data masukan.
3. Menganalisis dan merancang proses pelatihan dan proses pengenalan wajah dengan menggunakan PCA.
4. Membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan (implementasi sistem).
5. Melakukan uji coba sistem dengan memasukkan citra wajah yang berbeda dengan data latihan.
6. Mengevaluasi hasil pengujian.

Tahapan penelitian tersebut dapat digambarkan dalam bentuk diagram alir yang ditunjukkan oleh Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.1 Deskripsi Sistem

Secara umum, sistem yang akan dibangun adalah sebuah perangkat lunak yang digunakan untuk mengenali wajah dengan menggunakan PCA sebagai media representasi wajah. Sistem terdiri dari dua proses utama, yaitu proses pelatihan dan proses pengenalan. Proses pelatihan bertujuan untuk menemukan wajah-wajah eigen dari data latihan. Wajah-wajah eigen tersebut kemudian digunakan dalam proses pengenalan. Pada proses pengenalan, input wajah akan dikenali berdasarkan wajah-wajah eigen tersebut.

3.2 Data yang Digunakan

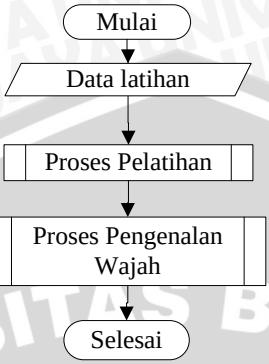
Data yang akan digunakan dalam proses pengujian adan pembelajaran diperoleh dari data citra wajah *faces94* yang diperoleh dari http://www.essex.ac.uk/mv/allfaces/faces94.html

Data tersebut berisi data citra wajah berwarna yang terdiri dari 153 individu dengan resolusi 90 x 100 pixel. Setiap individu memiliki 20 citra dengan posisi wajah serta pencahayaan yang hampir sama namun dengan ekspresi yang sedikit berbeda.

3.3 Perancangan Proses

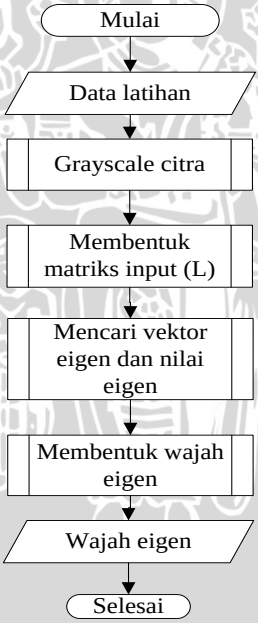
Gambar 3.1 menggambarkan jalannya proses pengenalan wajah secara keseluruhan. Sebelum dapat mengenali wajah, sistem terlebih dahulu melakukan pelatihan dengan menggunakan data latihan. Data latihan diperoleh dari wajah-wajah yang dideskripsikan sebagai wajah yang dikenali. Dari proses pelatihan ini kemudian dihasilkan data latihan berupa wajah eigen yang akan digunakan dalam proses pengenalan.

Pada proses pengenalan, wajah baru akan diproyeksikan ke dalam setiap wajah eigen untuk kemudian dicari kedekatannya dengan wajah eigen - wajah eigen tersebut. Wajah dapat diklasifikasikan sebagai “dikenal” atau “tidak dikenal” berdasarkan kedekatan wajah baru dengan wajah eigen-wajah eigen tersebut.



Gambar 3.2 Proses pengenalan wajah dengan menggunakan PCA

3.3.1 Proses Pelatihan



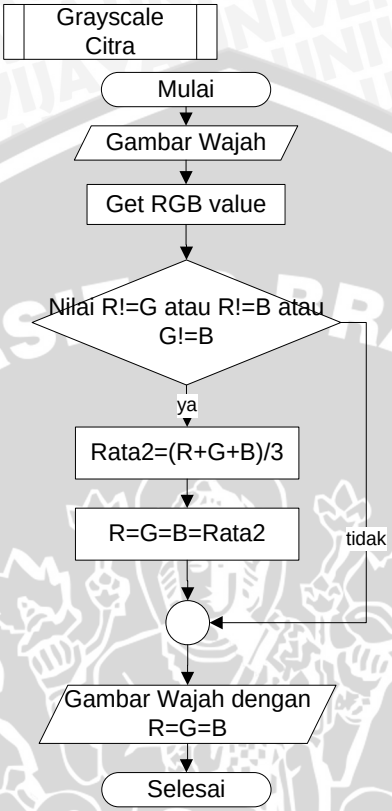
Gambar 3.3 Proses Pelatihan

Proses pelatihan bertujuan untuk menyediakan data awal sebagai pembanding untuk data yang akan dimasukkan. Proses pelatihan pada sistem ini bertujuan untuk menghasilkan representasi wajah yang berupa wajah eigen. Wajah eigen-wajah eigen yang dihasilkan pada proses pelatihan ini nantinya digunakan sebagai pembanding pada proses pengenalan. Secara garis besar, proses pelatihan ditunjukkan pada Gambar 3.3.

Proses di awali dengan menjadikan citra-citra yang menjadi data masukan menjadi citra keabuan (*grayscale*). Setelah itu, seluruh citra yang telah di-*grayscale* digabung menjadi sebuah matriks untuk membentuk matriks input (L). Matriks input tersebut kemudian dicari nilai eigen dan vektor eigennya. Vektor eigen yang memiliki nilai eigen terbesar kemudian disusun kembali sehingga terbentuklah wajah eigen.

3.3.1.1 Grayscale Citra

Pada proses pelatihan digunakan data latihan yang berupa citra wajah. Setiap gambar yang akan digunakan terlebih dahulu akan dikonversi ke dalam citra *grayscale*. Hal ini bertujuan untuk memudahkan perhitungan pada proses selanjutnya. Untuk mengkonversi citra berwarna ke dalam citra *grayscale*, dicari rata-rata dari komponen warna RGB gambar wajah pada setiap pixel gambar wajah. Proses tersebut digambarkan pada Gambar 3.4.



Gambar 3.4 Proses Grayscale Citra

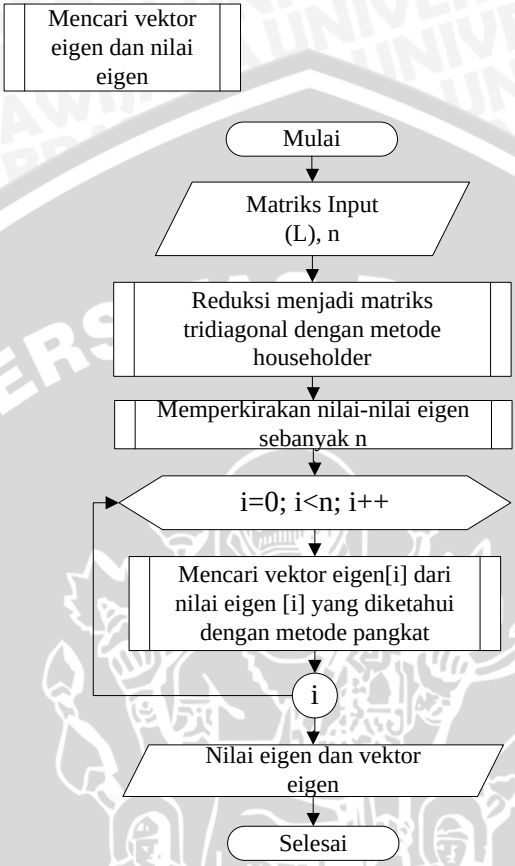
3.3.1.2 Pembentukan Matriks Input

Gambar 3.5 menggambarkan proses pembentukan matriks input. Untuk membentuk matriks input, setiap nilai *grayscale* pada gambar wajah, disusun menjadi sebuah vektor Γ_n. Vektor-vektor Γ tersebut kemudian dicari rata-ratanya. Untuk membentuk matriks A yang merupakan sekumpulan selisih gambar dengan rata-rata, maka setiap gambar wajah Γ dikurangi dengan rata-rata. Matriks input, dalam hal ini disimbolkan dengan L, dicari dengan mengalikan matriks A^T dengan A. setelah matriks input terbentuk, nilai eigen dan vektor eigen dari matriks input tersebut yang kemudian digunakan untuk membentuk wajah eigen.

Gambar 3.5 Proses Pembentukan Matriks Input

3.3.1.3 Mencari Vektor Eigen dan Nilai Eigen

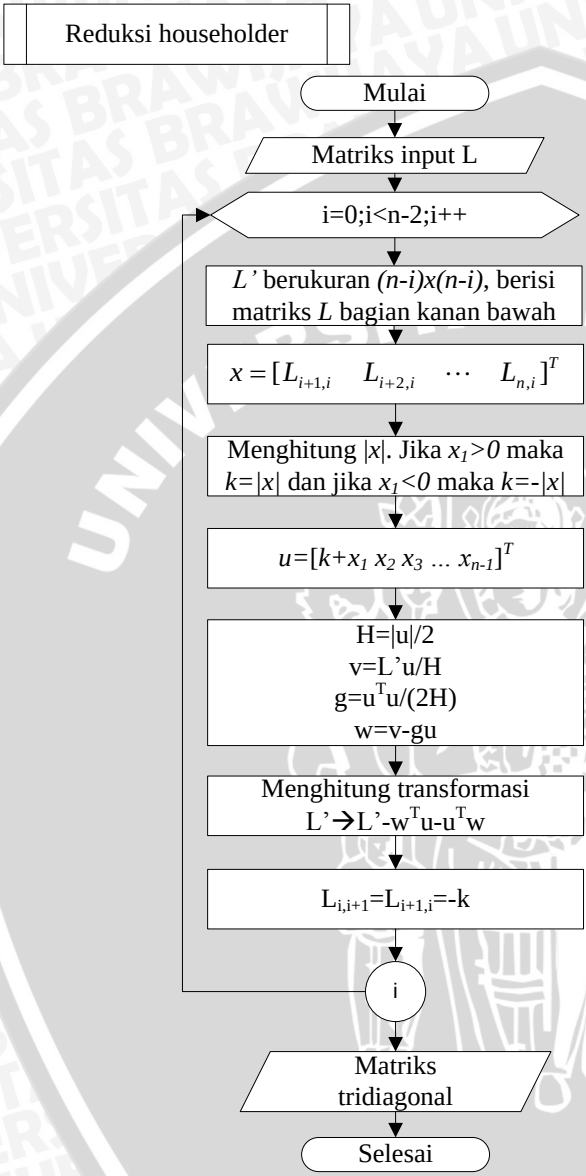
Gambar 3.6 menunjukkan proses untuk menemukan nilai eigen dan vektor eigen. Sebelum dicari nilai eigen dan vektor eigennya, matriks input L terlebih dahulu direduksi menjadi bentuk tridiagonal untuk mengurangi kompleksitas perhitungan. Langkah selanjutnya adalah melakukan pencarian tebakan nilai eigen sebanyak *n*. Setelah tebakan nilai-nilai eigen ditemukan, maka vektor eigen yang bersesuaian dihitung pada langkah berikutnya dengan menggunakan metode pangkat. Subproses reduksi matriks dengan metode Householder ditunjukkan pada Gambar 3.7.



Gambar 3.6 Proses Pencarian Nilai Eigen dan Vektor Eigen

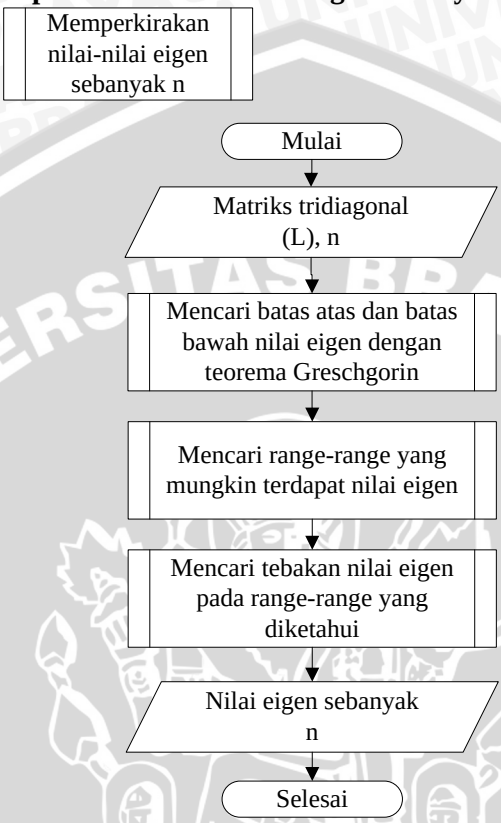
3.3.1.3.1 Reduksi Householder

Gambar 3.7 menunjukkan reduksi matriks simetris *L* menjadi bentuk tridiagonal. Proses reduksi matriks menjadi bentuk tridiagonal dilakukan dengan mentransformasi submatriks *L'*. Proses transformasi akan dilakukan sebanyak *n-2* kali dimana *n* merupakan dimensi baris matriks *L*. Proses reduksi matriks menjadi bentuk tridiagonal mengikuti langkah-langkah pada bagian 2.2.2.2.



Gambar 3.7 Proses Reduksi Householder

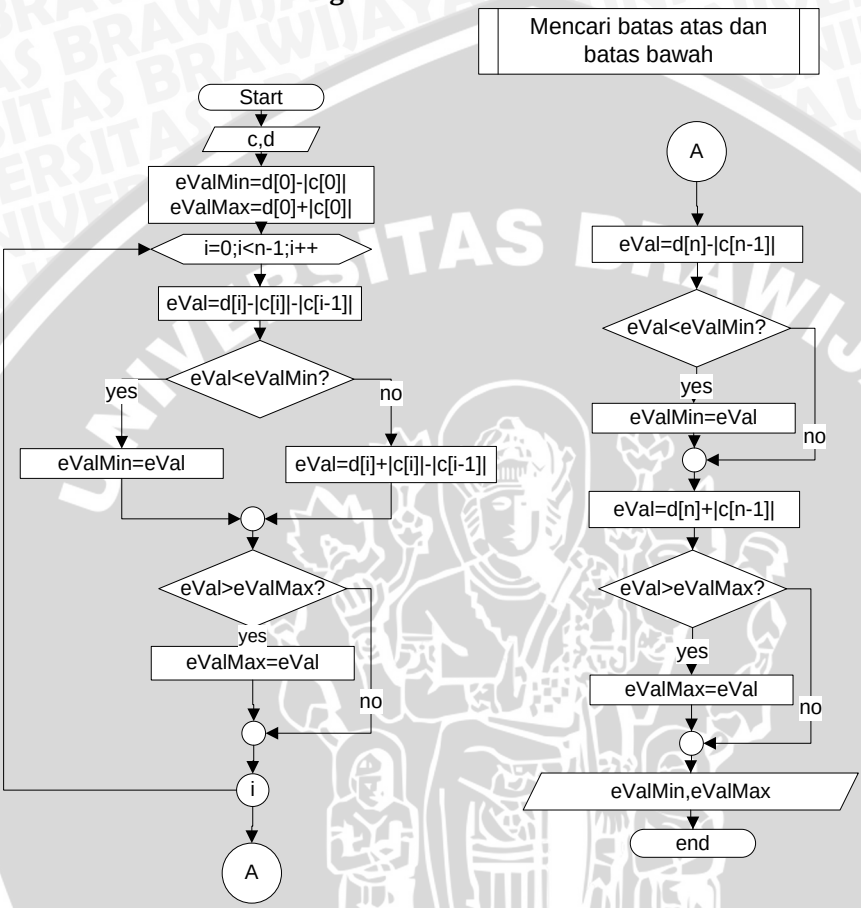
3.3.1.3.2 Memperkirakan nilai-nilai eigen sebanyak n



Gambar 3.8 Proses Pencarian Nilai Eigen

Gambar 3.8 menunjukkan proses pencarian nilai eigen untuk matriks tridiagonal *L*. Untuk *n* nilai eigen yang akan dicari, terlebih dahulu dicari nilai minimal dan maksimalnya. Setelah ditemukan nilai minimal dan maksimalnya, range atau daerah terdapatnya nilai eigen, dicari untuk setiap nilai eigen yang ingin diketahui.

3.3.1.3.3 Mencari batas atas dan batas bawah nilai eigen dengan metode Greschgorin



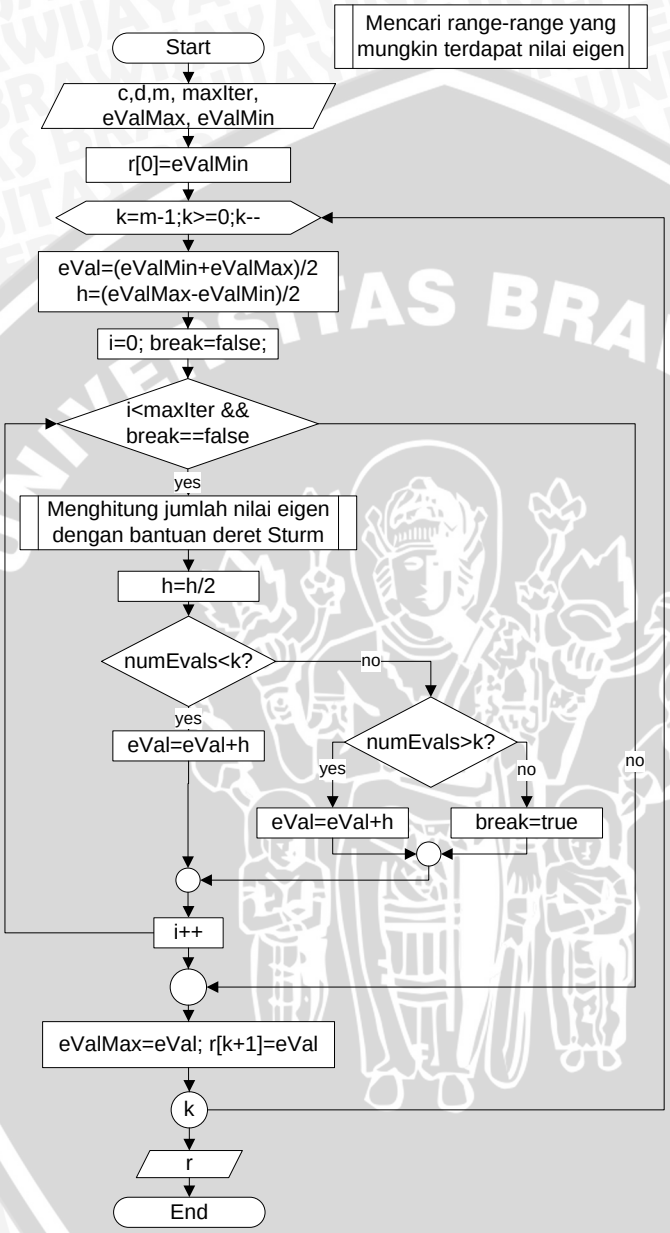
Gambar 3.9 Mencari batas atas dan batas bawah nilai eigen dengan metode Greschgorin

Batas maksimal (*eValMax*) dan batas minimal (*eValMin*) nilai eigen ditentukan dengan metode Greschgorin. Proses metode Greschgorin ditunjukkan pada Gambar 3.9. masukan dari proses ini adalah diagonal matriks input *L* (*d*) dan updiagonal matriks input *L* (*c*). Batas-batas nilai eigen yaitu *eValMin* dan *eValMax* nantinya digunakan untuk pencarian range-range yang mungkin terdapat nilai eigen.

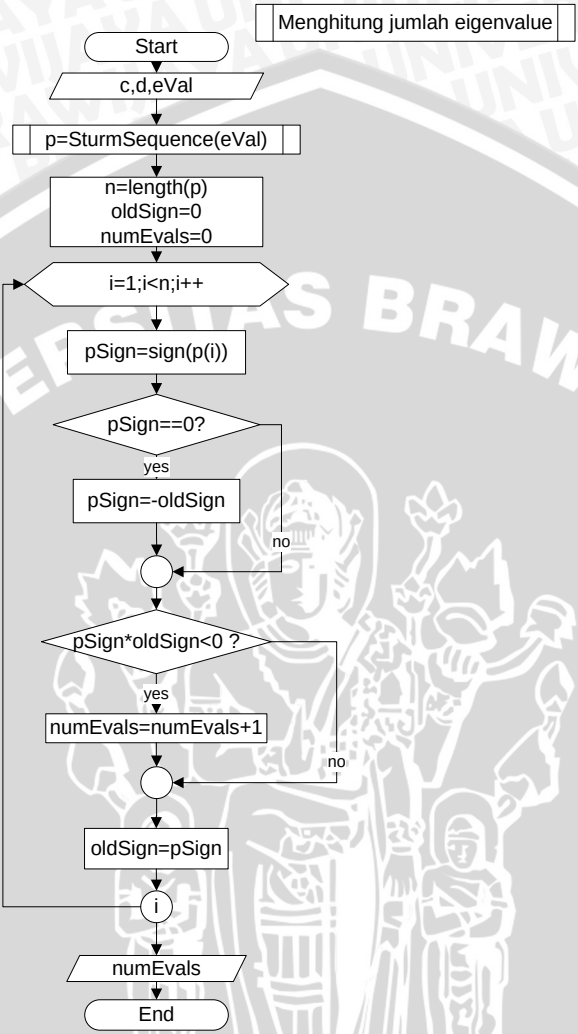
3.3.1.3.4 Mencari range-range yang mungkin terdapat nilai eigen

Daerah-daerah terdapatnya nilai eigen perlu diketahui untuk memudahkan pencarian tebakan awal nilai eigen secara iteratif. Jika daerah-daerah terdapatnya nilai eigen tidak diketahui terlebih dahulu, maka proses penemuan nilai eigen secara iteratif akan membutuhkan perulangan yang sangat banyak. Proses pencarian range-range terdapatnya nilai eigen dibantu oleh deret Sturm. Deret Sturm untuk sebuah nilai eigen, *eVal*, ditemukan untuk kemudian dihitung banyaknya perubahan tanda yang ada dalam deret tersebut. Langkah-langkah perhitungan jumlah nilai eigen ditujukkan pada Gambar 3.11, sedangkan langkah-langkah deret Sturm ditunjukkan pada Gambar 3.12.

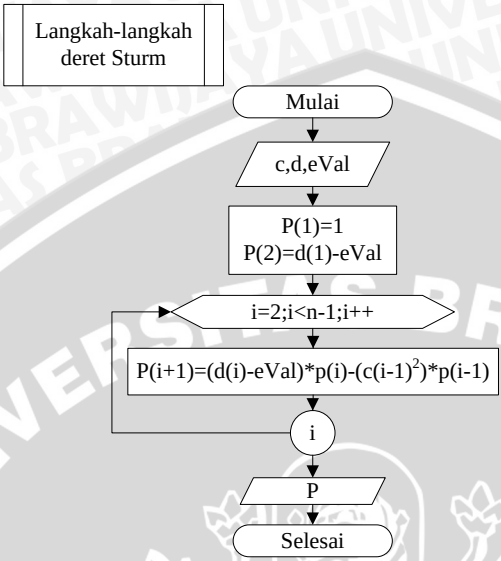
Proses pencarian range-range nilai eigen dimulai dengan menginisialisasi array *r* indeks ke 0 dengan nilai *eValMin*. Array *r* merupakan sebuah array yang menyimpan nilai-nilai yang menjadi batas range-range nilai eigen. Misalnya, range pertama nilai eigen memiliki rentang nilai antara *r*[0] dan *r*[1], range kedua nilai eigen memiliki rentang nilai antara *r*[1] dan *r*[2], dan demikian seterusnya. Nilai yang sesuai untuk tiap batas ditentukan dengan bantuan deret Sturm. Untuk range pertama, yaitu antara *r*[0] dan *r*[1], harus memiliki jumlah nilai eigen sebanyak 1. Dengan kata lain, harus ditemukan sebuah nilai *eVal* yang memiliki perubahan tanda sebanyak 1 kali dalam deret Sturmnya. Dan untuk batas *r*[2], maka harus ditemukan sebuah nilai *eVal* yang memiliki perubahan tanda sebanyak 2 kali dalam deret Sturmnya, dan demikian seterusnya. Hasil akhir dari proses ini adalah sebuah array *r* yang memuat batas-batas daerah terdapatnya nilai eigen. Proses ini digambarkan pada Gambar 3.10.



Gambar 3.10 Proses pencarian range-range terdapatnya nilai eigen

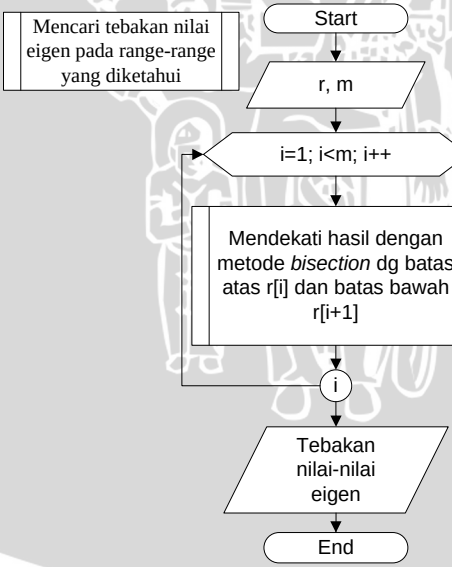


Gambar 3.11 Flowchart penghitungan jumlah nilai eigen



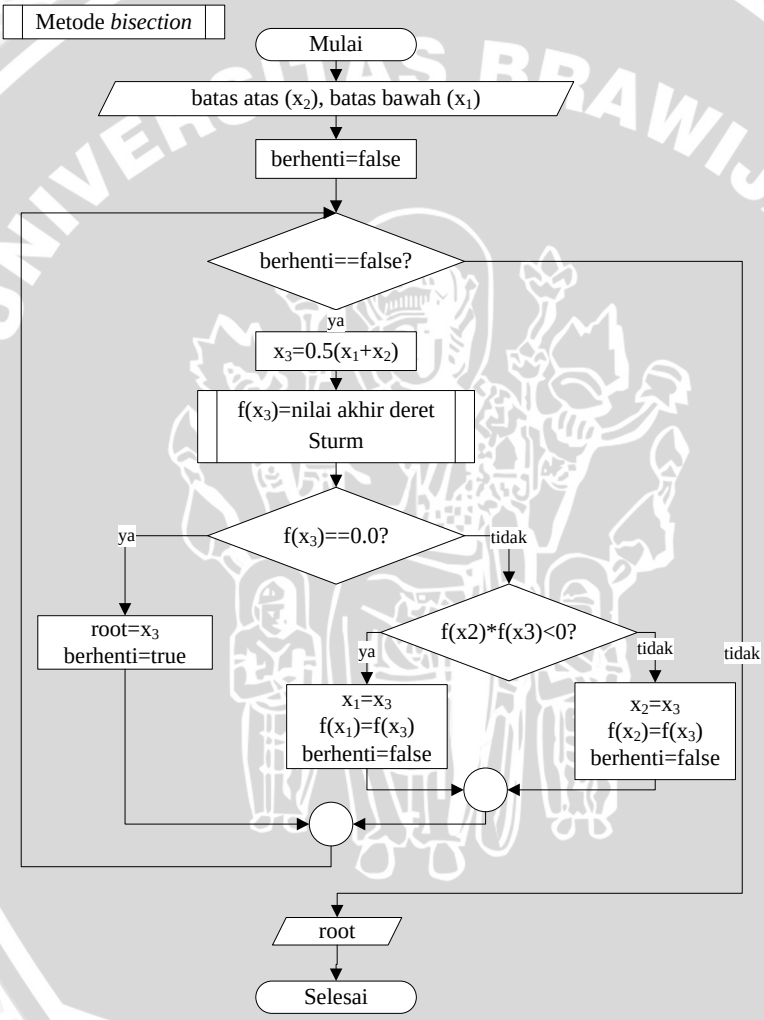
Gambar 3.12 Langkah-langkah Deret Sturm

3.3.1.3.5 Mencari tebakan nilai eigen pada range-range yang diketahui



Gambar 3.13 Proses pencarian tebakan awal nilai eigen

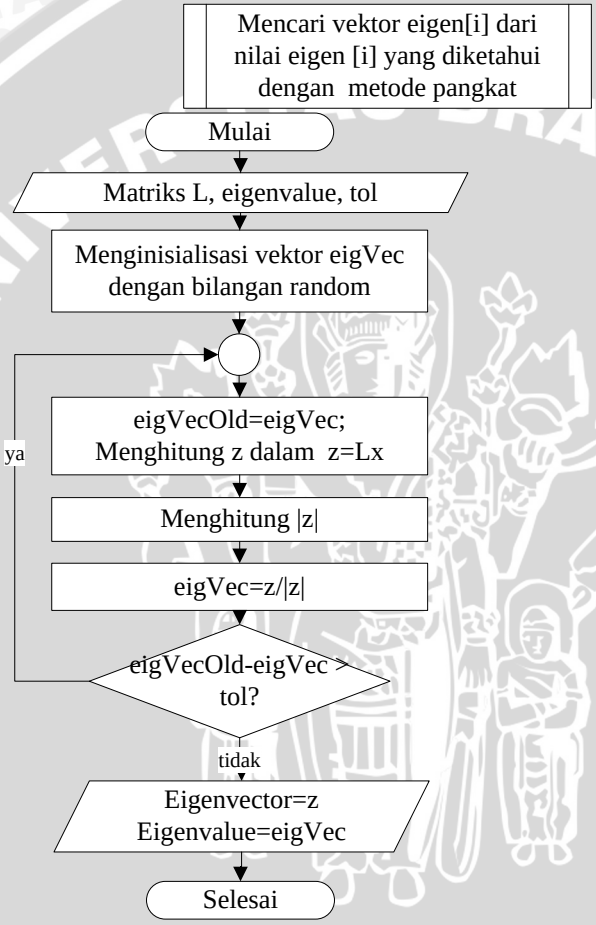
Setelah batas-batas range untuk setiap nilai eigen diketahui, maka langkah selanjutnya adalah mencari nilai eigen di dalam range tersebut dengan metode iteratif *bisection*. Langkah-langkah metode *bisection* digambarkan dalam Gambar 3.14. Nilai eigen dianggap ditemukan jika nilai akhir dari deret Sturm adalah 0. Langkah-langkah menebak nilai eigen digambarkan dalam Gambar 3.13.



Gambar 3.14 Metode Bisection

3.3.1.3.6 Mencari Vektor Eigen dari Nilai-Nilai Eigen yang Diketahui

Gambar 3.15 menunjukkan proses metode pangkat yang bertujuan untuk mencari vektor eigen berdasarkan nilai eigen yang telah diperkirakan sebelumnya. Selain dapat menemukan vektor eigen, metode pangkat membantu menemukan nilai eigen yang lebih tepat.

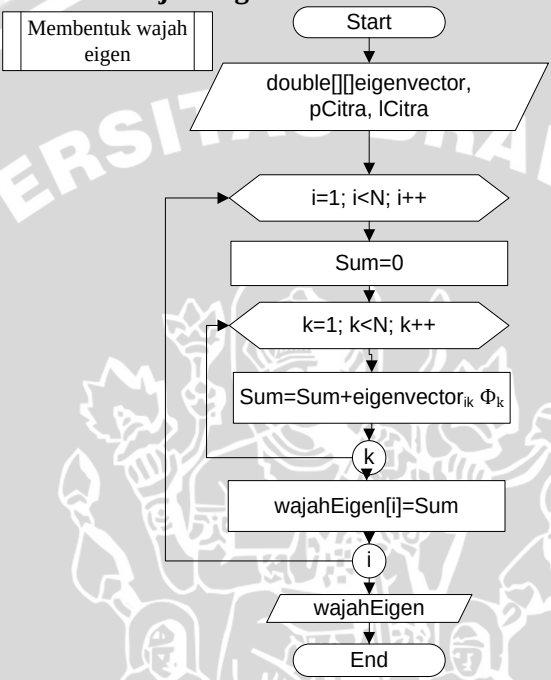


Gambar 3.15 Langkah-langkah Metode Pangkat

Proses ini memerlukan masukan berupa matriks input L, *eigenvalue*, dan *tol* (menyatakan batas toleransi yang diinginkan). Proses ini dimulai dengan menginisialisasi vektor *eigVec* dengan bilangan random. Nilai *z*

dicari sebagai pendekatan terhadap vektor eigen. Proses akan berhenti jika nilai selisih *eigVec* lebih kecil dari nilai toleransi, *tol*, yang telah ditentukan. Dari proses ini akan didapatkan nilai eigen *eigVec* dan vektor eigen *z*.

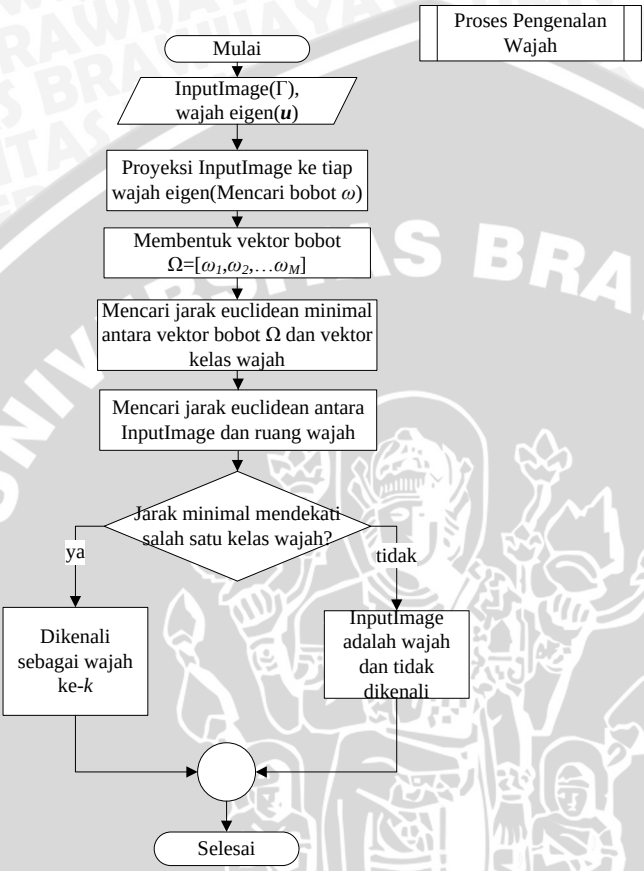
3.3.1.4 Membentuk Wajah Eigen



Gambar 3.16 Proses pembentukan wajah Eigen

Gambar 3.16 menggambarkan aliran proses pembentukan wajah eigen berdasarkan persamaan 2.41. Setiap wajah eigen merupakan jumlah dari hasil perikalian vektor eigen dengan vektor selisih Φ.

3.3.2 Proses Pengenalan



Gambar 3.17 Proses Pengenalan Wajah

Gambar 3.17 menunjukkan proses pengenalan wajah. Wajah eigen yang dihasilkan pada proses pelatihan, digunakan untuk memproyeksikan *ImageInput*. Setiap proyeksi *ImageInput* dengan wajah eigen, akan dihitung jaraknya dengan ruang wajah dan setiap kelas wajah yang ada. Jarak-jarak tersebut kemudian dibandingkan untuk memperoleh kesimpulan apakah *InputImage* tersebut dikenali atau tidak.

3.4 Contoh Perhitungan untuk Mencari Nilai Eigen Terkecil

Dalam perhitungan manual ini, dimisalkan matriks input berupa matriks A dengan kovarian $\begin{bmatrix} 7 & 2 & 3 & 6 \\ 2 & 1 & 2 & 6 \\ 3 & 2 & 1 & 7 \\ 6 & 6 & 7 & 4 \end{bmatrix}$. Matriks A merupakan sebuah image dengan derajat keabuan 8.

3.4.1 Reduksi Householder

Untuk menyederhanakan perhitungan, matriks input A direduksi terlebih dahulu dengan menggunakan metode Householder. Perhitungan akan berlangsung dalam $n-2$ iterasi. Karena dimensi matriks A adalah 4, maka akan berulang sebanyak 2 iterasi.

a) Reduksi baris dan kolom pertama (iterasi pertama)

- Pengambilan submatriks kanan bawah sebesar $(n-1)x(n-1)$

$$A' = \begin{bmatrix} 1 & 2 & 6 \\ 2 & 1 & 7 \\ 6 & 7 & 4 \end{bmatrix}$$

- Mengambil nilai dari kolom sebelah kanan A' sebagai x

$$x = \begin{bmatrix} 2 \\ 3 \\ 6 \end{bmatrix}$$

- Menghitung $|x|$ dan menentukan tanda untuk k

$$k = |x| = 7$$

- Menghitung u dan H

$$u = \begin{bmatrix} 2+7 \\ 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 9 \\ 3 \\ 6 \end{bmatrix}$$

$$uu^T = \begin{bmatrix} 81 & 27 & 54 \\ 27 & 9 & 18 \\ 54 & 18 & 36 \end{bmatrix}$$

$$H = \frac{1}{2}\, u^T u = 63$$

- Menghitung transformasi untuk perulangan pertama

$$Q = I - \frac{uu^T}{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1.2857 & 0.4286 & 0.8571 \\ 0.4286 & 0.1429 & 0.2851 \\ 0.8571 & 0.2857 & 0.5714 \end{bmatrix}$$

$$= \begin{bmatrix} -0.2857 & -0.4286 & -0.8571 \\ -0.4286 & 0.8571 & -0.2857 \\ -0.8571 & -0.2857 & 0.4286 \end{bmatrix}$$

$$QA'Q = \begin{bmatrix} 11.7755 & -0.9796 & 3.8980 \\ -0.9796 & -2.1837 & 1.9184 \\ 3.8980 & 1.9184 & -3.5918 \end{bmatrix}$$

$$A \leftarrow \begin{bmatrix} A_{11} & (Q_x)^T \\ Q_x & QA'Q \end{bmatrix} = \begin{bmatrix} 7 & -7 & 0 & 0 \\ -7 & 11.7755 & -0.9796 & 3.8980 \\ 0 & -0.9796 & -2.1837 & 1.9184 \\ 0 & 3.8980 & 1.9184 & -3.5918 \end{bmatrix}$$

Pada langkah akhir digunakan persamaan 2.13, di mana $\mathbf{Qx} = \mathbf{x} - \mathbf{u} = -k\mathbf{e}_1 = [-k \quad 0 \quad 0 \quad ... \quad 0]^T$

b) Reduksi baris dan kolom kedua

- Pengambilan submatriks A dengan dimensi $(n-2)x(n-2)$

$$A' = \begin{bmatrix} -2.1837 & 1.9184 \\ 1.9184 & -3.5918 \end{bmatrix}$$

- Mengambil nilai dari kolom sebelah kanan A' sebagai x

$$x = \begin{bmatrix} -0.9796 \\ 3.8980 \end{bmatrix}$$

- Menghitung $|x|$ dan menentukan tanda untuk k. Tanda k negatif karena x_1 bertanda negatif.

$$k = -|x| = -4.0192$$

- Menghitung u dan H

$$u = \begin{bmatrix} -4.9988 \\ 3.8980 \end{bmatrix}$$

$$uu^T = \begin{bmatrix} 24.9876 & -19.4849 \\ -19.4849 & 15.1941 \end{bmatrix}$$

$$H = \frac{1}{2} u^T u = 20.0908$$

- Menghitung transformasi untuk perulangan kedua

$$Q = I - \frac{uu^T}{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1.2437 & -0.9698 \\ -0.9698 & 0.7536 \end{bmatrix} = \begin{bmatrix} -0.2437 & 0.9698 \\ 0.9698 & 0.2437 \end{bmatrix}$$

$$QA'Q = \begin{bmatrix} 11.7755 & -0.9796 & 3.8980 \\ -0.9796 & -2.1837 & 1.9184 \\ 3.8980 & 1.9184 & -3.5918 \end{bmatrix}$$

$$A \leftarrow \begin{bmatrix} A_{11} & A_{12} & 0^T \\ A_{21} & A_{22} & (Q_x)^T \\ 0 & Q_x & QA'Q \end{bmatrix} = \begin{bmatrix} 7 & -7 & 0 & 0 \\ -7 & 11.7755 & 4.0192 & 0 \\ 0 & 4.0192 & -4.4151 & 1.3576 \\ 0 & 0 & 1.3576 & -1.3604 \end{bmatrix}$$

c) Menghitung matriks transformasi

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Iterasi pertama

$$u = \begin{bmatrix} 9 \\ 3 \\ 6 \end{bmatrix}$$

$$H = \frac{1}{2}\, u^T u = 63$$

$$v = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1429 \\ 0.0476 \\ 0.0952 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.1429 \\ 0.0476 \\ 0.0952 \end{bmatrix}$$

$$vu^T = \begin{bmatrix} 0 \\ 0.1429 \\ 0.0476 \\ 0.0952 \end{bmatrix} [9 \quad 3 \quad 6] = \begin{bmatrix} 0 & 0 & 0 \\ 1.2857 & 0.4286 & 0.8571 \\ 0.4286 & 0.1429 & 0.2857 \\ 0.8571 & 0.2857 & 0.5714 \end{bmatrix}$$

$$P \leftarrow P - vu^T = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & -0.2857 & -0.4286 & 0.8571 \\ 0 & -0.4286 & 0.8571 & -0.2857 \\ 0 & -0.8571 & -0.2857 & 0.4286 \end{bmatrix}$$

- Iterasi ke-2

$$u = \begin{bmatrix} -4.9988 \\ 3.8980 \end{bmatrix}$$

$$H = \frac{1}{2}\, u^T u = 20.0908$$

$$v = \begin{bmatrix} 0 & 0 \\ -0.4286 & 0.8571 \\ 0.8571 & -0.2857 \\ -0.2857 & 0.4286 \end{bmatrix} \begin{bmatrix} -0.2488 \\ 0.1940 \end{bmatrix} = \begin{bmatrix} 0 \\ -0.0597 \\ -0.2687 \\ 0.1524 \end{bmatrix}$$

$$vu^T = \begin{bmatrix} 0 & 0 \\ 0.2983 & -0.2326 \\ 1.3432 & -1.0474 \\ -0.7710 & 0.6012 \end{bmatrix}$$

$$P \leftarrow P - vu^T = \begin{bmatrix} 1.0000 & 0 & 0 & 0 \\ 0 & -0.2857 & -0.7268 & -0.6246 \\ 0 & -0.4286 & -0.4860 & 0.7617 \\ 0 & -0.8571 & 0.4863 & -0.1727 \end{bmatrix}$$

3.4.2 Menyelesaikan Determinan Matriks Tridiagonal A

Penyelesaian determinan matriks tridiagonal A menggunakan metode pangkat dengan *eigevalue shifting*. Sebelum penggunaan metode tersebut, nilai tebakan awal untuk nilai eigen akan ditemukan terlebih dahulu.

a) Mencari batas atas dan batas bawah nilai eigen menggunakan pendekatan Greschgorin.

Dengan menggunakan persamaan (2.18) ditemukan dugaan nilai minimal dan maksimal dari nilai eigen adalah sebagai berikut:

$$a_1 = 7 \quad a_2 = 11.7755 \quad a_3 = -4.4151 \quad a_4 = -1.3604$$
$$r_1 = 7 \quad r_2 = 11.0192 \quad r_3 = 5.3768 \quad r_4 = 1.3576$$
$$a_1 - r_1 = 0 \quad a_2 - r_2 = 0.7563 \quad a_3 - r_3 = -9.7919 \quad a_4 - r_4 = -2.7187$$
$$a_1 + r_1 = 14 \quad a_2 + r_2 = 22.7947 \quad a_3 + r_3 = 0.9617 \quad a_4 + r_4 = -0.0025$$
$$evalmin = -9.7919 \quad evalmax = 22.7947$$

Dengan demikian dapat disimpulkan bahwa nilai eigen berkisar antara -9.7919 dan 22.7947.

b) Mencari daerah-daerah terdapatnya nilai eigen.

Setelah kisaran nilai eigen minimal dan maksimal ditemukan, perlu dicari kisaran setiap nilai eigen yang ingin diketahui.

Misalkan nilai eigen yang ingin dicari adalah 1 nilai eigen terkecil. Pencarian kisaran nilai eigen ini dibantu dengan menggunakan deret Sturm.

- Deret Sturm untuk mengecek banyaknya perubahan tanda

$$h = (22.7947 - (-9.7919))/2 = 16.2933$$
$$eval = (22.7947 + (-9.7919))/2 = 6.5014$$
$$P_0 = 1$$
$$P_1(eval) = d_1 - eval = 7 - 6.5014 = 0.4986$$
$$P_2(eval) = (d_2 - eval)P_1 - c_1^2 P_0 = (11.7755 - 6.5014) - (-7)^2 . 1 = -46.3707$$
$$P_3(eval) = (d_3 - eval)P_2 - c_2^2 P_1 = (-4.4151 - 6.5014) - (4.0192)^2\ 0.4986 = 498.1481$$
$$P_4(eval) = (d_4 - eval)P_3 - c_3^2 P_2 = (-1.3604 - 6.5014) - (1.3576)^2\ (-46.3707) = -3830.9$$

Perubahan tanda sebanyak 3, dengan demikian terdapat 3 buah nilai eigen pada interval -9.7919 hingga 22.7947. Karena nilai

eigen yang diinginkan hanya satu, maka nilai eval dikurangi dengan ½ h.

$$h = \frac{16.2933}{2} = 8.1466$$

$$eval = 6.5014 - 8.1466 = -1.6452$$

Dengan menggunakan deret Sturm, dihitung kembali banyaknya perubahan tanda pada deret $P(-1.6452)$

$$P_0 = 1$$
$$P_1(eval) = d_1 - eval = 7 - (-1.6452) = 8.6452$$
$$P_2(eval) = (d_2 - eval)P_1 - c_1^2 P_0$$
$$= (11.7755 - (-1.6452)) — 72.1 = 67.0247$$
$$P_3(eval) = (d_3 - eval)P_2 - c_2^2 P_1$$
$$= \left(-4.4151 - (-1.6452)\right) - (4.0192)^2\ 8.6452$$
$$= -325.3045$$
$$P_4(eval) = (d_4 - eval)P_3 - c_3^2 P_2$$
$$= \left(-1.3604 - (-1.6452)\right) - (1.3576)^2\ (67.0247)$$
$$= -216.1770$$

Pada deret Sturm di atas, terjadi perubahan tanda sebanyak satu, dengan demikian, ma maka pada daerah -9.7919 hingga -1.6452 terdapat 1 buah nilai eigen.

c) Mencari nilai eigen pada daerah-daerah yang telah diperkirakan

Setelah kisaran nilai eigen diketahui, nilai eigen sebenarnya didekati dengan metode *bisection*. Setiap nilai x_i akan dicari nilai $f(x_i)$-nya, di mana $f(x_i)$ merupakan deret terakhir dari deret Sturm. Jika $f(x_i)$ mendekati 0, maka x_i merupakan nilai eigen yang dicari.

Metode *bisection* dengan range -9.7919 hingga -1.6452

i	x_i	$f(x_i)$	Interval
1	-9.7919	11332.6018	-
2	-1.6452	-216.1770	(-9.7919, -1.6452)
3	(-9.7919+(-1.6452))/2=-5.7186	-229.5389	(-9.7919, -5.7186)
4	(-9.7919+(-5.7186))/2=-7.7553	3143,9238	(-7.7553, -5.7186)
5	(-7.7553+(-5.7186))/2=-6.7370	991.5431	(-6.7370, -5.7186)
6	(-6.7370+(-5.7186))/2=-6.2278	280.1776	(-6.2278, -5.7186)
7	(-6.2278+(-5.7186))/2=-5.9732	1.9776	(-5.9732, -5.7186)
8	(-5.9732+(-5.7186))/2=-5.8459	-119.3885	(-5.9732, -5.8459)
9	(-5.9732+(-5.8459))/2=-5.9096	-60.1356	(-5.9732, -5.9096)
10	(-5.9732+(-5.9096))/2=-5.9414	-29.4157	(-5.9732, -5.9414)
11	(-5.9732+(-5.9414))/2=-5.9573	-13.8096	(-5.9732, -5.9573)
12	(-5.9732+(-5.9573))/2=-5.9653	-5.9387	(-5.9732, -5.9653)
13	(-5.9732+(-5.9653))/2=-5.9693	-1.9613	(-5.9732, -5.9693)
14	(-5.9732+(-5.9693))/2=-5.9712	0.0317	(-5.9712, -5.9693)
15	(-5.9712+(-5.9693))/2=-5.9703	-0.9652	(-5.9712, -5.9703)

16	(-5.9712+(-5.9703))/2=-5.9707	-0.4669	(-5.9712, -5.9707)
17	(-5.9712+(-5.9707))/2=-5.9710	-0.2675	(-5.9712, -5.9710)
18	(-5.9712+(-5.9710))/2=-5.9711	-0.1179	(-5.9712, -5.9711)
19	(-5.9712+(-5.9711))/2=-**5.9711**	-0.1179	-

Dengan demikian, nilai eigen yang didapatkan adalah -5.9711

3.4.2.1 Mencari Vektor Eigen Dari Nilai Eigen Yang Diketahui

Pencarian nilai eigen dan vektor eigen pada bagian ini menggunakan metode pangkat dan *eigenvalue shifting*. Nilai eigen awal untuk memulai metode ini didapatkan pada batian 3.6.2, dengan nilai sebesar -5.9711.

- *Eigenvalue shifting*

$$s = -5.9711$$

$$d = d - s = \begin{bmatrix} 12.9711 \\ 17.7466 \\ 1.5560 \\ 4.6107 \end{bmatrix}$$

- Dekomposisi matriks A dengan menggunakan dekomposisi LU

  Secara iteratif, perhitungan dekomposisi matriks A adalah sebagai berikut:

Iterasi ke-1 (k=2)

$$\lambda = \frac{c_2}{d_2} = \frac{-7}{12.9711} = -0.5397$$
$$d_2 = d_2 - \lambda * e_1 = 13.9690$$
$$c_1 = \lambda = -0.5397$$

Iterasi ke-2 (k=3)

$$\lambda = \frac{c_2}{d_2} = \frac{4.0192}{13.9687} = 0.2877$$
$$d_3 = d_3 - \lambda * e_2 = 0.3996$$
$$c_2 = \lambda = 0.2887$$

Iterasi ke-3(k=4)

$$\lambda = \frac{c_{k-1}}{d_{k-1}} = \frac{1.3576}{0.3996} = 3.3974$$
$$d_4 = d_4 - \lambda * e_3 = -0.0015$$
$$c_3 = \lambda = 3.3974$$

Dengan demikian, diperoleh nilai c, d, dan e sebagai berikut:

$$c = \begin{bmatrix} -0.5397 \\ 0.2877 \\ 3.3965 \end{bmatrix} \quad d = \begin{bmatrix} 12.9711 \\ 13.9690 \\ 0.3996 \\ -0.0015 \end{bmatrix} \quad e = \begin{bmatrix} -7.0000 \\ 4.0192 \\ 1.3756 \end{bmatrix}$$

- Mencari Solusi Dari Matriks Yang Telah Terdekomposisi

Dicari hingga memenuhi batasan toleransi 0.00001

- Dugaan awal (random)

$$x = \begin{bmatrix} 0.8147 \\ 0.9058 \\ 0.1270 \\ 0.9134 \end{bmatrix}$$

- Normalisasi

- $|x| = 1.5280$ $\quad x = \frac{x}{|x|} = \begin{bmatrix} 0.5332 \\ 0.5928 \\ 0.0831 \\ 0.5978 \end{bmatrix}$

Iterasi ke-1

- Forward substitution

$x_2 = x_2 - c_1 * x_1 = 0.5928 - (-0.5397 * 0.5332) = 0.8806$
$x_3 = x_3 - c_2 * x_2 = 0.0831 - (0.2877 * 0.8806) = -0.1702$
$x_4 = x_4 - c_3 * x_3 = 0.5978 - (3.3965 * -0.1702) = 1.1759$

- Back substitution

$$x_4 = \frac{x_4}{d_4} = \frac{1.1759}{-0.0015} = -783.9333$$
$$x_3 = (x_3 - e_3 * x_4)/d_3$$
$$= \frac{-0.1702 - 1.3756 * (-783.9333)}{0.3996}$$
$$= 2698.2193$$
$$x_2 = (x_2 - e_2 * x_3)/d_2$$
$$= \frac{0.8806 - 4.0192 * 2698.2193}{13.9690}$$
$$= -776.2762$$
$$x_1 = (x_1 - e_1 * x_2)/d_1$$
$$= \frac{0.5332 - (-7) * -776.2762}{12.9711}$$
$$= -418.8851$$

Maka $x = \begin{bmatrix} -418.8851 \\ -776.2762 \\ 2698.2193 \\ -783.9333 \end{bmatrix}$ $\quad |x| = 2944{,}9795$

$$x = \frac{x}{|x|} = \begin{bmatrix} -0.1422 \\ -0.2636 \\ 0.9162 \\ -0.2662 \end{bmatrix}$$

$$xOld = \begin{bmatrix} 0.5332 \\ 0.5928 \\ 0.0831 \\ 0.5978 \end{bmatrix}$$

- Pengecekan konvergensi

$$xOld.x = -0.3151 \quad \rightarrow xSign = -1$$

$$x = x * xSign = \begin{bmatrix} 0.1422 \\ 0.2636 \\ -0.9162 \\ 0.2662 \end{bmatrix}$$

$$xOld - x = \begin{bmatrix} 0.3910 \\ 0.3292 \\ 0.9993 \\ 0.3316 \end{bmatrix} \qquad |xOld - x| = 1.1704$$

Karena $|xOld - x| > toleransi$, maka berlanjut ke iterasi ke-2

Iterasi ke-2

$$x = \begin{bmatrix} 0.1422 \\ 0.2636 \\ -0.9162 \\ 0.2662 \end{bmatrix}$$

- Forward substitution

$x_2 = x_2 - c_1 * x_1 = 0.2636 - (-0.5397 * 0.1422) = 0.3403$
$x_3 = x_3 - c_2 * x_2 = -0.9162 - (0.2877 * 0.3403) = -1.0141$
$x_4 = x_4 - c_3 * x_3 = 0.2662 - (3.3965 * -1.0141) = 3.7106$

- Back substitution

$$x_4 = \frac{x_4}{d_4} = \frac{3.7106}{-0.0015} = -2473.7333$$

$$x_3 = (x_3 - e_3 * x_4)/d_3$$
$$= \frac{-1.0141 - 1.3756 * (-2473.733)}{0.3996}$$
$$= 8513.1457$$

$$x_2 = (x_2 - e_2 * x_3)/d_2$$
$$= \frac{0.3403 - 4.0192 * 8513.1457}{13.9690}$$
$$= -2449.4019$$

$$x_1 = (x_1 - e_1 * x_2)/d_1$$
$$= \frac{0.1422 - (-7) * -2449.4019}{12.9711}$$
$$= -1321.8363$$

Maka $x = \begin{bmatrix} -1321{,}8363 \\ -2449.4019 \\ 8513.1457 \\ -2473.7333 \end{bmatrix}$ $|x| = 9291{,}9227$

$$x = \frac{x}{|x|} = \begin{bmatrix} -0.1423 \\ -0.2636 \\ 0.9162 \\ -0.2662 \end{bmatrix}$$

$$xOld = \begin{bmatrix} 0.1422 \\ 0.2636 \\ -0.9162 \\ 0.2662 \end{bmatrix}$$

- Pengecekan konvergensi

$$xOld.x = -1.0000 \quad \rightarrow xSign = -1$$

$$x = x * xSign = \begin{bmatrix} 0.1423 \\ 0.2636 \\ -0.9162 \\ 0.2662 \end{bmatrix}$$

$$xOld - x = \begin{bmatrix} -0{,}1947.10^{-4} \\ -0{,}1242.10^{-4} \\ -0{,}1558.10^{-4} \\ -0{,}3092.10^{-4} \end{bmatrix} \quad |xOld - x| = 4{,}1623.10^{-5}$$

Karena $|xOld - x| < toleransi$, maka iterasi berhenti, dan vektor eigen untuk matriks A tridiagonal adalah $\begin{bmatrix} 0.1423 \\ 0.2636 \\ -0.9162 \\ 0.2662 \end{bmatrix}$ dengan nilai eigen sebesar $-5.9711 + \left(\frac{-1}{9291{,}9277}\right) = -5.9712$.

Sedangkan vektor eigen untuk matriks A adalah

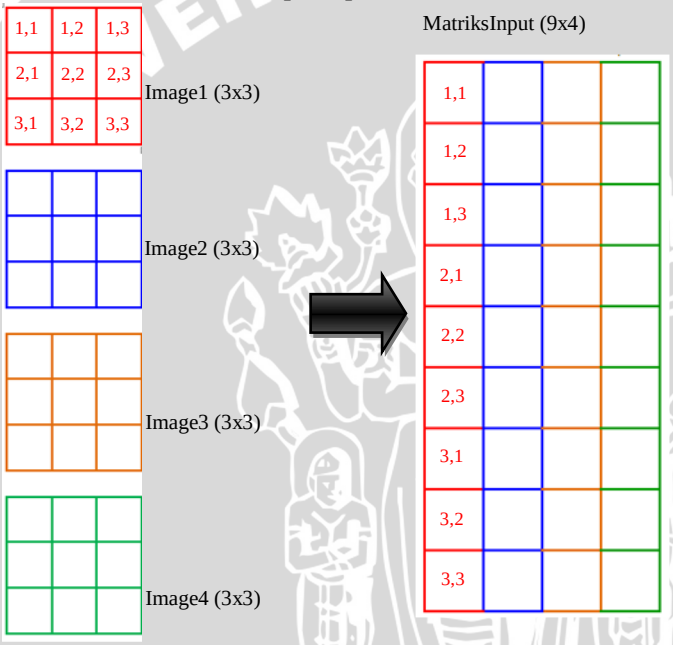
$$P * \begin{bmatrix} 0.1423 \\ 0.2636 \\ -0.9162 \\ 0.2662 \end{bmatrix} = \begin{bmatrix} 0.1423 \\ 0.4243 \\ 0.5351 \\ -0.7165 \end{bmatrix}$$

3.5 Contoh Perhitungan dalam Proses Pelatihan

3.5.1 Pembentukan Matriks Input

Sebuah citra wajah berukuran MxN akan dibaca dan dikonversi menjadi citra grayscale. Setelah itu, nilai keabuan dari tiap pixel citra akan diambil dan disimpan dalam sebuah array 1 dimensi. Jika terdapat 5 citra wajah yang digunakan sebagai data latih, maka akan terdapat 5 array yang menyimpan nilai keabuan tiap-tiap citra.

MatriksInput dibentuk dari tiap array citra wajah yang telah dikurangi dengan rata-rata. Setiap array citra tersebut dianggap sebagai vektor kolom dan disusun seperti pada Gambar 3.18



Gambar 3.18 Pembentukan Matriks Input

Dalam contoh perhitungan bab ini digunakan 5 buah gambar masing-masing berukuran 5 pixel (Ditunjukkan oleh Gambar 3.19). Setelah dilakukan proses seperti pada Gambar 3.18, diperoleh matriks input Γ,

Gambar 3.19 Citra Berukuran 5x5 yang Dipakai dalam Contoh Perhitungan

$$\Gamma = \begin{bmatrix} 199 & 196 & 186 & 207 & 207 \\ 254 & 255 & 254 & 253 & 253 \\ 254 & 254 & 254 & 254 & 255 \\ 255 & 254 & 253 & 254 & 254 \\ 254 & 253 & 255 & 255 & 255 \\ 167 & 195 & 162 & 185 & 186 \\ 197 & 224 & 170 & 193 & 192 \\ 255 & 253 & 254 & 255 & 255 \\ 255 & 254 & 255 & 254 & 255 \\ 253 & 255 & 255 & 254 & 254 \\ 165 & 194 & 161 & 185 & 186 \\ 166 & 200 & 161 & 186 & 186 \\ 198 & 229 & 179 & 201 & 200 \\ 253 & 254 & 254 & 254 & 254 \\ 255 & 254 & 253 & 254 & 254 \\ 165 & 189 & 161 & 185 & 185 \\ 166 & 197 & 173 & 254 & 184 \\ 165 & 207 & 161 & 185 & 186 \\ 198 & 222 & 169 & 195 & 196 \\ 254 & 254 & 255 & 254 & 254 \\ 166 & 186 & 161 & 186 & 186 \\ 165 & 189 & 161 & 184 & 255 \\ 166 & 200 & 159 & 186 & 184 \\ 164 & 192 & 161 & 185 & 185 \\ 202 & 192 & 174 & 192 & 193 \end{bmatrix} \quad \psi = \begin{bmatrix} 199.0 \\ 253.8 \\ 254.2 \\ 254.0 \\ 254.4 \\ 179.0 \\ 195.2 \\ 254.4 \\ 254.6 \\ 254,2 \\ 178.2 \\ 179.8 \\ 201.4 \\ 253.8 \\ 254.0 \\ 177.0 \\ 194.8 \\ 180.8 \\ 196.0 \\ 254.2 \\ 177.0 \\ 190.8 \\ 179.0 \\ 177.4 \\ 190.6 \end{bmatrix}$$

MatriksInput setelah normalisasi ($\Phi = \Gamma - \psi$)

$$\Phi = \begin{bmatrix} 0.0 & -3.0 & -13.0 & 8.0 & 8.0 \\ 0.2 & 1.2 & 0.2 & -0.8 & -0.8 \\ -0.2 & -0.2 & -0.2 & -0.2 & 0.8 \\ 1.0 & 0.0 & -1.0 & 0.0 & 0.0 \\ -0.4 & -1.4 & 0.6 & 0.6 & 0.6 \\ -12.0 & 16.0 & -17.0 & 6.0 & 7.0 \\ 1.8 & 28.8 & -25.2 & -2.2 & -3.2 \\ 0.6 & -1.4 & -0.4 & 0.6 & 0.6 \\ 0.4 & -0.6 & 0.4 & -0.6 & 0.4 \\ -1.2 & 0.8 & 0.8 & -0.2 & -0.2 \\ -13.2 & 15.8 & -17.2 & 6.8 & 7.8 \\ -13.8 & 20.2 & -18.8 & 6.2 & 6.2 \\ -3.4 & 27.6 & -22.4 & -0.4 & -1.4 \\ -0.8 & 0.2 & 0.2 & 0.2 & 0.2 \\ 1.0 & 0.0 & -1.0 & 0.0 & 0.0 \\ -12.0 & 12.0 & -16.0 & 8.0 & 8.0 \\ -28.8 & 2.2 & -21.8 & 59.2 & -10.8 \\ -15.8 & 26.2 & -19.8 & 4.2 & 5.2 \\ 2.0 & 26.0 & -27.0 & -1.0 & 0.0 \\ -0.2 & -0.2 & 0.8 & -0.2 & -0.2 \\ -11.0 & 9.0 & -16.0 & 9.0 & 9.0 \\ -25.8 & -1.8 & -29.8 & -6.8 & 64.2 \\ -13.0 & 21.0 & -20.0 & 7.0 & 5.0 \\ -13.4 & 14.6 & -16.4 & 7.6 & 7.6 \\ 11.4 & 1.4 & -16.6 & 1.4 & 2.4 \end{bmatrix}$$

Sedangkan kovariannya adalah

$$C = \Phi^T \Phi$$

$$C = \begin{bmatrix} 3020.6 & -1797.2 & 3031.8 & -2220.0 & -2035.4 \\ -1797.2 & 4772.0 & -4462.0 & 891.4 & 595.8 \\ 3031.8 & -4462.0 & 6189.0 & -2080.0 & -2679.2 \\ -2219.8 & 891.4 & -2079.6 & 4014.8 & -606.8 \\ -2035.4 & 595.8 & -2679.2 & -606.8 & 4725.6 \end{bmatrix}$$

Matriks ini kemudian dicari nilai dan vector eigennya

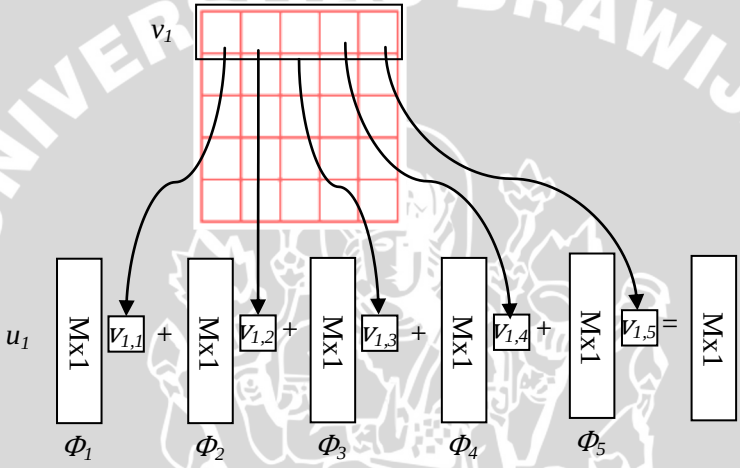
Nilai eigen=$\{13339, 5014.34, 3594.3, 513.79, 513.79\}$

Sedangkan vektor eigennya

$$v = \begin{bmatrix} 0.3983 & 0.0675 & 0.9412 & 0.1300 & 0.1300 \\ -0.4790 & -0.1816 & 0.1226 & 0.5485 & 0.5485 \\ 0.6639 & -0.0107 & -0.0520 & 0.6700 & 0.6700 \\ -0.2680 & -0.6230 & -0.2670 & 0.2266 & 0.2266 \\ -0.3150 & 0.7578 & -0.1600 & 0.4267 & 0.4267 \end{bmatrix}$$

3.5.2 Perolehan Wajah Eigen

Wajah eigen diperoleh dengan menggunakan persamaan 2.45. Proses perhitungannya ditunjukkan seperti pada Gambar 3.20.



Gambar 3.20 Ilustrasi Perhitungan Wajah Eigen

Setiap wajah eigen diperoleh dengan menggunakan persamaan 2.45 tersebut. agar dapat ditampilkan sebagai sebuah citra, maka nilai dari tiap wajah eigen tersebut diskala menjadi nilai antara 0-255.

Setelah diskala, maka nilai eigenface untuk contoh kasus ini adalah

$$u = \begin{bmatrix} 21 & 7 & 20 & 40 & 61 \\ 21 & 26 & 26 & 24 & 22 \\ 22 & 21 & 21 & 20 & 22 \\ 19 & 19 & 20 & 20 & 20 \\ 22 & 16 & 15 & 17 & 18 \\ 44 & 115 & 131 & 147 & 164 \\ 18 & 147 & 171 & 165 & 157 \\ 20 & 14 & 14 & 16 & 17 \\ 21 & 18 & 17 & 16 & 17 \\ 23 & 27 & 26 & 26 & 25 \\ 46 & 117 & 133 & 150 & 170 \\ 47 & 138 & 155 & 171 & 187 \\ 28 & 151 & 173 & 172 & 168 \\ 23 & 24 & 23 & 24 & 24 \\ 19 & 19 & 20 & 20 & 20 \\ 44 & 97 & 113 & 133 & 153 \\ 75 & 85 & 105 & 255 & 228 \\ 51 & 168 & 187 & 198 & 211 \\ 18 & 134 & 160 & 157 & 157 \\ 22 & 21 & 20 & 19 & 19 \\ 42 & 82 & 97 & 120 & 143 \\ 96 & 61 & 89 & 72 & 234 \\ 45 & 140 & 159 & 176 & 189 \\ 46 & 112 & 127 & 146 & 166 \\ 0 & 6 & 22 & 26 & 32 \end{bmatrix}$$

Setiap kolom dari matriks u mewakili sebuah wajah eigen.

3.6 Contoh Perhitungan Dalam Proses Pengenalan

a) Preproses wajah input (Γ)

Wajah input menggunakan img1 yang disertakan dalam proses pelatihan. Sebelum diproyeksikan ke ruang wajah, terlebih dulu Γ dikurangi dengan rata-rata seluruh data latih ψ hingga didapatkan,

$$\Gamma = \begin{bmatrix} 199 \\ 254 \\ 254 \\ 255 \\ 254 \\ 167 \\ 197 \\ 255 \\ 255 \\ 253 \\ 165 \\ 166 \\ 198 \\ 253 \\ 255 \\ 165 \\ 166 \\ 165 \\ 198 \\ 254 \\ 166 \\ 165 \\ 166 \\ 164 \\ 202 \end{bmatrix} \quad \psi = \begin{bmatrix} 199.0 \\ 253.8 \\ 254.2 \\ 254.0 \\ 254.4 \\ 179.0 \\ 195.2 \\ 254.4 \\ 254.6 \\ 254,2 \\ 178.2 \\ 179.8 \\ 201.4 \\ 253.8 \\ 254.0 \\ 177.0 \\ 194.8 \\ 180.8 \\ 196.0 \\ 254.2 \\ 177.0 \\ 190.8 \\ 179.0 \\ 177.4 \\ 190.6 \end{bmatrix} \quad \Gamma - \psi = \begin{bmatrix} 0.0 \\ 0.2 \\ -0.2 \\ 1.0 \\ -0.4 \\ -12.0 \\ 1.8 \\ 0.6 \\ 0.4 \\ -1.2 \\ -13.2 \\ -13.8 \\ -3.4 \\ -0.8 \\ 1.0 \\ -12.0 \\ -28.8 \\ -15.8 \\ 2.0 \\ -0.2 \\ -11.0 \\ -25.8 \\ -13.0 \\ -13.4 \\ 11.4 \end{bmatrix}$$

b) Perhitungan bobot wajah input(Γ)

Perhitungan bobot wajah input Ω menggunakan persamaan 2.46, di mana *k* adalah urutan wajah eigen

$$\omega_k = u_k^T(\Gamma - \Psi)$$

Dengan demikian diperoleh nilai ω sebagai berikut:

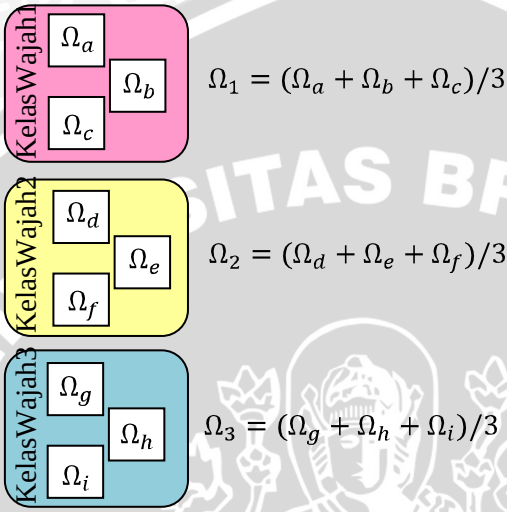
$\omega_1 = 154698.2;\ \omega_2 = 324948; \omega_3 = 374564.5; \omega_4 = 422026.6;$

$\omega_5 = 471444.2;$

$\Omega^T = [154698.2 \quad 324948 \quad 374564.5 \quad 422026.6 \quad 471444.2]$

Selanjutnya sebagai perbandingan, dihitung bobot setiap kelas wajah Ω_i di mana i menunjukkan urutan kelas wajah. Ω_i diperoleh

dari rata-rata Ω tiap wajah dalam kelas wajah i. Proses perhitungan tersebut ditunjukkan oleh Gambar 3.21.



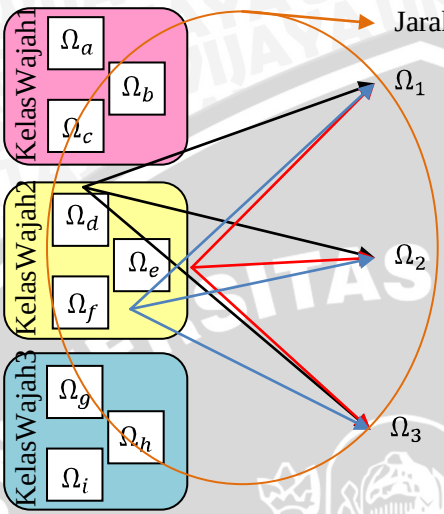
Gambar 3.21 Ilustrasi Perhitungan Ω_i

Karena dalam kasus ini hanya terdapat 1 kelas wajah, maka nilai Ω_1 adalah

$\Omega_1^T = [1634446.2 \quad 341761.1 \quad 394242.7 \quad 447313.1 \quad 501873.1]$

c) Menentukan threshold (θ)

Untuk menemukan *threshold*, terlebih dahulu dihitung jarak minimal tiap kelas wajah yang ada. Seperti yang ditunjukkan pada gambar Gambar 3.22



Gambar 3.22 Ilustrasi Pencarian Jarak Euclidean Minimal

Setelah jarak euclidean minimal tiap kelas ditemukan, maka kemudian dicari jarak yang paling maksimal dari jarak euclidean minimal tersebut.

$$threshold = 0.8x\ maksEuclideanDistance$$

Threshold yang didapat untuk kasus ini adalah 66636,4215.

d) Pencarian jarak Euclidean minimal

Selanjutnya untuk menentukan sebuah wajahInput (Γ)dikenali atau tidak, dilakukan pencarian jarak terkecil antara bobot wajahInput (Γ) tersebut dengan bobot rata-rata tiap kelas. wajahInput dianggap dikenali sebagai bagian dari suatu kelas jika jaraknya paling dekat dengan kelas tersebut dan besarnya lebih kecil dari threshold.

Jarak minimal yang didapatkan untuk kasus ini adalah 48080,64968(didapatkan dari perhitungan jarak euclidean antara Ω dan Ω_1). Karena nilai tersebut lebih kecil dari threshold, maka disimpulkan wajahInput dikenali.

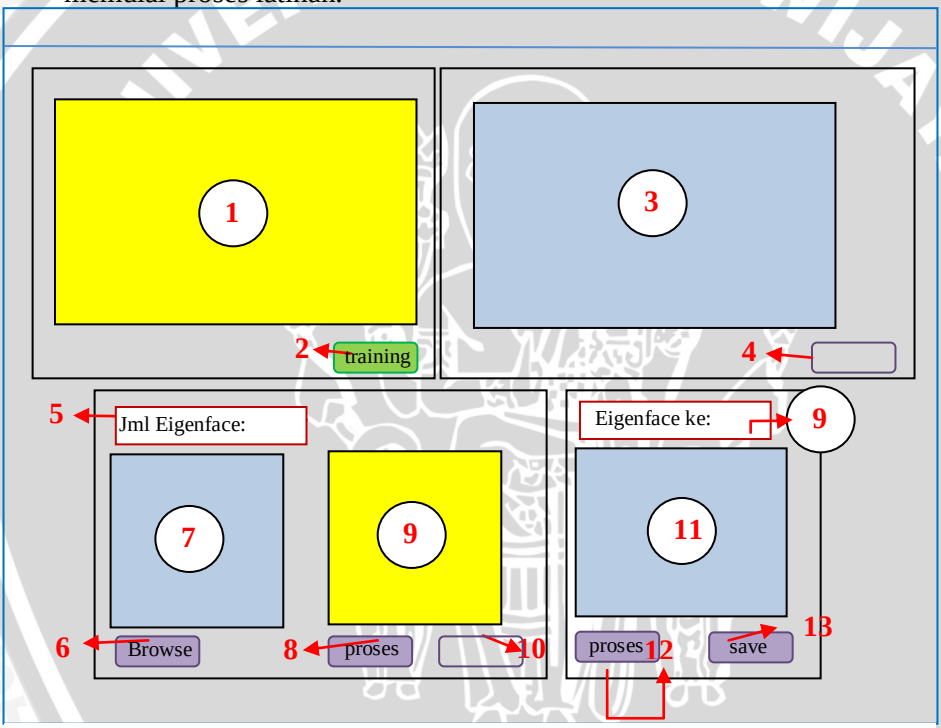
3.7 Perancangan Antarmuka

Perancangan Antarmuka untuk aplikasi pengenalan wajah ini terdiri dari beberapa bagian, yaitu:

a. Bagian Pelatihan

Bagian ini digunakan untuk menginputkan data pelatihan untuk kemudian dilatih. Pada form ini ditampilkan data latihan dan juga data hasil pelatihan yang berupa wajah-wajah eigen.

Area yang diberi nomor 1 pada Gambar 3.23 menunjukkan area yang berfungsi menampilkan citra wajah yang digunakan sebagai data latihan. Sedangkan nomor 2 adalah tombol yang digunakan untuk memulai proses latihan.



Gambar 3.23 Rancangan Antarmuka

b. Tampilan matriks eigenface

Pada bagian ini (yang ditunjukkan dengan nomor 3 pada Gambar 3.23) menampilkan eigenface-eigenface yang dihasilkan setelah proses pelatihan. Nomor 4 merupakan waktu yang diperlukan selama proses pelatihan berlangsung.

c. Bagian Pengenalan

Bagian pengenalan digunakan untuk melakukan pengenalan terhadap suatu wajah inputan, apakah wajah tersebut dikenali atau tidak. Sebelum dilakukan proses pengenalan, jumlah eigenface terlebih dahulu (nomor 5). Setelah itu dilakukan pencarian file citra wajah yang akan dikenali dengan menekan tombol "browse" (nomor 6), citra yang didapatkan akan ditampilkan pada area yang ditandai dengan nomor 7. Selanjutnya jika tombol proses (nomor 8) ditekan akan dilakukan proses pengenalan. Jika wajah tersebut dikenali, maka citra wajah yang diduga sebagai wajah yang satu jenisdengan wajah input akan ditampilkan pada area yang ditunjukkan dengan nomor 9. Waktu yang diperlukan untuk proses pengenalan akan ditampilkan pada bagian yang ditandai dengan nomor 10.

d. Bagian untuk menampilkan eigenface

Untuk menampilkan eigenface, terlebih dulu pengguna memilih eigenface ke berapa yang ingin ditampilkan (nomor9). Setelah ditekan tombol proses (nomor 12) eigenface akan ditampilkan pada area yang ditunjukkan dengan nomor11. Jika ingin menyimpan gambar eigenface tersebut, maka proses penyimpanan dapat dijalankan dengan menekan tombol save (nomor 13).

3.8 Perancangan Uji Coba

Untuk mengetahui kemampuan sistem dalam mengenali wajah, dilakukan pengujian dengan skenario sebagai berikut:

a) Skenario 1: Pengujian Data Latih

Pengujian pertama dilakukan untuk mengetahui kemampuan sistem dalam mengenali kembali wajah yang digunakan sebagai data latihan. Data wajah yang digunakan sebagai latihan sekaligus sebagai data uji adalah 50 citra wajah dari 5 individu yang berbeda (masing-masing 10 citra wajah). Hasil dari proses pengenalan kemudian dicatat dalam Tabel 3.1, dan kemudian dihitung *precision* dan *recall*-nya.

Tabel 3.1 Hasil Proses Pengenalan Wajah

No.	Nama citra wajah	Dikenali sebagai

b) Skenario 2: Pengujian Banyaknya Jumlah Wajah Eigen

Pengujian ke-2 dilakukan untuk mengetahui hubungan antara jumlah wajah eigen yang digunakan dengan *precision* dan *recall*. Data latihan yang digunakan adalah sebanyak 50 citra wajah dari 5 individu, sedangkan data uji adalah 13 wajah dari individu yang tidak dilatihkan, dan 10 citra wajah dari individu yang termasuk dalam pelatihan. Hasil dari proses pengenalan dengan menggunakan jumlah eigenface yang berbeda akan dicatat dalam Tabel 3.1. Kemudian nilai Precision dan Recall yang didadaptkan dicatat dalam Tabel 3.2.

Tabel 3.2 Hubungan antara Jumlah Eigenface dengan Precision dan Recall

	Jumlah eigenface									
	5	10	15	20	25	30	35	40	45	50
Precision										
Recall										

c) Skenario 3: Pengujian Banyaknya Jumlah Data Latihan

Pengujian ini bertujuan untuk mengetahui hubungan antara jumlah data latihan dengan *precision* dan *recall*. Data latihan yang akan digunakan adalah kelipatan 5 dari 5 hingga 50. Sedangkan data uji yang digunakan adalah 13 wajah dari individu yang tidak dilatihkan, dan 10 citra wajah dari individu yang termasuk dalam pelatihan. Hasil dari proses pengenalan menggunakan jumlah data latihan yang berbeda akan dicatat dalam Tabel 3.1. Kemudian nilai precision dan recall dihitung dan dicatat dalam Tabel 3.3.

Tabel 3.3 Hubungan antara Jumlah Data Latihan dengan Precision dan Recall

	Jumlah data latihan									
	5	10	15	20	25	30	35	40	45	50
Precision										
Recall										

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan perangkat lunak ini adalah:

1. Intel ® Core TM 2 Duo Processor T5500
2. Harddisk 80 GB
3. Monitor 14'
4. Keyboard
5. Mouse

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan aplikasi ini adalah:

1. Sistem Operasi Windows XP
2. Microsoft Visual Studio Express 2008

4.2 Implementasi Program

4.2.1 Implementasi Antarmuka

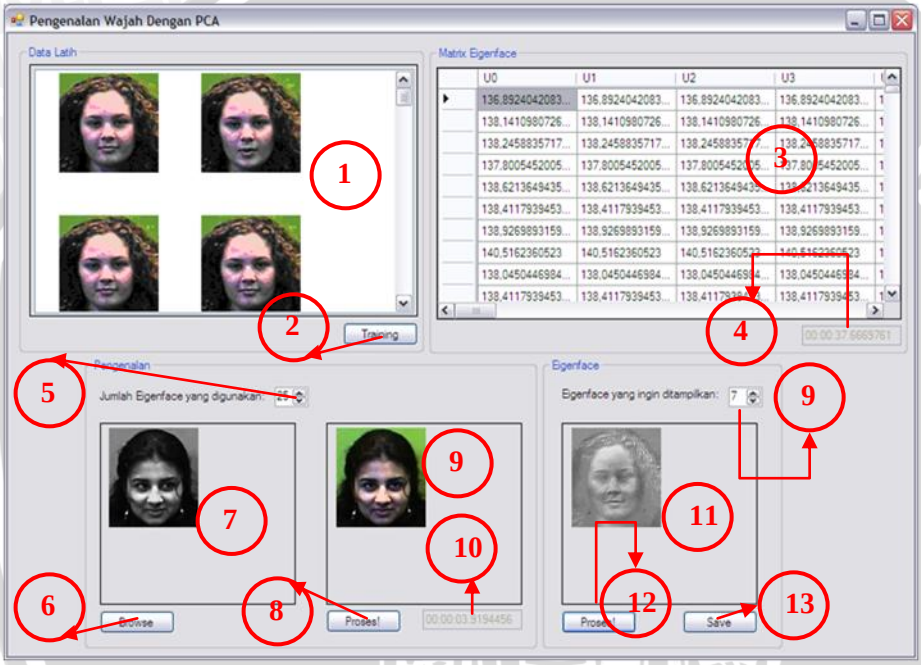
Implementasi antarmuka terdiri dari satu form utama dengan 4 bagian utama, yaitu bagian "Data Latih", "Matrix Eigenface", "Pengenalan", dan "Eigenface". Implementasi antarmuka ditunjukkan oleh Gambar 4.1.

Untuk melakukan proses pelatihan, maka digunakan tombol "Training" yang ditunjukkan oleh nomor 2. Setelah folder data latihan dipilih, maka wajah-wajah tersebut akan ditampilkan pada *picturebox* yang ditunjukkan oleh nomor 1. Dan hasil dari eigenface akan ditampilkan pada *dataGridView* (ditunjukkan oleh nomor 3), dan waktu yang diperlukan selama proses pelatihan ditampilkan pada *textbox* (nomor 4).

Untuk melakukan proses pengenalan terhadap sebuah wajah masukan, terlebih dahulu wajah masukan tersebut dipilih menggunakan tombol *Browse* yang ditunjukkan oleh nomor 6. Setelah wajah input ditampilkan, proses pengenalan akan dilakukan setelah tombol "Proses!" (ditunjukkan oleh nomor 8) ditekan. Jika ditemukan wajah masukan

tersebut dikenali, maka akan ditampilkan wajah yang menyerupai di *picturebox* yang ditandai dengan nomor 9, jika tidak, maka akan muncul *message box* yang bertuliskan "unknown".

Jika ingin menampilkan eigenface dalam bentuk gambar, maka tombol "Proses" yang ditujukkan oleh nomor 12 akan mengeksekusi proses penampilan wajah eigen setelah ditekan. Wajah eigen akan ditampilkan pada *picturebox*(nomor 11).



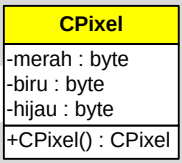
Gambar 4.1 Implementasi Antarmuka

4.2.2 Implementasi kelas

Pada implementasi program, *listing program* diimplementasikan dalam bentuk kelas-kelas. Berdasarkan fungsinya, kelas-kelas yang membangun aplikasi ini dapat dikategorikan menjadi dua, yaitu kelompok kelas yang digunakan untuk menangani citra serta kelompok kelas yang menangani perhitungan. Kelompok kelas pertama terdiri dari kelas `CPixel` dan kelas `CGambar`. Sedangkan kelompok kedua terdiri dari kelas `CEigenSys` dan `CMatrix`.

1. Kelas CPixel

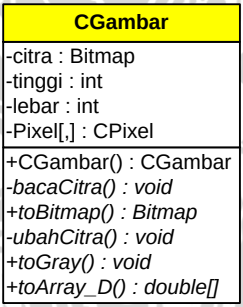
Kelas ini berfungsi merepresentasikan pixel yang terdapat dalam gambar. Setiap pixel memiliki komponen *Red*, *Green* dan *Blue*. Kelas ini tidak terdapat method selain konstruktor, karena kelas ini merupakan satuan data gambar yang akan diolah. Class diagram kelas ini ditunjukkan oleh Gambar 4.2.



Gambar 4.2 *Class Diagram* Kelas CPixel

2. Kelas CGambar

Gambar wajah yang digunakan dalam aplikasi ini direpresentasikan oleh kelas CGambar. Seperti halnya gambar tersusun atas pixel-pixel, kelas CGambar pun juga tersusun dari CPixel. Kelas ini berisi method-method yang berhubungan dengan pengolahan gambar. Prosedur dan fungsi yang terdapat pada kelas ini ditunjukkan pada Tabel 4.1. Sedangkan *class diagram*-nya ditunjukkan pada Gambar 4.3.



Gambar 4.3 *Class Diagram* kelas CGambar

Tabel 4.1 Daftar Prosedur dan Fungsi Kelas CGambar

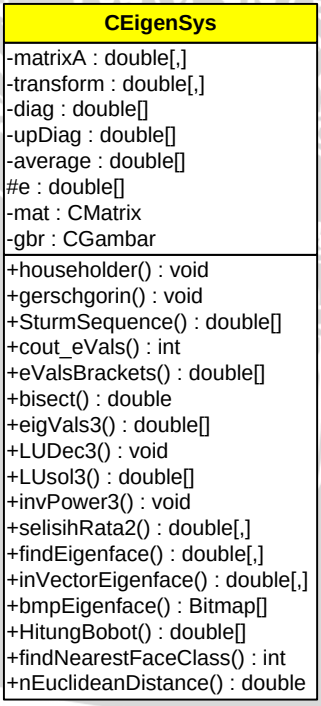
Nama Prosedur/Fungsi	Deskripsi
`public Bitmap[] toBitmap(double[,] value, int height, int width)`	Menyusun sekumpulan gambar Bitmap dari sebuah matriks 2 dimensi dengan ukuran height x width
`private void bacaCitra()`	Membaca nilai-nilai RGB dari gambar

`private void ubahCitra()`	Mengubah nilai RGB citra dengai nilai RGB yang baru
`public void toGray()`	Mengubah citra berwarna menjadi citra *grayscale*
`public double[] toArray_D(CGambar gbr)`	Menyusun nilai grayscale sebuah citra menjadi sebuah array 1 dimensi

3. Kelas `CEigenSys`

Kelas CEigenSys berisi semua prosedur /fungsi yang digunakan untuk menemukan wajah eigen. Segala proses perhitungan mulai dari reduksi matriks, pencarian nilai eigen dan vektor eigen, pencarian wajah eigen, hingga perhitungan saat proses pengenalan, terdapat dalam kelas ini. Pada Gambar 4.4 ditunjukkan diagram kelas SEigenSys.

Semua prosedur/fungsi yang terdapat dalam kelas ini dideskripsikan pada Tabel 4.2. Sekumpulan wajah yang akan dilatih dikumpulkan menjadi sebuah matriks terlebih dahulu. Kovarian dari matriks tersebut yang akan diolah dalam kelas ini.



Gambar 4.4 *Class Diagram* kelas CEigenSys

Tabel 4.2 Daftar Prosedur dan Fungsi kelas CEigenSys

Nama Prosedur/Fungsi	Deskripsi
`public void householder(double[,]A)`	Mengubah matriks menjadi bentuk matriks tridiagonal
`public void gerschgorin(ref double eValMin, ref double eValMax)`	Menemukan nilai eigen terendah dan tertinggi
`public double[] SturmSequence(double lambda)`	Menghitung deret Sturm untuk membantu menemukan akar
`public int count_eVals(double lambda)`	Menghitung jumlah nilai eigen yang terdapat pada selang lambda
`public double[] eValBrackets(int m)`	Pengurungan range yang mungkin terdapat nilai eigen
`public double[] eigVals3(int maxEigVals)`	Menemukan nilai eigen sebanyak maxEigVals
`public void powerMethod(double s, ref double[] eigVec, ref double eigVal)`	Fungsi untuk menemukan nilai eigen dengan metode pangkat
`public double[,] selisihRata2(double[,] matinput)`	Fungsi untuk menghitung selisih antara wajah dengan rata-rata wajah
`public double[,] findEigenfaces(double[,] matinput)`	Fungsi untuk menemukan wajah eigen dalam bentuk matriks
`public Bitmap[] bmpEigenface(double[,]eVecMat, double[,] matinput, int height, int width)`	Fungsi untuk membentuk wajah eigen dari matriks eigen
`public double[] HitungBobot(double[,] eigenface, double[] inputFace, double[] avg)`	Fungsi untuk menghitung bobot yang diperoleh dari proyeksi sebuah wajah ke wajah-wajah eigen
`public int findNearestFaceClass(CData []dataLatih, double[] newImage, double [,] eigenface, double [] avg)`	Fungsi untuk menemukan kelas wajah yang paling mirip dengan sebuah wajah input
`public double nEuclideanDistance(double [] vec1, double[] vec2)`	Fungsi untuk menghitung jarak euclidean

Beberapa fungsi dan prosedur utama yang terdapat dalam kelas ini akan dijelaskan sebagai berikut:

a. Prosedur Householder

Prosedur ini mengubah nilai matriks input (matriks A) menjadi bentuk tridiagonal. Prosedur ini merupakan aplikasi dari prosedur Householder yang terdapat pada Gambar 2.11.

```
1   public void householder(double[,]A)
2   {
3      int N = A.GetLength(0);
4      double[] uTemp1;
5      double[,] u,v, vTemp,uTemp;
6      double uMag, H, g;
7      int h, i, j, k, m, n;
8
9      for (k = 0; k < N - 2; k++)
10     {
11      u = new double[N - k - 1, 1];
12      v = new double[N - k - 1, N - k - 1];
13      double[,]simpan,simpan2 =new double[N -k-1,N-k-1];
14       //inisialisasi vektor u
15       for (i = 0; i < N - k - 1; i++)
16       {
17         u[i,0] = A[i + k + 1, k];
18      }
19      uTemp1 = mat.dot(u, u);
20      uMag = Math.Sqrt(uTemp1[0]);
21      if (u[0,0] < 0.0)
22      {
23       uMag = -uMag;
24      }
25      u[0,0] = u[0,0] + uMag;
26      //save u in lower part of A
27
28      for (j = 0; j < N - k - 1; j++)
29      {
30         A[j + k + 1,k] = u[j,0];
31      }
32
33     uTemp1 = mat.dot(u, u);
34     H = uTemp1[0] / 2;
35
36     for (j = 0; j < N - k - 1; j++)
37     {
38      for (n = 0; n < N - k - 1; n++)
39      {
40        v[j, n] = A[j + k + 1, n + k + 1];
```

```
41        }
42      }
43
44
45      v = mat.multiply(v, u);
46      v = mat.constdiv(v, H);
47      uTemp1 = mat.dot(u, v);
48      g = uTemp1[0] / (2 * H);
49      v = mat.subtract(v, mat.multiply(u, g));
5
51      vTemp = mat.multiply(v, mat.transpose(u));
52      uTemp = mat.multiply(u, mat.transpose(v));
53      simpan = mat.add(vTemp, uTemp);
54      for (h = 0; h < N - k - 1; h++)
55      {
56        for (int o = 0; o < N - k - 1; o++)
57        {
58          simpan2[h, o] = A[h + k + 1, o + k + 1];
59        }
60      }
61      simpan2 = mat.subtract(simpan2, simpan);
62      for (h = 0; h < N - k - 1; h++)
63      {
64        for (int o = 0; o < N - k - 1; o++)
65        {
66          A[h + k + 1, o + k + 1] = simpan2[h, o];
67        }
68      }
69
70     A[k, k + 1] = -uMag;
71  }//endfor k
72
73  for (m = 0; m < N; m++)
74  {
75     diag[m] = A[m, m];
76      if (m + 1 < N)
77      {
78        upDiag[m] = A[m, m + 1];
79        e[m] = A[m, m + 1];
80      }
81  }
82
83  for (k = 0; k < N - 2; k++)
84  {
85      //nilai u
86      u = new double[N - k - 1,1];
87      v = new double[N, N - k - 1];
88     double[,] subMatTrans = new double[N, N - k - 1];
```

```
89
90      for (i = 0; i < N - k - 1; i++)
91      {
92        u[i,0] = A[i + k + 1, k];
93      }
94      uTemp1 = mat.dot(u, u);
95      H = uTemp1[0] / 2;
96
97      //nilai v
98      for (i = 0; i < N; i++)
99      {
100       for (j = 0; j < N - k - 1; j++)
101       {
102         v[i, j] = transform[i, j + k + 1];
103       }
104     }
105     v = mat.multiply(v, u);
106     v = mat.constdiv(v, H);
107
108     for (i = 0; i < N; i++)
109     {
110       for (j = 0; j < N - k - 1; j++)
112       {
113         subMatTrans[i, j] = transform[i, j + k + 1];
114       }
115     }
116     vTemp = mat.multiply(v, mat.transpose(u));
117     subMatTrans = mat.subtract(subMatTrans, vTemp);
118     for (i = 0; i < N; i++)
119     {
120       for (j = 0; j < N - k - 1; j++)
121       {
123         transform[i, j + k + 1] = subMatTrans[i, j];
124       }
125     }
126 }//endfor k
127 }//endhouseholder
```

Source Code 4.1 Prosedur Householder

Prosedur ini mengubah `matrixA` menjadi matriks tridiagonal dan menghitung `transform` yang merupakan matriks transformasi yang dihasilkan dari proses transformasi `matrixA`. Matriks transformasi ini nantinya digunakan untuk menghitung vektor eigen. Saat proses reduksi berlangsung, bagian `matrixA` yang terletak di bawah diagonal digunakan untuk menyimpan vektor u yang nantinya digunakan untuk menghituk matriks `transform`.

b. Prosedur Greschgorin

Source Code 4.2 menunjukkan sebuah prosedur yang berfungsi menghitung batas atas dan batas bawah nilai-nilai eigen dari sebuah matriks simetris. `diag` dan `upDiag` merupakan variabel yang masing-masing menyimpan diagonal dan bagian atas diagonal matriks simetris yang akan dicari batas atas(`eValMax`) dan batas bawah nilai eigennya(`eValMin`).

```
1   public void gerschgorin(ref double eValMin, ref double
2   eValMax)
3   {
4      int N = diag.Length;
5      double eVal = 0.0;
6      eValMin = diag[0] - Math.Abs(upDiag[0]);
7      eValMax = diag[0] + Math.Abs(upDiag[0]);
8      //mencari eValMin dan eValMax
9      for (int i = 1; i < N - 1; i++)
10     {
11       eVal = diag[i] - Math.Abs(upDiag[i]) -
12             Math.Abs(upDiag[i - 1]);
13       if (eVal < eValMin)
14       {
15         eValMin = eVal;
16       }
17       eVal = diag[i] + Math.Abs(upDiag[i]) +
18             Math.Abs(upDiag[i - 1]);
19       if (eVal > eValMax)
20       {
21         eValMax = eVal;
22       }
23     }
24     eVal = diag[N - 1] - Math.Abs(upDiag[N - 2]);
25     if (eVal < eValMin)
26     {
27       eValMin = eVal;
28     }
29     eVal = diag[N - 1] + Math.Abs(upDiag[N - 2]);
30     if (eVal > eValMax)
31     {
32       eValMax = eVal;
33     }
34  }//end gerschgorin
```

Source Code 4.2 Prosedur Greschgorin

c. Fungsi Sturm Sequence

Berdasarkan `diag` dan `upDiag` matriksA, serta variabel input `lambda`, fungsi ini akan mengembalikan sebuah deret Sturm. Fungsi Strums Sequence ditunjukkan oleh Source Code 4.3.

```
1   public double[] SturmSequence(double lambda)
2   {
3      int n = diag.Length+1;
4      double[] P = mat.init(n, 1.0);
5
6      P[1] = diag[0] - lambda;
7      for (int i = 1; i < n - 1; i++)
8      {
9        P[i + 1] = (diag[i] - lambda) * P[i] - (upDiag[i -
10           1] * upDiag[i - 1]) * P[i - 1];
11     }
12     return P;
13  }//endSturmSeq
```

Source Code 4.3 Fungsi Sturm Sequence

d. Fungsi count_eVals

Source Code 4.4 menunjukkan sebuah fungsi yang mengembalikan jumlah nilai eigen berdasarkan perubahan tanda yang terdapat dalam deret Sturm `P`.

```
1   public int count_eVals(double lambda)
2   {
3      double[] P = SturmSequence(lambda);
4      int n = P.Length;
5      int pSign=1;int oldSign=1; int num_eVals = 0;
6      for (int i = 0; i < n; i++)
7      {
8        if (Double.IsNaN(P[i]))
9        {
10          pSign = oldSign;
11       }
12
13       else
14       {
15          pSign = Math.Sign(P[i]);
16          if (pSign == 0)
17          {
18            pSign = (-1) * oldSign;
19          }
20          if (pSign * oldSign < 0)
21          {
```

```
22                  num_eVals++;
23              }
24          }
25          oldSign = pSign;
26      }
27      return num_eVals;
28  }//end count_eVals
```

Source Code 4.4 Fungsi untuk Menghitung Banyaknya Nilai Eigen

e. Fungsi eValBrackets

Source Code 4.5 menunjukkan sebuah fungsi yang mengembalikan sebuah array `r` yang menunjukkan batas-batas atau range-range terdapatnya nilai eigen. Input `m` menunjukkan nilai eigen yang ingin didapatkan. Fungsi ini diawali dengan mencari nilai batas atas (`eValMax`) dan nilai batas bawah (`eValMin`). Untuk range yang pertama kali dicari adalah range yang memiliki nilai eigen sebanyak `m`. Jumlah nilai eigen dalam range tersebut dihitung dengan bantuan fungsi `count_eVals`. Range pertama ini akan disimpan sebagai `r[0]` dan `r[m+1]`, range kedua yang dicari, akan disimpan sebagai `r[0]` dan `r[m]` yang memiliki `m-1` nilai eigen, demikian seterusnya, hingga range terakhir yang disimpan sebagai `r[0]` dan `r[1]` yang memiliki nilai eigen sebanyak 1.

```
1   public double[] eValBrackets(int m)
2   {
3      double[] r = new double[m + 1];
4      double eVal, h;
5      int num_eVals = 1;
6      double eValMin, eValMax;
7      eValMin = eValMax = 0.0;
8      //eigenvalue terkecil
9
10     gerschgorin(ref eValMin, ref eValMax);
11     r=mat.init(m+1, 1);
12     r[0]= eValMin;
13     for (int k = m - 1; k >= 0; k--)
14     {
15       eVal = (eValMax + eValMin) / 2;
16       h = (eValMax - eValMin) / 2;
17       for (int i = 0; i < 100; i++)
18       {
19        //temukan jumlah eigenvalue <eVal
20        num_eVals = count_eVals(eVal);
```

```
21        //bisect again & find half containing eVal
22        h = h / 2;
23        if (num_eVals < k+1 )
24        {
25           eVal = eVal + h;
26        }
27        else if (num_eVals > k+1)
28        {
29           eVal = eVal - h;
30        }
31         else break;
32       }
33       eValMax = eVal;
34       r[k + 1] = eValMax;
35     }
36     return r;
37  }//end eValBrackets
```

Source Code 4.5 Fungsi eValBrackets

f. Fungsi eigVals3

Fungsi ini mengembalikan nilai eigen yang didekati dengan sebuah iteratif *bisection* sebanyak `maxEigVals`. Fungsi ini ditunjukkan oleh Source Code 4.6.

```
1   public double[] eigVals3(int maxEigVals)
2   {
3     double[] eVals = new double[maxEigVals];
4     double[] r;
5     r = eValBrackets(maxEigVals);
6     for (int i = maxEigVals-1; i >=0 ; i--)
7     {
8       eVals[maxEigVals-i-1] = bisect(r[i + 1], r[i]);
9     }
10    return eVals;
11  }//end eigVals3
```

Source Code 4.6 Fungsi untuk Menghitung Nilai Eigen

g. Prosedur powerMethod

Prosedur ini merupakan penerapan dari metode pangkat. Input dari prosedur ini adalah s (hasil eigenvalue shifting), vektor eigen `eigVec` yang diinisialisasi 0, serta nilai eigen `eigVal` yang sebelumnya ditebak menggunakan metode bisection melalui fungsi `eigVals3`. Prosedur ini ditunjukkan oleh Source Code 4.7.

```
1   public void powerMethod(double s, ref double[] eigVec,
2   ref double eigVal)
3   {
4      double tol = 0.000001;
5      double xSign,eVal = 0;
6      int maxIter = 200;
7      int n = diag.Length;
8      double[] x = new double[eigVec.Length];
9      double [] xOld = new double[eigVec.Length];
10     double xMag;
11     double[] tempUpDiag = upDiag;
12     double[] tempDiag = new double[diag.Length];
13     double[] eVec = new double[n];
14     xSign = 0;
15     x = new double[n];
16     tempDiag = diag.Subtract(s);
17     LUdec3(ref tempDiag, ref tempUpDiag);
18     x = mat.randMat(n);
19     upDiag = tempUpDiag;
20
21     xMag = Math.Sqrt(x.InnerProduct(x));
22     x = x.Divide(xMag);
23
24     for (int i = 0; i < maxIter; i++)
25     {
26      xOld = x;
27      x = LUsol3(tempDiag, upDiag, x);
28      xMag = Math.Sqrt(x.InnerProduct(x));
29      x = x.Divide(xMag);
30      xSign = Math.Sign(xOld.InnerProduct(x));
31      x = mat.constmul(x,xSign);
32      double[] cc = xOld.Subtract(x);
33      double cektol=Math.Sqrt((cc.InnerProduct(cc)));
34      if (cektol < tol)
35      {
36        eVal = s + xSign / xMag;
40        eVec = x;
41        break;
42      }
43    }
44    eigVal = eVal;
45    eigVec=eVec;
46  }//end of powerMethod
```

Source Code 4.7 Fungsi untuk Menghitung Nilai Eigen dengan Metode Pangkat

h. Fungsi selisihRata2

Fungsi yang ditunjukkan oleh Source Code 4.8 ini digunakan untuk menghitung selisih antara matriks input awal dengan rata-rata wajah.

```
1   public double[,] selisihRata2(double[,] matinput)
2   {
3      double sum;
4      double[] rata2=new double [matinput.GetLength(0)];
5      double[,] selisih=new double
6          [matinput.GetLength(0),matinput.GetLength(1)];
7      for (int i = 0; i < matinput.GetLength(0); i++)
8      {
9        sum = 0.0;
10       for (int j = 0; j < matinput.GetLength(1); j++)
11       {
12         sum += matinput[i, j];
13       }
14         rata2[i] = sum / matinput.GetLength(1);
15     }
16
17     for (int k = 0; k < matinput.GetLength(0); k++)
18      {
19       for (int l = 0; l < matinput.GetLength(1); l++)
20       {
21         selisih[k, l] = matinput[k, l] - rata2[1];
22       }
23     }
24     average = rata2;
25     return selisih;
26  }
```

Source Code 4.8 Fungsi selisihRata2

i. Fungsi findEigenfaces

Fungsi ini mengembalikan sebuah matriks yang berisi eigenface-eigenface yang dihasilkan dalam proses latihan. Satu kolom matriks mewakili 1 eigenface. Fungsi ini ditunjukkan oleh Source Code 4.9. Hasil dari perhitungan ini akan ditampilkan pada stringgrid.

```
1   public double[,] findEigenfaces(double[,] matinput)
2   {
3      double []eigvec = new double [matrixA.GetLength(0)];
4      double[,] eVecMat = null   double[,] matSelisih;
5      double[] eigvals;
6       double s;
7       CMatrix mat = new CMatrix();
8
9       matSelisih = selisihRata2(matinput);
```

```
10      householder(matrixA);
11      eigvals = eigVals3(matrixA.GetLength(0));
12      eVecMat     =     new     double[matrixA.GetLength(0),
13          matrixA.GetLength(0)];
14
15      for (int i = 0; i < matrixA.GetLength(0); i++)
16      {
17          s = eigvals[i] * 1.0000001;
18          powerMethod(s, ref eigvec, ref eigvals[i]);
19          eVecMat.SetColumn<double>(i, eigvec);
20      }
21      eVecMat = mat.multiply(transform, eVecMat);
22      return eVecMat;
23  }//end findEigenface
24
```

Source Code 4.9 Fungsi findEigenfaces

j. Fungsi bmpEigenface

Fungsi yang ditunjukkan oleh Source Code 4.10 ini mengembalikan wajah-wajah eigen yang bertipe Bitmap.

```
1   public      Bitmap[]      bmpEigenface(double[,]eVecMat,
2   double[,] matinput, int height, int width)
3   {
4      Bitmap[] eigface;
5      double[,]                                          sum=
6       inVectorEigenface(eVecMat,matinput,height,width);
7       eigface = new Bitmap[eVecMat.GetLength(0)];
8       gbr = new CGambar(width, height);
9
10      for (int I = 0; I < eVecMat.GetLength(1); I++)
11      {
12        eigface = gbr.toBitmap(sum, height, width);
13      }
14      return eigface;
15  }
```

Source Code 4.10 Fungsi bmpEigenface

k. Fungsi HitungBobot

Untuk menghitung hasil proyeksi sebuah wajah ke dalam ruang eigen, digunakan fungsi HitungBobot yang ditunjukkan oleh Source Code 4.11. Bobot yang dihasilkan dalam fungsi ini nantinya akan dipergunakan saat proses klasifikasi.

```
1   public   double[]   HitungBobot(double[,]   eigenface,
2   double[] inputFace, double[] avg)
3   {
4      ArrayList listHasil = new ArrayList();
5      double[]hasil=new double [eigenface.GetLength(0)];
6      double[]tempSelisih=mat.subtract1(inputFace, avg);
7
8      for (int i = 0; i < eigenface.GetLength(0); i++)
9       {
10       double[,]                    tempRow                    =
11          (eigenface.GetRow<double>(i).Transpose<double>(
12          )).Transpose<double>();
13       listHasil.Add(mat.multiply(tempRow,
14           tempSelisih.Transpose<double>()));
15     }
16      double[,] tempor;
17      for (int u = 0; u < listHasil.Count; u++)
18      {
19       tempor = (double[,])listHasil[u];
20       hasil[u] = tempor[0, 0];
21     }
22     return hasil;
23  }
```

Source Code 4.11 Fungsi HitungBobot

l. Fungsi findNearestFaceClass

Fungsi ini mengembalikan sebuah nilai yang menunjukkan sebuah kelas wajah yang memiliki jarak paling dekat dengan wajah input. Fungsi ini ditunjukkan oleh Source Code 4.12.

```
1   public   int   findNearestFaceClass(CData   []dataLatih,
2   double[] newImage, double [,] eigenface, double [] avg)
3   {
4      double[,]         bobotEigenface          =          new
5      double[dataLatih.Length, eigenface.GetLength(0)];
6      double[]          bobotNewImg             =          new
7      double[eigenface.GetLength(0)];
8      int nearestIndex =dataLatih.Length;CGambar tempGb;
9
10     //hitung bobot tiap kelas wajah
11     for (int i = 0; i < dataLatih.Length; i++)
12      {
13       double[]tempBobot=new
14            double[eigenface.GetLength(0)];
15       for (int j=0;j<dataLatih[i].FilePath.Length;j++)
16       {
17         //hitung bobotnya
```

```
18          tempGb=new CGambar (dataLatih[i].FilePath[j]);
19
20          tempBobot = mat.add(tempBobot,
21             HitungBobot(eigenface,
22             tempGb.toArray_D(tempGb), avg));
23        }
24             tempBobot =
25          tempBobot.Divide(dataLatih[i].FilePath.Length);
26          bobotEigenface.SetRow<double>(i, tempBobot);
27      }
28       //hitung bobot newImage
29       bobotNewImg = HitungBobot(eigenface, newImage, avg);
30
31       //-----------------hitung threshold------------//
32      //menemukan jarak minimal tiap data latih
33      double max = 0.0; double jarak, tempjarak;
34      double threshold=0.1;
35      for (int l = 0; l < dataLatih.Length; l++)
36       {
40        for (int m=0;m<dataLatih[l].FilePath.Length;m++)
41        {
42          tempGb = new CGambar(dataLatih[l].FilePath[m]);
43          double [] bobot = HitungBobot(eigenface,
44          tempGb.toArray_D(tempGb), avg);
45          jarak = 99999999999999999999.9;
46          for(int n=0;n< bobotEigenface.GetLength(0);n++)
47          {
48            tempjarak      =      nEuclideanDistance(bobot,
49                bobotEigenface.GetRow<double>(n));
50            if (tempjarak < jarak)
51            {
52              jarak = tempjarak;
53            }
54          }//end for n, dapet minimum jarak
55          if (jarak > max)
56          {
57            max = jarak;
58          }
59        }
60        threshold = 0.8*max;
61      }
62
63       jarak = 9999999999999999999.9;
64       tempjarak = 0.0;
65       for(int k =0;k<bobotEigenface.GetLength(0); k++)
66       {
67        tempjarak = nEuclideanDistance(bobotNewImg,
68           bobotEigenface.GetRow<double>(k));
```

```
69
70      if ((tempjarak < jarak) && (tempjarak<threshold))
71      {
72        jarak = tempjarak;
73        nearestIndex = k;
74      }
75    }
76    return nearestIndex;
77  }
78
79
```

Source Code 4.12 Fungsi findNearestFaceClass

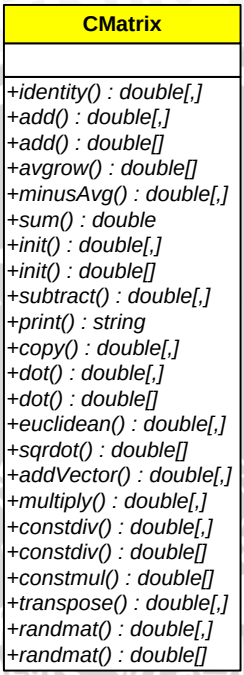
m. Fungsi nEuclideanDistance

Fungsi yang ditunjukkan oleh Source Code 4.13 ini mengembalikan sebuah nilai yang merupakan jarak euclidean dari 2 vektor.

```
1   public double nEuclideanDistance(double[] vec1,
2   double[] vec2)
3   {
4     int N;
5      double hasil, temp;
6      hasil = 0.0;
7      if (vec1.Length == vec2.Length)
8      {
9        N = vec2.Length;
10       temp = 0.0;
11       for (int i = 0; i < N; i++)
12       {
13         temp += Math.Pow((vec1[i] - vec2[i]), 2);
14       }
15       hasil = Math.Sqrt(temp);
16    }
17     return hasil;
18  }
```

Source Code 4.13 Fungsi nEuclideanDistance

4. Kelas CMatrix



Gambar 4.5 *Class Diagram* kelas CMatrix

Kelas CMatrix berisi semua perhitungan yang menyangkut matriks. *Class Diagram* dari kelas ini ditunjukkan oleh Gambar 4.5, sedangkan prosedur/fungsi yang terdapat dalam Kelas CMatrix disajikan pada Tabel 4.3.

Tabel 4.3 Daftar nama Prosedur dan Fungsi Kelas CMatrix

Nama Prosedur/Fungsi	Deskripsi
`public double[,] add(double[,] A, double[,] B)`	Fungsi untuk menghitung penjumlahan antara dua matriks
`public double[] add(double[] A, double[] B)`	Fungsi untuk menghitung penjumlahan antara dua vektor
`public double[] avgrow(double[,] matrix)`	Fungsi untuk menghitung rata-rata matriks pada tiap

	barisnya
`public double[,] minusAvg(double[,] matrix)`	Fungsi untuk menghitung sebuah matriks dikurangi rata-ratanya
`public double sum(double[,] Data)`	Fungsi untuk menghitung jumlah seluruh komponen dari matriks `Data`
`public double[,] init(int row, int col, double n)`	Fungsi untuk menginisialisasi matriks berukuran `row` x `col` dengan bilangan `n`
`public double[] init(int row, double n)`	Fungsi untuk menginisialisasi vektor berukuran 1 x `row` dengan bilangan `n`
`public double[,] subtract(double[,] A, double B)`	Fungsi untuk proses pengurangan matriks
`public double[,] subtract(double[,] A, double[,] B)`	Fungsi untuk proses pengurangan matriks
`public double[,] subtract(double[,] A, double[] B)`	Fungsi untuk proses pengurangan matriks
`public double[,] subtract(double[] A, double[] B)`	Fungsi untuk proses pengurangan matriks
`public double[] subtract1(double[] A, double[] B)`	Fungsi untuk proses pengurangan matriks
`public String print(double[,] data)`	Fungsi untuk mencetak matriks
`public String print(double[] data)`	Fungsi untuk mencetak vektor
`public double[,] copy(double[,] m)`	Fungsi untuk menyalin matriks
`public double[] dot(double[,] A, double[,] B)`	Fungsi perkalian titik antara dua matriks
`public double[] dot(double[] A, double[,] B)`	Fungsi perkalian titik antara vektor dan matriks
`public double[] dot(double[] A, double[] B)`	Fungsi perkalian titik antara vektor dan vektor

`public double[,] dot(double[,] A)`	Fungsi perkalian titik antara matriks dengan dirinya senduru
`public double[,] euclidean(double[,] A)`	Fungsi untuk menghitung jarak euclidean
`public double[] sqrdot(double[,] A, double[,] B)`	Fungsi untuk menghitung akar dari perkalian titik antara 2 matriks
`public double[] sqrdot(double[] A, double[] B)`	Fungsi untuk menghitung akar dari perkalian titik antara 2 vektor
`public double[,] addVector(double[,] n, double[] arr)`	Fungsi menambahkan atau menyisipkan vektor `arr` ke dalam matriks `n`
`public double[,] multiply(double[] a, double[,] b)`	Fungsi untuk mengalikan vektor `a` dengan matriks `b`
`public double[,] multiply(double[,] a, double[] b)`	Fungsi untuk mengalikan matriks `a` dengan vektor `b`
`public double[,] multiply(double[,] matrix1, double[,] matrix2)`	Fungsi perkalian dua matriks
`public double[,] multiply(double[,] Data, double cons)`	Fungsi perkalian matriks dengan sebuah konstanta `cons`
`public double[,] constdiv(double[,] Data, double cons)`	Fungsi pembagian matriks oleh sebuah konstanta `cons`
`public double[] constdiv(double[] Data, double cons)`	Fungsi pembagian vektor dengan sebuah konstanta `cons`
`public double[] constmul(double[] Data, double cons)`	Fungsi perkalian matriks dengan sebuah konstanta `cons`
`public double[,] transpose(double[,]Data)`	Fungsi transpose matriks
`public double[,] transpose(double[] m)`	Fungsi transpose vektor
`public double[,] randMat(int n,int t)`	Fungsi pembangkitan matriks acak dengan dimensi `n` x `t`

`public double[] randMat(int n)`	Fungsi pembangkitan vektor acak dengan dimensi 1x`n`

4.3 Implementasi Uji Coba

Pada subbab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan pada sistem. Untuk mengimplementasikan pengujian, digunakan 2 macam data latih dan 2 data uji. Data Latihan yang digunakan dalam uji coba ini ditunjukkan oleh Tabel 4.4 dan Tabel 4.5. Sedangkan data uji yang digunakan ditunjukkan oleh Tabel 4.6 danTabel 4.7

Tabel 4.4 Data Latihan 1

No.	Nama Data Wajah		
1	anpage.	1	.jpg
2	anpage.	8	.jpg
3	anpage.	9	.jpg
4	anpage.	10	.jpg
5	anpage.	11	.jpg
6	anpage.	12	.jpg
7	anpage.	13	.jpg
8	anpage.	14	.jpg
9	anpage.	15	.jpg
10	anpage.	17	.jpg
11	asamma.	1	.jpg
12	asamma.	2	.jpg
13	asamma.	3	.jpg
14	asamma.	4	.jpg
15	asamma.	5	.jpg
16	asamma.	10	.jpg
17	asamma.	11	.jpg
18	asamma.	12	.jpg
19	asamma.	13	.jpg
20	asamma.	17	.jpg
21	asewil.	1	.jpg
22	asewil.	2	.jpg
23	asewil.	8	.jpg
24	asewil.	9	.jpg
25	asewil.	10	.jpg
26	asewil.	12	.jpg
27	asewil.	13	.jpg
28	asewil.	15	.jpg
29	asewil.	16	.jpg
30	asewil.	17	.jpg
31	astefa.	1	.jpg
32	astefa.	8	.jpg
33	astefa.	9	.jpg
34	astefa.	10	.jpg
35	astefa.	11	.jpg
36	astefa.	12	.jpg
37	astefa.	14	.jpg
38	astefa.	15	.jpg
39	astefa.	18	.jpg
40	astefa.	19	.jpg
41	kclar.	1	.jpg
42	kclar.	3	.jpg
43	kclar.	6	.jpg
44	kclar.	8	.jpg
45	kclar.	9	.jpg
46	kclar.	12	.jpg
47	kclar.	13	.jpg
48	kclar.	15	.jpg
49	kclar.	16	.jpg
50	kclar.	18	.jpg

Tabel 4.5 Data Latihan 2

No.	**Nama Data Wajah**		
1	anpage.	1	.jpg
2	anpage.	2	.jpg
3	anpage.	3	.jpg
4	anpage.	4	.jpg
5	anpage.	5	.jpg
6	anpage.	6	.jpg
7	anpage.	7	.jpg

8	anpage.	8	.jpg
9	anpage.	9	.jpg
10	anpage.	10	.jpg
11	asamma.	1	.jpg
12	asamma.	2	.jpg
13	asamma.	3	.jpg
14	asamma.	4	.jpg
15	asamma.	5	.jpg
16	asamma.	6	.jpg
17	asamma.	7	.jpg
18	asamma.	8	.jpg
19	asamma.	9	.jpg
20	asamma.	10	.jpg
21	asewil.	1	.jpg
22	asewil.	2	.jpg
23	asewil.	3	.jpg
24	asewil.	4	.jpg
25	asewil.	5	.jpg
26	asewil.	6	.jpg
27	asewil.	7	.jpg
28	asewil.	8	.jpg
29	asewil.	9	.jpg
30	asewil.	10	.jpg
31	astefa.	1	.jpg
32	astefa.	2	.jpg
33	astefa.	3	.jpg
34	astefa.	4	.jpg
35	astefa.	5	.jpg
36	astefa.	6	.jpg
37	astefa.	7	.jpg
38	astefa.	8	.jpg
39	astefa.	9	.jpg
40	astefa.	10	.jpg
41	kclar.	1	.jpg
42	kclar.	2	.jpg
43	kclar.	3	.jpg
44	kclar.	4	.jpg
45	kclar.	5	.jpg
46	kclar.	6	.jpg
47	kclar.	7	.jpg
48	kclar.	8	.jpg
49	kclar.	9	.jpg
50	kclar.	10	.jpg

Tabel 4.6 Data Uji 1

No	Nama Data Uji		
1	anpage.	6	.jpg
2	anpage.	16	.jpg
3	anpage.	18	.jpg
4	anpage.	19	.jpg
5	anpage.	20	.jpg
6	asamma.	6	.jpg
7	asamma.	16	.jpg
8	asamma.	18	.jpg
9	asamma.	19	.jpg
10	asamma.	20	.jpg
11	asewil.	3	.jpg
12	asewil.	11	.jpg
13	asewil.	18	.jpg
14	asewil.	19	.jpg
15	asewil.	20	.jpg
16	astefa.	7	.jpg
17	astefa.	10	.jpg
18	astefa.	17	.jpg
19	astefa.	19	.jpg
20	astefa.	20	.jpg
21	kclar.	11	.jpg
22	kclar.	14	.jpg
23	kclar.	17	.jpg
24	kclar.	19	.jpg
25	kclar.	20	.jpg
26	9336923.	1	.jpg
27	9338535.	20	.jpg
28	drbost.	10	.jpg
29	elduns.	12	.jpg
30	kaknig.	14	.jpg
31	ksunth.	11	.jpg
32	lfso.	19	.jpg
33	mbutle.	20	.jpg
34	phuge.	11	.jpg
35	sbains.	18	.jpg
36	slbirc.	14	.jpg
37	vstros.	20	.jpg
38	yfhsie.	11	.jpg

Tabel 4.7 Data Uji 2

No	Nama Data Uji		
1	anpage.	12	.jpg
2	anpage.	14	.jpg
3	anpage.	15	.jpg
4	anpage.	19	.jpg
5	anpage.	20	.jpg
6	asamma.	12	.jpg
7	asamma.	14	.jpg
8	asamma.	17	.jpg
9	asamma.	19	.jpg
10	asamma.	20	.jpg
11	asewil.	12	.jpg
12	asewil.	14	.jpg
13	asewil.	17	.jpg
14	asewil.	19	.jpg
15	asewil.	20	.jpg
16	astefa.	12	.jpg
17	astefa.	14	.jpg
18	astefa.	17	.jpg
19	astefa.	19	.jpg
20	astefa.	20	.jpg
21	kclar.	12	.jpg
22	kclar.	14	.jpg
23	kclar.	17	.jpg
24	kclar.	19	.jpg
25	kclar.	20	.jpg
26	9336923.	1	.jpg
27	9338535.	20	.jpg
28	drbost.	10	.jpg
29	elduns.	12	.jpg
30	kaknig.	14	.jpg
31	ksunth.	11	.jpg
32	lfso.	19	.jpg
33	mbutle.	20	.jpg
34	phuge.	11	.jpg
35	sbains.	18	.jpg
36	slbirc.	14	.jpg
37	vstros.	20	.jpg
38	yfhsie.	11	.jpg

4.4 Hasil Pengujian

4.4.1 Pengujian Data Latih

Dalam pengujian pertama ini dilakukan proses pengenalan dengan menggunakan data latihan sebagai data masukan. Dengan demikian jika data latihan yang digunakan sebanyak 50 wajah, maka data uji juga sejumlah 50 wajah. Tujuan dari pengujian ini adalah untuk mengetahui hubungan antara jumlah data latihan dengan *precision* dan *recall.* Dengan demikian dalam pengujian ini, akan dilakukan 10 kali pengujian dengan jumlah data latihan sebanyak kelipatan 5 mulai dari 5 hingga 50. Karena terdapat 5 individu yang dikategorikan sebagai wajah yang dikenali, maka jika terdapat 5 data latihan berarti 1 wajah untuk setiap kelas wajah, sedangkan jika data latihan sebanyak 10 data latihan berarti 2 wajah untuk setiap kelas wajah, dan seterusnya.

Hasil dari pengujian untuk Data Latihan 1(Tabel 4.4) ditunjukkan pada Lampiran 11 hingga Lampiran 20. Sedangkan Lampiran 25 hingga Lampiran 34 menunjukkan hasil pengujian terhadap Data Latihan 2(Tabel 4.5).

Tabel 4.8 menunjukkan banyaknya kondisi pengenalan wajah (tp, fp, tn, dan fn) untuk tiap Data Latihan 1 dengan jumlah berbeda. berdasarkanTabel 4.8, kemudian dicari *precision* dan *recall*-nya, dan hasilnya terdapat dalam Tabel 4.9. Sedangkan representasinya ditunjukkan pada Gambar 4.6.

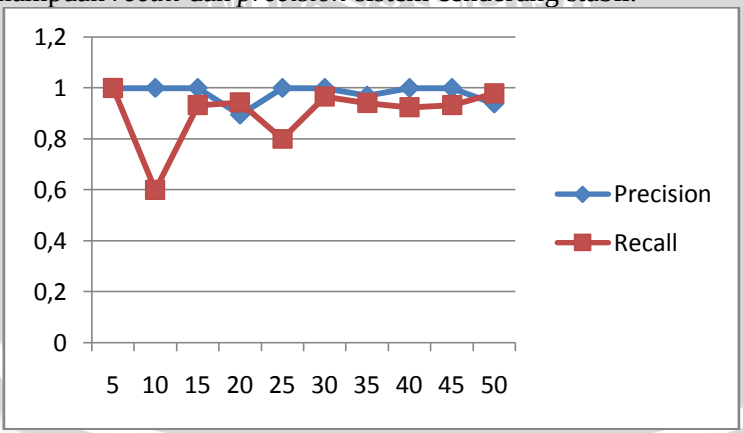
Tabel 4.8 Hasil Pengujian Data Latihan 1

Kondisi	Jumlah Data Latihan									
	5	10	15	20	25	30	35	40	45	50
Tp	5	6	14	17	20	29	32	37	42	46
fp	0	0	0	2	0	0	1	0	0	3
tn	0	0	0	0	0	0	0	0	0	0
fn	0	4	1	1	5	1	2	3	3	1

Tabel 4.9 *Precision* dan *Recall* Hasil Pengujian Data Latihan 1

	Jumlah Data Latihan/Data Uji									
	5	10	15	20	25	30	35	40	45	50
precision	1	1	1	0.89	1	1	0.97	1	1	0.94
Recall	1	0.6	0.93	0.94	0.8	0.97	0.94	0.93	0.93	0.98

Gambar 4.6 secara umum menunjukkan bahwa semakin banyak jumlah data latih, maka kemampuan *precision* dan *recall*-nya semakin baik. Pada saat Data Latihan berjumlah 25, sistem kurang dapat mengenali wajah dengan baik. Banyak di antara wajah yang seharusnya dikenali menjadi tidak dikenali. Namun setelah itu, kemampuan *recall* dan *precision* sistem cenderung stabil.



Gambar 4.6 Grafik *Precision* dan *Recall* untuk Data Latihan 1

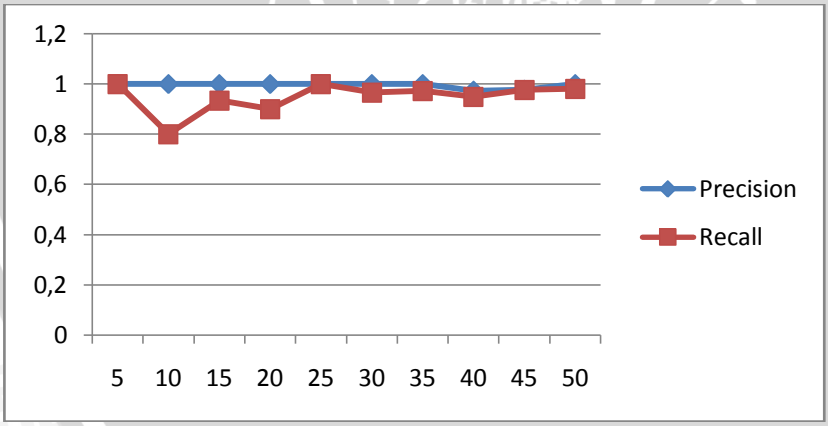
Pengujian yang sama dilakukan terhadap Data Latihan 2. Hasil dari pengujian dapat dilihat padaTabel 4.10 , hasil perhitungan *precision* dan *recall* dapat dilihat pada Tabel 4.11, dan hasil tersebut direpresentasikan dalam bentuk grafik yang ditunjukkan oleh Gambar 4.7.

Tabel 4.10 Hasil Pengujian Data Latihan 2

Kondisi	Jumlah Data Latihan									
	5	10	15	20	25	30	35	40	45	50
tp	5	8	14	18	25	29	34	37	43	49
fp	0	0	0	0	0	0	0	1	1	0
tn	0	0	0	0	0	0	0	0	0	0
fn	0	2	1	2	0	1	1	2	1	1

Tabel 4.11 *Precision* dan *Recall* Hasil Pengujian Data Latihan 2

	Jumlah Data Latihan/Data Uji									
	5	10	15	20	25	30	35	40	45	50
precision	1	1	1	1	1	1	1	1	1	1
recall	1	0.8	0.9	0.9	1	1	1	0.9	1	1



Gambar 4.7 Grafik *Precision* dan *Recall* untuk Data Latihan 2

Pada Gambar 4.7 dapat dilihat bahwa *Precision* untuk Data Latihan 2 setelah jumlah Data Latihan berjumlah 10 selalu

mendekati 1. Ini berarti sistem dapat mengenali hampir setiap wajah yang digunakan sebagai data latihan dengan benar.

4.4.2 Pengujian Banyaknya Jumlah Wajah Eigen

Pengujian ini dilakukan untuk mengetahui hubungan antara jumlah wajah eigen dengan *precision* dan *recall*. Pengujian dilakukan dalam dua tahap. Pada tahap pertama sistem dilatih dengan menggunakan Data Latihan 1 (

Tabel **4.4**) dan diuji menggunakan Data Uji 1 (Tabel 4.6) sebanyak 10 kali dengan jumlah eigenface yang berbeda. Sedangkan pada tahap 2, sistem dilatih dengan menggunakan Data Latihan 2 (Tabel 4.5) dan diuji menggunakan Data Uji 2 (Tabel 4.7) sebanyak 10 kali dengan jumlah eigenface yang berbeda.

Hasil dari pengujian ini secara rinci dapat dilihat pada Lampiran 7 dan Lampiran 8 untuk tahap 1, dan Lampiran 21 dan Lampiran 22 untuk tahap 2.

Tabel 4.12 Hasil Pengujian Data Uji 1 dengan Jumlah Wajah Eigen Berbeda

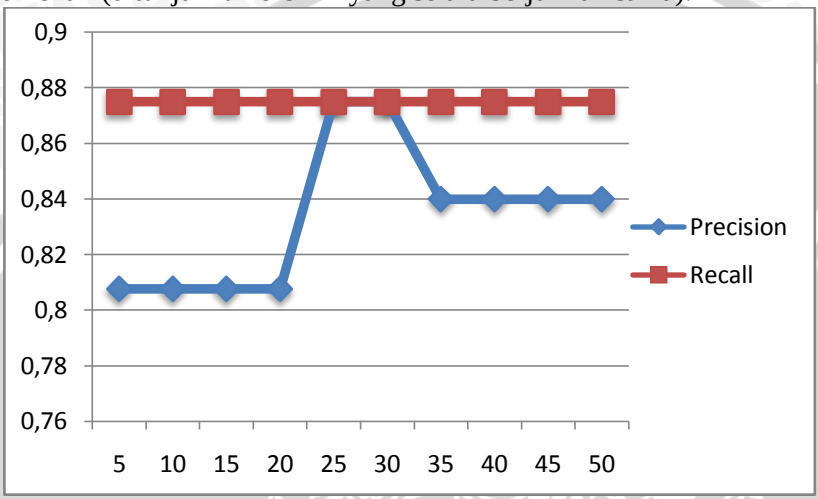
Kondisi	Jumlah Wajah Eigen									
	5	10	15	20	25	30	35	40	45	50
tp	21	21	21	21	21	21	21	21	21	21
fp	4	3	3	3	3	4	4	3	3	3
tn	10	11	11	11	11	10	10	11	11	11
fn	3	3	3	3	3	3	3	3	3	3

Tabel 4.13 *Precison* dan *Recall* Hasil Pengujian Data Uji 1 dengan Jumlah Wajah Eigen Berbeda

	Jumlah Wajah Eigen									
	5	10	15	20	25	30	35	40	45	50
precision	0.84	0.88	0.88	0.88	0.88	0.84	0.84	0.88	0.88	0.88
recall	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88	0.88

Tabel 4.12 menunjukkan hasil pengujian untuk Data Uji 1. Berdasarkan hasil tersebut dapat dilihat bahwa nilai *precision* dan *recall* memiliki nilai yang stabil. Dengan menggunakan wajah eigen sebanyak 5 menghasilkan nilai *precision* dan *recall* yang mendekati nilai *precision* dan *recall* saat seluruh wajah eigen digunakan. Tabel 4.13 direpresentasikan dalam bentuk grafik pada Gambar 4.8.

Pada Gambar 4.8 dapat dilihat bahwa sistem memiliki kemampuan *recall* yang stabil. Hal tersebut disebabkan karena sistem selalu tidak dapat mengenali 3 wajah yang seharusnya dikenali (ditunjukkan oleh fn yang selalu berjumlah sama).



Gambar 4.8 Grafik Hubungan *Precision* dan *Recall* terhadap Jumlah Wajah Eigen pada Data Uji 1

Tabel 4.14 Hasil Pengujian Data Uji 2 dengan Jumlah Wajah Eigen Berbeda

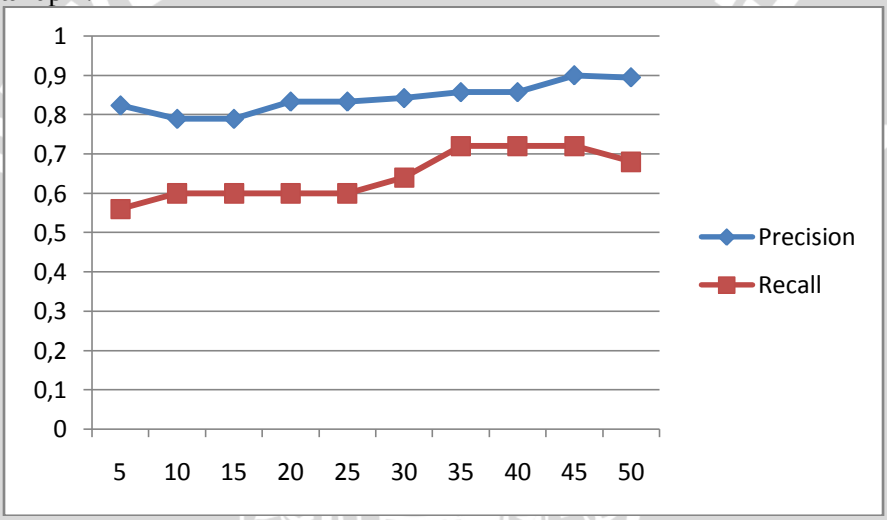
Kondisi	Jumlah Wajah Eigen									
	5	10	15	20	25	30	35	40	45	50
tp	15	17	17	17	17	17	17	17	17	17
fp	2	2	2	2	2	2	2	2	2	2
tn	11	11	11	11	11	11	11	11	11	11
fn	10	8	8	8	8	8	8	8	8	8

Tabel 4.14 dan Tabel 4.15 masing-masing menunjukkan hasil pengujian serta *Precision* dan *Recall* yang diperoleh dengan mengujikan Data Uji 2 dengan data latihan yang berbeda.

Tabel 4.15 *Precision* dan *Recall* Hasil Pengujian Data Uji 2 dengan Jumlah Wajah Eigen Berbeda

	Jumlah Wajah Eigen									
	5	10	15	20	25	30	35	40	45	50
precision	0.88	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89	0.89
Recall	0.60	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68	0.68

Pada pengujian ini dapat dilihat bahwa nilai *precision* dan *recall* memiliki nilai yang sama saat wajah eigen yang digunakan berjumlah 10 hingga 50. Gambar 4.9 merepresentasikan hubungan *precision* dan *recall* terhadap jumlah wajah eigen pada pengujian 2 tahap 2.



Gambar 4.9 Grafik Hubungan *Precision* dan *Recall* terhadap Jumlah Wajah Eigen pada Data Uji 2

4.4.3 Pengujian Banyaknya Jumlah Data Latihan yang Digunakan

Pengujian ini dilakukan untuk mengetahui hubungan antara jumlah data latihan dengan *precision* dan *recall*. Pengujian dilakukan dalam dua tahap. Pada tahap pertama sistem dilatih dengan menggunakan Data Latihan dengan jumlah yang berbeda dan diuji menggunakan Data Uji 1 (Tabel 4.6). Sedangkan pada

tahap 2, sistem dilatih dengan menggunakan Data Latihan dengan jumlah yang berbeda dan diuji menggunakan Data Uji 2 (Tabel 4.7). Hasil dari pengujian ini dapat dilihat pada Lampiran 9 dan Lampiran 10 untuk Data Uji 1, dan Lampiran 23 dan Lampiran 24 untuk Data Uji 2. Kondisi hasil pengujian 3 tahap 1 disajikan pada Tabel 4.8. Sedangkan Tabel 4.17 menunjukkan hasil perhitungan *precision* dan *recall* berdasarkan Tabel 4.8. Representasi hubungan antara *precision* dan *recall* dengan jumlah data latihan ditunjukkan oleh Gambar 4.10.

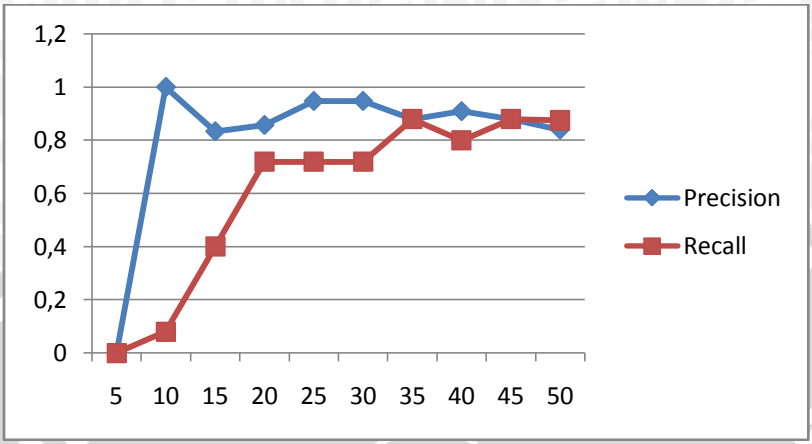
Tabel 4.16 Hasil Pengujian Data Uji 1 dengan Jumlah Data Latihan Berbeda

Kondisi	Jumlah Data Latihan									
	5	10	15	20	25	30	35	40	45	50
tp	0	2	10	18	18	18	22	20	22	21
fp	0	0	2	3	1	1	3	2	3	4
tn	13	13	11	10	12	12	10	11	10	10
fn	25	23	15	7	7	7	3	5	3	3

Tabel 4.17 *Precision* dan *Recall* Hasil Pengujian Data Uji 1 dengan Jumlah Data Latihan Berbeda

	Jumlah Data Latihan									
	5	10	15	20	25	30	35	40	45	50
precision	~	1	0.83	0.86	0.95	0.95	0.88	0.91	0.88	0.84
Recall	0	0.08	0.4	0.72	0.72	0.72	0.88	0.8	0.88	0.88

Berdasarkan Gambar 4.10, dapat dilihat bahwa kemampuan sistem dalam mengenali wajah dengan benar semakin baik seiring bertambahnya jumlah data latihan (ditunjukkan oleh nilai tp yang semakin naik seiring dengan bertambahnya jumlah data latihan). Selain nilai tp, nilai fp juga cenderung naik setelah data latihan berjumlah lebih dari 30. Hal ini berarti sistem salah mengenali sebuah wajah dengan wajah yang identitasnya berbeda. dengan demikian akan mempengaruhi nilai *precision*-nya.



Gambar 4.10 Grafik Hubungan *Precision* dan *Recall* terhadap Jumlah Data Latihan pada Data Uji 1

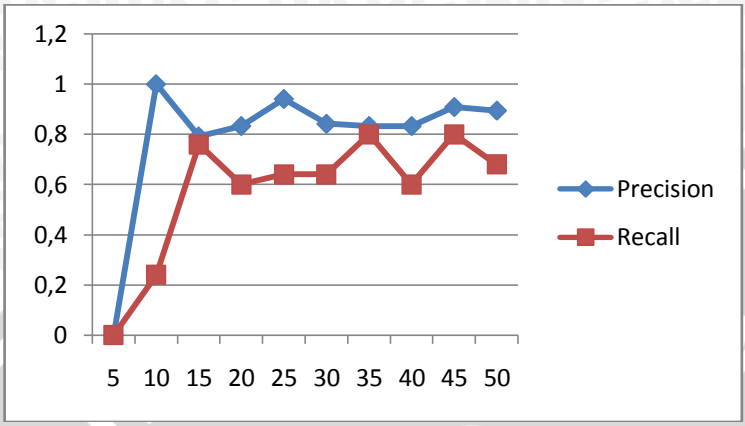
Selanjutnya kondisi hasil pengujian 3 tahap 2 disajikan dalam Tabel 4.18. Sedangkan Tabel 4.19 menunjukkan hasil perhitungan *precision* dan *recall*-nya. Representasi hubungan antara *precision* dan *recall* dengan jumlah data latihan ditunjukkan oleh Gambar 4.11.

Tabel 4.18 Hasil Pengujian Data Uji 2 dengan Jumlah Data Latihan Berbeda

Kondisi	Jumlah Data Latihan									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
tp	0	6	19	15	16	16	20	15	20	17
fp	0	0	5	3	1	3	4	3	2	2
tn	13	13	8	10	12	10	9	10	11	11
fn	25	19	6	10	9	9	5	10	5	8

Tabel 4.19 *Precision* dan *Recall* Hasil Pengujian Data Uji 2 dengan Jumlah Data Latihan Berbeda

	Jumlah Data Latihan									
	5	**10**	**15**	**20**	**25**	**30**	**35**	**40**	**45**	**50**
precision	~	1	0.8	0.8	0.9	0.8	0.8	0.8	0.9	0.9
recall	0	0.2	0.8	0.6	0.6	0.6	0.8	0.6	0.8	0.7



Gambar 4.11 Grafik Hubungan *Precision* dan *Recall* terhadap Jumlah Data Latihan pada Data Uji 2

4.5 Analisis Hasil dan Pembahasan

4.5.1 Kemampuan Sistem dalam Mengenali Kembali Data Latihan

Dari pengujian yang menggunakan data latihan sebagai data input, dapat dilihat bahwa pada saat data latihan yang digunakan berjumlah 5, sistem dapat mengenali kembali kelima data latihan tersebut, sehingga menyebabkan nilai *precision* dan *recall*-nya 1. Tetapi pada saat data latihan yang digunakan adalah 10, sistem tak dapat mengenal kembali seluruh data latihan, sehingga menyebabkan nilai *recall*-nya menjadi turun. Hal ini disebabkan karena pada saat pelatihan dengan data latihan sejumlah 5 (berarti 1 wajah tiap kelas wajah), nilai threshold (Θ) akan bernilai 0. Nilai 0 tersebut diperoleh karena jarak euclidean dihitung terhadap dirinya sendiri.

Pada Tabel 4.20 dapat dilihat bahwa pada saat data wajah pada DataLatihan 1 sejumlah 5 dimasukkan sebagai data input, jarak euclidean bobotInput dan bobotTiap kelas untuk kelas wajah yang sesuai dengan kelas wajah asal data input itu akan selalu bernilai 0.

Tabel 4.20 Jarak bobotInput dan bobotTiapKelas Data Latihan 1 sejumlah 5

No	Input	Kls Wajah	Jarak bobotInput dan bobotTiapKelas
1	anpage1.jpg	1	0.00
		2	78698707.98
		3	151932679.61
		4	36108619.60
		5	9283413.13
2	asamma1.jpg	1	78698707.98
		2	0.00
		3	73248098.59
		4	42940747.26
		5	70346859.58
3	asewil1.jpg	1	151932679.61
		2	73248098.59
		3	0.00
		4	116043454.13
		5	143490417.65
4	astefa1.jpg	1	36108619.60
		2	42940747.26
		3	116043454.13
		4	0.00
		5	27777755.85
5	kclar1.jpg	1	9283413.13
		2	70346859.58
		3	143490417.65
		4	27777755.85
		5	0.00

- Data Latihan : 5 data latih
- Threshold : 0.00
- Jarak minimal data latihan terhadap tiap kelas wajah:

Kelas Wajah	Nama individu	Jarak Minimal
1	anpage	0.00
2	asamma	0.00
3	asewil	0.00
4	astefa	0.00
5	kclar	0.00

Sedangkan pada Tabel 4.21, threshold tidak lagi bernilai 0. Karena setiap kelas wajah memiliki wajah sejumlah 2, maka jarak euclidean tidak lagi dihitung terhadap nilai bobot yang sama,sehingga nilainya tidak mungkin 0. Jika ditemukan jarak terkecil antara bobotInput dan bobotTiapKelas dari suatu kelas tertentu, dan jarak terkecil tersebut bernilai lebih kecil dari threshold, maka wajah input dikategorikan sebagai anggota kelas tersebut. Pada data input no. 1 dan no. 2 pada Tabel 4.21, jarak terkecil wajah input dengan kelas wajah bernilai lebih dari threshold, sehingga dikategikan sebagai *unknown*. Inilah yang menyebabkan nilai *recall* pada saat data latihan berjumlah 10 menjadi turun.

Berdasarkan hasil pengujian ini, dapat disimpulkan bahwa sistem dapat mengenali kembali data latihan dengan cukup baik. Hal ini ditunjukkan oleh nilai *precision* yang berkisar antara 0,8 hingga 1, dan nilai *recall* antara 0,6 hingga 1.

Tabel 4.21Jarak bobotInput dan bobotTiapKelas Data Latihan 1 sejumlah 10

No.	Input	No. Kelas Wajah	Jarak bobotInput dan bobotTiapKelas				
				5	asewil1.jpg	1	135330436.45
						2	92729794.47
1	anpage1.jpg	1	2832809.94			3	824142.37
		2	40859105.64			4	148967028.21
		3	131838333.43			5	160302343.24
		4	17365110.96				
		5	28612469.58	6	asewil 2.jpg	1	133686949.63
						2	91088260.25
2	anpage8.jpg	1	2832809.94			3	824142.37
		2	46017509.23			4	147320071.90
		3	137185540.69			5	158660232.48
		4	13395299.58				
		5	23252628.31	7	astefa 1.jpg	1	16547585.47
						2	58420993.04
3	asamma1.jpg	1	42962523.98			3	149372469.74
		2	468676.37			4	1333341.57
		3	92375158.01			5	15964705.81
		4	56682920.66				
		5	67567796.10		astefa8.jpg	1	13952222.76
				8		2	55873745.61
4	asamma2.jpg	1	43882671.91			3	146916444.08
		2	468676.37			4	1333341.57
		3	91442862.02			5	16889163.90
		4	57609184.15				
		5	68489858.47	9	kclar1.jpg	1	28565463.90

		2	70805214.32	10	kclar3.jpg	1	23300087.07
		3	162253593.31			2	65253516.10
		4	18343410.02			3	156709587.85
		5	2790275.11			4	14685049.15
						5	2790275.11

- Data Latihan : 10 data latih
- Threshold : 2266247,95
- Jarak minimal data latihan terhadap tiap kelas wajah:

No. Kelas Wajah	Nama individu	Jarak Minimal
1	anpage	2832809,94
2	asamma	468676,37
3	asewil	824142,37
4	astefa	1333341,57
5	kclar	2790275,11

4.5.2 Pengaruh Banyaknya Jumlah Wajah Eigen yang Digunakan

Berdasarkan hasil pengujian dengan menggunakan wajah eigen yang berbeda jumlahnya, dapat dilihat bahwa nilai *precision* dan *recall* memiliki nilai yang hampir sama walaupun jumlah wajah eigen yang digunakan berbeda-beda.

Wajah eigen yang memiliki nilai eigen terbesar, menyimpan variasi wajah paling besar (ditunjukkan oleh nilai eigen). Dengan menggunakan beberapa wajah eigen yang memiliki nilai eigen terbesar, maka sistem seharusnya dapat mengenali wajah dengan cukup baik. Grafik pada Gambar 4.8 dan Gambar 4.9 mendukung pernyataan ini. Pada Gambar 4.8 nilai *precision* pada saat wajah eigen berjumlah 5 hingga 50 berkisar antara 0.84 hingga 0.88, sedangkan nilai *recall*-nya tetap sama, yaitu 0.88. Sedangkan pada Gambar 4.9 nilai *precision* dan *recall* saat wajah eigen berjumlah 10 hingga 50 memiliki nilai yang sama. Berdasarkan pengujian ini dapat disimpulkan bahwa pengenalan menggunakan wajah eigen sejumlah 10 (yang nilai eigennya tertinggi) akan menghasilkan hasil yang hampir sama dengan pengenalan menggunakan 50 wajah eigen.

4.5.3 Pengaruh Banyaknya Jumlah Data Latihan yang Digunakan

Berdasarkan hasil pengujian, semakin banyak jumlah data latihan yang digunakan, maka kemampuan sistem untuk mengenali wajah dengan benar semakin meningkat. Hal ini ditunjukkan oleh kondisi tp pada Tabel 4.16 dan Tabel 4.18 yang cenderung meningkat seiring dengan bertambahnya jumlah data latihan. Namun demikian, peningkatan nilai tp diikuti juga dengan peningkatan kondisi fp di mana sistem mengenali seorang individu sebagai individu yang lain, serta peningkatan pada kondisi tn, yaitu di mana sistem mengenali seorang individu yang seharusnya tidak dikenali. Peningkatan nilai fp dan juga tn ini lah yang menyebabkan nilai *precision* dan *recall* menjadi naik dan turun seiring meningkatnya jumlah data latihan.

Sebuah wajah dinyatakan dikenali oleh sistem jika jarak minimal dari wajah tersebut terhadap ruang wajah memiliki nilai lebih kecil dari threshold. Sedangkan nilai threshold sendiri dihitung dari 0,8 kali nilai jarak terbesar dari jarak-jarak minimal dari tiap kelas wajah ke ruang wajah. Jika nilai jarak wajah input minimal yang ditemukan sedikit lebih besar dari threshold, maka wajah tersebut dikategorikan sebagai *unknown*. Lampiran 35 dan Lampiran 36 menunjukkan nilai threshold dan jarak wajah input terhadap ruang wajah.

Kondisi tp (*true positive*) didapatkan jika jarak minimal proyeksi bobotInput dan bobotTiapKelas memiliki nilai yang lebih kecil dari nilai threshold. Kondisi fp (*false positive*) diperoleh jika jarak minimal proyeksi proyeksi bobotInput dan bobotTiapKelas memiliki nilai yang lebih kecil dari nilai threshold, tetapi tidak sesuai dengan kelas wajah yang seharusnya.

Sedangkan kondisi tn (*true negative*) diperoleh jika jarak minimal proyeksi bobotInput dan bobotTiapKelas memiliki nilai lebih besar dari threshold dan dalam kondisi sebenarnya wajah input tersebut memang bukan termasuk individu yang dikenali. Kondisi fp (*false positive*) diperoleh jika nilai proyeksi bobotInput dan bobotTiapKelas lebih besar dari threshold, namun sebenarnya wajahInput tersebut termasuk ke dalam individu yang dikenali.

Penentuan kondisi-kondisi ini adalah dengan mencocokkan antara hasil keluaran sistem dengan kondisi sebearnnya secara manual.

Nilai *precision* dan *recall* sangat bergantung pada kondisi-kondisi ini. Jika nilai fp semakin besar, maka semakin kecil nilai *precision*, dan semakin besar nilai fn, akan semakin kecil nilai *recall*. Dengan demikian dapat disimpulkan bahwa semakin banyak data latihan, kemungkinan sistem dalam mengenali wajah dengan benar semakin baik (ditunjukkan oleh nilai tp yang cenderung naik). Namun demikian kemungkinan wajah dikenali sebagai individu lain (ditunjukkan oleh nilai fp) serta kemungkinan wajah tidak dikenali (ditunjuukan oleh nilai fn) juga semakin besar. Hal ini menyebabkan nilai *precision* dan *recall* menjadi tidak stabil.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh dari penelitian ini adalah:

1. Penerapan algoritma PCA untuk membuat model dari sekumpulan gambar menghasilkan gambar wajah baru yang disebut *eigenface*.
2. PCA merupakan sebuah metode yang dapat digunakan untuk mengenali wajah dengan cukup baik. Dari hasil penelitian didapatkan nilai *precision* dan *recall* untuk pengujian data latih (pada pelatihan menggunakan data latihan sebanyak 50 wajah) berkisar antara 0.94 hingga 1.
3. Pada implementasi PCA untuk pengenalan wajah, data latihan yang digunakan harus lebih dari 1 untuk masing-masing individu (kelas wajah). Jika hanya menggunakan 1 wajah untuk masing-masing individu, maka sistem tidak dapat mengenali wajah lain selain wajah yang dilatihkan. Hal ini disebabkan nilai threshold akan selalu bernilai 0.
4. Semakin banyak data latihan yang digunakan, semakin besar kemungkinan sebuah wajah untuk dikenali dengan benar, namun semakin besar pula kemungkinan wajah dikenali sebagai individu lain atau tidak dikenali. Hal ini menyebabkan nilai *precision* dan *recall* cenderung naik turun.
5. Berdasarkan hasil pengujian pada sistem yang menggunakan 50 data latihan, wajah eigen dengan 10 nilai eigen tertinggi menghasilkan nilai *precision* dan *recall* yang hampir sama dengan proses pengenalan menggunakan 50 wajah eigen.

5.2 Saran

Beberapa saran yang mungkin menjadi pertimbangan adalah sebagai berikut:

1. Dalam penelitian ini perlu dilakukan percobaan terhadap variasi nilai *threshold* agar proses klasifikasi dapat menghasilkan nilai *precision* dan *recall* yang lebih baik.

2. Dilakukan pengujian dengan variasi data uji yang berbeda dengan jumlah data latihan dan jumlah wajah eigen yang sama untuk mengetahui hubungan antara *precision* terhadap *recall*.

DAFTAR PUSTAKA

Brown, William A. 1991. *Matrices and Vector Spaces*, New York: M. Dekker, ISBN 978-0-8247-8419-5.

Delac, Kresimir dan Mislav Grgic. 2007. *Face Recognition*. Vienna: I-Tech Education and Publishing.

Dunteman, George H. 1989. *Principal Component Analysis*. Sage University Paper Series on Quantitative Application in the Social Sciences, series no. 07-069. Newbury Park, CA: Sage.

Gunadi, Kartika dan Sonny Reinard P. *Pembuatan Perangkat Lunak Pengenalan Wajah Menggunakan Principal Component Analysis*. Jurnal Informatika Vol.2 No.2 November 2001:57-61.

Gupta, Sheifali; O.P.Sahoob; Ajay Goel; Rupesh Gupta. 2010. *A New Optimized Approach to Face Recognition using Eigenfaces*. Global Journal of Computer Science and Technology: Vol.10 Issue 1 (Ver 1.0) April 2010.

Joliffe, I. T. 2002. *Principal Component Analysis, Second Edition*. New York: Springer-Verlag, Inc.

Kiusalaas, Jaan. 2009. *Numerical Methods in Engineering with Matlab: Second Edition*. Cambridge University Press.

Li, Stan.Z dan Anil K, Jain. 2005. *Handbook of Face Recognition*. Springer Science+Business Media, Inc.

Leon, Steven J. 2001. *Aljabar Linear dan Aplikasinya*. Jakarta: Penerbit Erlangga.

Manning, Christopher D; Prabhakar Raghavan; Hinrich Schutze. 2008. *Introduction to Information Retrieval*. New York: Cambridge University Press.

Smith, Lindsay I. 2002. *A Tutorial on Principal Component Analysis*. http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf . Diakses pada: 17 Agustus 2010, 9:55.

Turk, Matthew and Alex Pentlad. 1991. *Eigenfaces for Recognition*. Journal of Cognitive Neuroscience Volume 3, Number 1. Massachusetts Institute of Technology.

LAMPIRAN

Lampiran 1 Kelas Wajah anpage


anpage.1.jpg


anpage.2.jpg


anpage.3.jpg


anpage.4.jpg


anpage.5.jpg


anpage.6.jpg


anpage.7.jpg


anpage.8.jpg


anpage.9.jpg


anpage.10.jpg


anpage.11.jpg


anpage.12.jpg


anpage.13.jpg


anpage.14.jpg


anpage.15.jpg


anpage.16.jpg


anpage.17.jpg


anpage.18.jpg


anpage.19.jpg


anpage.20.jpg

Lampiran 2 Kelas Wajah Asamma


asamma.1.jpg


asamma.2.jpg


asamma.3.jpg


asamma.4.jpg


asamma.5.jpg


asamma.6.jpg


asamma.7.jpg


asamma.8.jpg


asamma.9.jpg


asamma.10.jpg


asamma.11.jpg


asamma.12.jpg


asamma.13.jpg


asamma.14.jpg


asamma.15.jpg


asamma.16.jpg


asamma.17.jpg


asamma.18.jpg


asamma.19.jpg


asamma.20.jpg

Lampiran 3 Kelas Wajah asewil


asewil.1.jpg


asewil.2.jpg


asewil.3.jpg


asewil.4.jpg


asewil.5.jpg


asewil.6.jpg


asewil.7.jpg


asewil.8.jpg


asewil.9.jpg


asewil.10.jpg


asewil.11.jpg


asewil.12.jpg


asewil.13.jpg


asewil.14.jpg


asewil.15.jpg


asewil.16.jpg

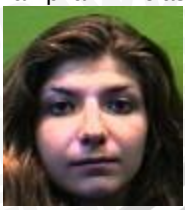

asewil.17.jpg

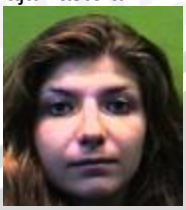

asewil.18.jpg


asewil.19.jpg


asewil.20.jpg

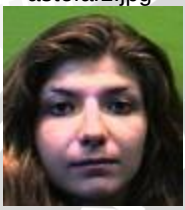
Lampiran 4 Kelas Wajah astefa

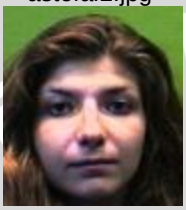

astefa.1.jpg

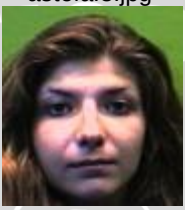

astefa.2.jpg

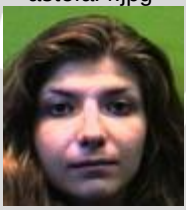

astefa.3.jpg

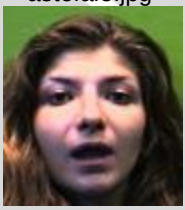

astefa.4.jpg

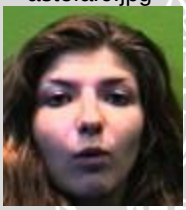

astefa.5.jpg


astefa.6.jpg


astefa.7.jpg

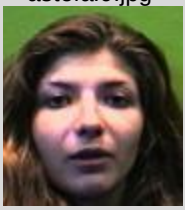

astefa.8.jpg

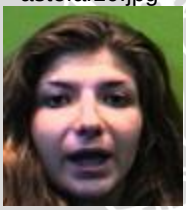

astefa.9.jpg

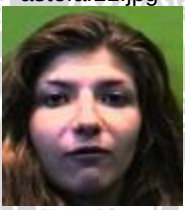

astefa.10.jpg

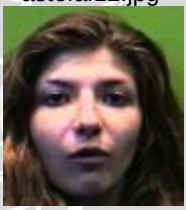

astefa.11.jpg

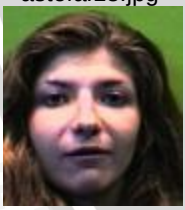

astefa.12.jpg

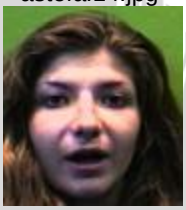

astefa.13.jpg


astefa.14.jpg

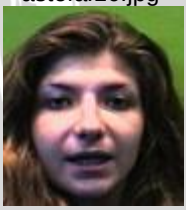

astefa.15.jpg


astefa.16.jpg


astefa.17.jpg


astefa.18.jpg


astefa.19.jpg


astefa.20.jpg

Lampiran 5 Kelas Wajah kclar


kclar.1.jpg


kclar.2.jpg


kclar.3.jpg


kclar.4.jpg


kclar.5.jpg


kclar.6.jpg

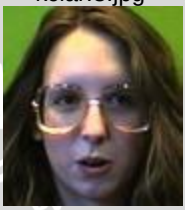

kclar.7.jpg


kclar.8.jpg


kclar.9.jpg


kclar.10.jpg


kclar.11.jpg


kclar.12.jpg


kclar.13.jpg


kclar.14.jpg


kclar.15.jpg


kclar.16.jpg


kclar.17.jpg


kclar.18.jpg


kclar.19.jpg

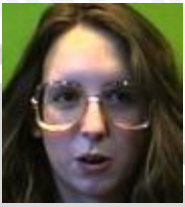

kclar.20.jpg

Lampiran 6 Kelas wajah unknown


9336923.1.jpg


9338535.20.jpg


drbost.10.jpg


elduns.12.jpg


kaknig.14.jpg


ksunth.11.jpg


lfso.19.jpg


mbutle.20.jpg


phuge.11.jpg


sbains.18.jpg


slbirc.14.jpg


vstros.20.jpg


yfhsie.11.jpg

Lampiran 7 Hasil Pengenalan dengan Jumlah Eigenface 50, 45, 40, 35, dan 30

No	Nama Data Uji	Jumlah Eigenface				
		50	45	40	35	30
1	anpage. 6 .jpg	anpage	anpage	anpage	anpage	anpage
2	anpage. 16 .jpg	anpage	anpage	anpage	anpage	anpage
3	anpage. 18 .jpg	anpage	anpage	anpage	anpage	anpage
4	anpage. 19 .jpg	anpage	anpage	anpage	anpage	anpage
5	anpage. 20 .jpg	anpage	anpage	anpage	anpage	anpage
6	asamma. 6 .jpg	asamma	asamma	asamma	asamma	asamma
7	asamma. 16 .jpg	asamma	asamma	asamma	asamma	asamma
8	asamma. 18 .jpg	unknown	unknown	unknown	unknown	unknown
9	asamma. 19 .jpg	unknown	unknown	unknown	unknown	unknown
10	asamma. 20 .jpg	unknown	unknown	unknown	unknown	unknown
11	asewil. 3 .jpg	asewil	asewil	asewil	asewil	asewil
12	asewil. 11 .jpg	asewil	asewil	asewil	asewil	asewil
13	asewil. 18 .jpg	asewil	asewil	asewil	asewil	asewil
14	asewil. 19 .jpg	asewil	asewil	asewil	asewil	asewil
15	asewil. 20 .jpg	asewil	asewil	asewil	asewil	asewil
16	astefa. 7 .jpg	astefa	astefa	astefa	astefa	astefa
17	astefa. 10 .jpg	anpage	anpage	anpage	anpage	anpage
18	astefa. 17 .jpg	astefa	astefa	astefa	astefa	astefa
19	astefa. 19 .jpg	astefa	astefa	astefa	astefa	astefa
20	astefa. 20 .jpg	astefa	astefa	astefa	astefa	astefa
21	kclar. 11 .jpg	kclar	kclar	kclar	kclar	kclar
22	kclar. 14 .jpg	kclar	kclar	kclar	kclar	kclar
23	kclar. 17 .jpg	kclar	kclar	kclar	kclar	kclar
24	kclar. 19 .jpg	kclar	kclar	kclar	kclar	kclar
25	kclar. 20 .jpg	kclar	kclar	kclar	kclar	kclar

26	9336923.	1	.jpg	unknown	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	Unknown	Unknown	Unknown	unknown	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	astefa	astefa	astefa	astefa	astefa
35	sbains.	18	.jpg	asewil	asewil	asewil	asewil	asewil
36	slbirc.	14	.jpg	unknown	unknown	unknown	asamma	asamma
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	unknown	unknown	unknown	unknown

Lampiran 8 Hasil Pengenalan dengan Jumlah Eigenface 25, 20, 15, 10, dan 5

No	Nama Data Uji			Jumlah Eigenface				
				25	20	15	10	5
1	anpage.	6	.jpg	anpage	anpage	anpage	anpage	anpage
2	anpage.	16	.jpg	anpage	anpage	anpage	anpage	anpage
3	anpage.	18	.jpg	anpage	anpage	anpage	anpage	anpage
4	anpage.	19	.jpg	anpage	anpage	anpage	anpage	anpage
5	anpage.	20	.jpg	anpage	anpage	anpage	anpage	anpage
6	asamma.	6	.jpg	asamma	asamma	asamma	asamma	asamma
7	asamma.	16	.jpg	asamma	asamma	asamma	asamma	asamma
8	asamma.	18	.jpg	unknown	unknown	unknown	unknown	unknown
9	asamma.	19	.jpg	unknown	unknown	unknown	unknown	unknown
10	asamma.	20	.jpg	unknown	unknown	unknown	unknown	unknown
11	asewil.	3	.jpg	asewil	asewil	asewil	asewil	asewil

12	asewil.	11	.jpg	asewil	asewil	asewil	asewil	asewil
13	asewil.	18	.jpg	asewil	asewil	asewil	asewil	asewil
14	asewil.	19	.jpg	asewil	asewil	asewil	asewil	asewil
15	asewil.	20	.jpg	asewil	asewil	asewil	asewil	asewil
16	astefa.	7	.jpg	astefa	astefa	astefa	astefa	astefa
17	astefa.	10	.jpg	anpage	anpage	anpage	anpage	anpage
18	astefa.	17	.jpg	astefa	astefa	astefa	astefa	astefa
19	astefa.	19	.jpg	astefa	astefa	astefa	astefa	astefa
20	astefa.	20	.jpg	astefa	astefa	astefa	astefa	astefa
21	kclar.	11	.jpg	kclar	kclar	kclar	kclar	kclar
22	kclar.	14	.jpg	kclar	kclar	kclar	kclar	kclar
23	kclar.	17	.jpg	kclar	kclar	kclar	kclar	kclar
24	kclar.	19	.jpg	kclar	kclar	kclar	kclar	kclar
25	kclar.	20	.jpg	kclar	kclar	kclar	kclar	kclar
26	9336923.	1	.jpg	Unknown	Unknown	Unknown	Unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	unknown	asewil	asewil	asewil
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	astefa	astefa	astefa	astefa	astefa
35	sbains.	18	.jpg	asewil	asewil	asewil	asewil	asewil
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	asamma	asamma	asamma	asamma

Lampiran 9 Hasil Pengenalan dengan Jumlah Data Latihan 50, 45, 40, 35, dan 30

No	Nama Data Uji			Jumlah Data Latihan				
				50	45	40	35	30
1	anpage.	6	.jpg	anpage	anpage	anpage	anpage	anpage
2	anpage.	16	.jpg	anpage	anpage	anpage	anpage	anpage
3	anpage.	18	.jpg	anpage	anpage	anpage	anpage	anpage
4	anpage.	19	.jpg	anpage	anpage	anpage	anpage	anpage
5	anpage.	20	.jpg	anpage	anpage	anpage	anpage	anpage
6	asamma.	6	.jpg	asamma	asamma	asamma	asamma	asamma
7	asamma.	16	.jpg	asamma	asamma	asamma	asamma	unknown
8	asamma.	18	.jpg	unknown	unknown	unknown	unknown	unknown
9	asamma.	19	.jpg	unknown	unknown	unknown	unknown	unknown
10	asamma.	20	.jpg	unknown	unknown	unknown	unknown	unknown
11	asewil.	3	.jpg	asewil	asewil	asewil	asewil	asewil
12	asewil.	11	.jpg	asewil	asewil	asewil	asewil	asewil
13	asewil.	18	.jpg	asewil	asewil	asewil	asewil	asewil
14	asewil.	19	.jpg	asewil	asewil	asewil	asewil	asewil
15	asewil.	20	.jpg	asewil	asewil	asewil	asewil	asewil
16	astefa.	7	.jpg	astefa	astefa	astefa	astefa	astefa
17	astefa.	10	.jpg	anpage	astefa	astefa	astefa	astefa
18	astefa.	17	.jpg	astefa	astefa	unknown	astefa	unknown
19	astefa.	19	.jpg	astefa	astefa	astefa	astefa	unknown
20	astefa.	20	.jpg	astefa	astefa	unknown	astefa	unknown
21	kclar.	11	.jpg	kclar	kclar	kclar	kclar	kclar
22	kclar.	14	.jpg	kclar	kclar	kclar	kclar	kclar
23	kclar.	17	.jpg	kclar	kclar	kclar	kclar	kclar
24	kclar.	19	.jpg	kclar	kclar	kclar	kclar	kclar
25	kclar.	20	.jpg	kclar	kclar	kclar	kclar	kclar

26	9336923.	1	.jpg	unknown	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	unknown	unknown	asewil	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	astefa	astefa	astefa	astefa	astefa
35	sbains.	18	.jpg	asewil	asewil	unknown	asewil	unknown
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	asamma	asamma	unknown	unknown

Lampiran 10 Hasil Pengenalan dengan Jumlah Data Latihan 25, 20, 15, 10, dan 5

No	Nama Data Uji			Jumlah Data Latihan				
				25	20	15	10	5
1	anpage.	6	.jpg	anpage	anpage	anpage	unknown	unknown
2	anpage.	16	.jpg	anpage	anpage	anpage	anpage	unknown
3	anpage.	18	.jpg	anpage	anpage	unknown	unknown	unknown
4	anpage.	19	.jpg	anpage	anpage	anpage	unknown	unknown
5	anpage.	20	.jpg	anpage	anpage	unknown	unknown	unknown
6	asamma.	6	.jpg	unknown	asamma	unknown	unknown	unknown
7	asamma.	16	.jpg	unknown	unknown	unknown	unknown	unknown
8	asamma.	18	.jpg	unknown	unknown	unknown	unknown	unknown
9	asamma.	19	.jpg	unknown	unknown	unknown	unknown	unknown
10	asamma.	20	.jpg	unknown	unknown	unknown	unknown	unknown
11	asewil.	3	.jpg	asewil	asewil	asewil	unknown	unknown

12	asewil.	11	.jpg	asewil	asewil	unknown	unknown	unknown
13	asewil.	18	.jpg	asewil	asewil	asewil	unknown	unknown
14	asewil.	19	.jpg	asewil	asewil	asewil	unknown	unknown
15	asewil.	20	.jpg	asewil	asewil	asewil	unknown	unknown
16	astefa.	7	.jpg	astefa	unknown	astefa	astefa	unknown
17	astefa.	10	.jpg	astefa	astefa	astefa	unknown	unknown
18	astefa.	17	.jpg	unknown	unknown	unknown	unknown	unknown
19	astefa.	19	.jpg	astefa	astefa	unknown	unknown	unknown
20	astefa.	20	.jpg	unknown	astefa	unknown	unknown	unknown
21	kclar.	11	.jpg	kclar	kclar	unknown	unknown	unknown
22	kclar.	14	.jpg	kclar	unknown	unknown	unknown	unknown
23	kclar.	17	.jpg	kclar	kclar	unknown	unknown	unknown
24	kclar.	19	.jpg	kclar	kclar	kclar	unknown	unknown
25	kclar.	20	.jpg	kclar	kclar	unknown	unknown	unknown
26	9336923.	1	.jpg	unknown	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	asewil	asewil	unknown	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	unknown	astefa	unknown	unknown	unknown
35	sbains.	18	.jpg	asewil	asewil	asewil	unknown	unknown
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	unknown	unknown	unknown	unknown

Lampiran 11 Hasil Pengenalan Data Latihan sebanyak 50 terhadap dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	1	.jpg	anpage
2	anpage.	8	.jpg	anpage
3	anpage.	9	.jpg	anpage
4	anpage.	10	.jpg	anpage
5	anpage.	11	.jpg	anpage
6	anpage.	12	.jpg	anpage
7	anpage.	13	.jpg	anpage
8	anpage.	14	.jpg	anpage
9	anpage.	15	.jpg	anpage
10	anpage.	17	.jpg	anpage
11	asamma.	1	.jpg	asamma
12	asamma.	2	.jpg	asamma
13	asamma.	3	.jpg	asamma
14	asamma.	4	.jpg	asamma
15	asamma.	5	.jpg	asamma
16	asamma.	10	.jpg	asamma
17	asamma.	11	.jpg	asamma
18	asamma.	12	.jpg	asamma
19	asamma.	13	.jpg	asamma
20	asamma.	17	.jpg	unknown
21	asewil.	1	.jpg	asewil
22	asewil.	2	.jpg	asewil
23	asewil.	8	.jpg	asewil
24	asewil.	9	.jpg	asewil
25	asewil.	10	.jpg	asewil
26	asewil.	12	.jpg	asewil
27	asewil.	13	.jpg	asewil
28	asewil.	15	.jpg	asewil
29	asewil.	16	.jpg	asewil
30	asewil.	17	.jpg	asewil
31	astefa.	1	.jpg	astefa
32	astefa.	8	.jpg	astefa
33	astefa.	9	.jpg	anpage
34	astefa.	10	.jpg	anpage
35	astefa.	11	.jpg	anpage
36	astefa.	12	.jpg	astefa
37	astefa.	14	.jpg	astefa
38	astefa.	15	.jpg	astefa
39	astefa.	18	.jpg	astefa
40	astefa.	19	.jpg	astefa
41	kclar.	1	.jpg	kclar
42	kclar.	3	.jpg	kclar
43	kclar.	6	.jpg	kclar
44	kclar.	8	.jpg	kclar
45	kclar.	9	.jpg	kclar
46	kclar.	12	.jpg	kclar
47	kclar.	13	.jpg	kclar
48	kclar.	15	.jpg	kclar
49	kclar.	16	.jpg	kclar
50	kclar.	18	.jpg	kclar

Lampiran 12 Hasil Pengenalan Data Latihan sebanyak 45 terhadap dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	1	.jpg	anpage
2	anpage.	8	.jpg	anpage
3	anpage.	9	.jpg	anpage
4	anpage.	10	.jpg	anpage
5	anpage.	11	.jpg	anpage
6	anpage.	12	.jpg	anpage
7	anpage.	13	.jpg	anpage
8	anpage.	14	.jpg	anpage
9	anpage.	15	.jpg	anpage
10	asamma.	1	.jpg	asamma
11	asamma.	2	.jpg	asamma
12	asamma.	3	.jpg	asamma
13	asamma.	4	.jpg	asamma
14	asamma.	5	.jpg	asamma
15	asamma.	10	.jpg	asamma
16	asamma.	11	.jpg	asamma
17	asamma.	12	.jpg	asamma
18	asamma.	13	.jpg	asamma
19	asewil.	1	.jpg	asewil
20	asewil.	2	.jpg	asewil
21	asewil.	8	.jpg	asewil
22	asewil.	9	.jpg	asewil
23	asewil.	10	.jpg	asewil
24	asewil.	12	.jpg	asewil
25	asewil.	13	.jpg	asewil
26	asewil.	15	.jpg	asewil
27	asewil.	16	.jpg	asewil

28	astefa.	1	.jpg	astefa
29	astefa.	8	.jpg	astefa
30	astefa.	9	.jpg	unknown
31	astefa.	10	.jpg	unknown
32	astefa.	11	.jpg	astefa
33	astefa.	12	.jpg	astefa
34	astefa.	14	.jpg	astefa
35	astefa.	15	.jpg	astefa
36	astefa.	18	.jpg	unknown
37	kclar.	1	.jpg	kclar
38	kclar.	3	.jpg	kclar
39	kclar.	6	.jpg	kclar
40	kclar.	8	.jpg	kclar
41	kclar.	9	.jpg	kclar
42	kclar.	12	.jpg	kclar
43	kclar.	13	.jpg	kclar
44	kclar.	15	.jpg	kclar
45	kclar.	16	.jpg	kclar

Lampiran 13 Hasil Pengenalan Data Latihan sebanyak 40 terhadap dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	1	.jpg	anpage
2	anpage.	8	.jpg	anpage
3	anpage.	9	.jpg	anpage
4	anpage.	10	.jpg	anpage
5	anpage.	11	.jpg	anpage
6	anpage.	12	.jpg	anpage
7	anpage.	13	.jpg	anpage
8	anpage.	14	.jpg	anpage
9	asamma.	1	.jpg	asamma
10	asamma.	2	.jpg	asamma
11	asamma.	3	.jpg	asamma
12	asamma.	4	.jpg	asamma
13	asamma.	5	.jpg	asamma
14	asamma.	10	.jpg	asamma
15	asamma.	11	.jpg	asamma
16	asamma.	12	.jpg	asamma
17	asewil.	1	.jpg	asewil
18	asewil.	2	.jpg	asewil
19	asewil.	8	.jpg	asewil
20	asewil.	9	.jpg	asewil
21	asewil.	10	.jpg	asewil
22	asewil.	12	.jpg	asewil
23	asewil.	13	.jpg	asewil
24	asewil.	15	.jpg	asewil
25	astefa.	1	.jpg	astefa
26	astefa.	8	.jpg	astefa
27	astefa.	9	.jpg	unknown
28	astefa.	10	.jpg	astefa
29	astefa.	11	.jpg	astefa
30	astefa.	12	.jpg	astefa
31	astefa.	14	.jpg	unknown
32	astefa.	15	.jpg	unknown
33	kclar.	1	.jpg	kclar
34	kclar.	3	.jpg	kclar
35	kclar.	6	.jpg	kclar
36	kclar.	8	.jpg	kclar
37	kclar.	9	.jpg	kclar
38	kclar.	12	.jpg	kclar
39	kclar.	13	.jpg	kclar
40	kclar.	15	.jpg	kclar

Lampiran 14 Hasil Pengenalan Data Latihan sebanyak 35 terhadap dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 1 .jpg	anpage
2	anpage. 8 .jpg	anpage
3	anpage. 9 .jpg	anpage
4	anpage. 10 .jpg	anpage
5	anpage. 11 .jpg	anpage
6	anpage. 12 .jpg	anpage
7	anpage. 13 .jpg	anpage
8	asamma. 1 .jpg	asamma
9	asamma. 2 .jpg	asamma
10	asamma. 3 .jpg	asamma
11	asamma. 4 .jpg	asamma
12	asamma. 5 .jpg	asamma
13	asamma. 10 .jpg	asamma
14	asamma. 11 .jpg	unknown
15	asewil. 1 .jpg	asewil
16	asewil. 2 .jpg	asewil
17	asewil. 8 .jpg	asewil
18	asewil. 9 .jpg	asewil
19	asewil. 10 .jpg	asewil
20	asewil. 12 .jpg	asewil
21	asewil. 13 .jpg	asewil
22	astefa. 1 .jpg	astefa
23	astefa. 8 .jpg	unknown
24	astefa. 9 .jpg	anpage
25	astefa. 10 .jpg	astefa
26	astefa. 11 .jpg	astefa
27	astefa. 12 .jpg	astefa
28	astefa. 14 .jpg	astefa
29	kclar. 1 .jpg	kclar
30	kclar. 3 .jpg	kclar
31	kclar. 6 .jpg	kclar
32	kclar. 8 .jpg	kclar
33	kclar. 9 .jpg	kclar
34	kclar. 12 .jpg	kclar
35	kclar. 13 .jpg	kclar

Lampiran 15 Hasil Pengenalan Data Latihan sebanyak 30 terhadap dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 1 .jpg	anpage
2	anpage. 8 .jpg	anpage
3	anpage. 9 .jpg	anpage
4	anpage. 10 .jpg	anpage
5	anpage. 11 .jpg	anpage
6	anpage. 12 .jpg	anpage
7	asamma. 1 .jpg	asamma
8	asamma. 2 .jpg	asamma
9	asamma. 3 .jpg	asamma
10	asamma. 4 .jpg	asamma
11	asamma. 5 .jpg	asamma
12	asamma. 10 .jpg	asamma
13	asewil. 1 .jpg	asewil
14	asewil. 2 .jpg	asewil
15	asewil. 8 .jpg	asewil
16	asewil. 9 .jpg	asewil
17	asewil. 10 .jpg	asewil
18	asewil. 12 .jpg	asewil
19	astefa. 1 .jpg	astefa
20	astefa. 8 .jpg	astefa
21	astefa. 9 .jpg	unknown
22	astefa. 10 .jpg	astefa
23	astefa. 11 .jpg	astefa
24	astefa. 12 .jpg	astefa
25	kclar. 1 .jpg	kclar
26	kclar. 3 .jpg	kclar
27	kclar. 6 .jpg	kclar
28	kclar. 8 .jpg	kclar
29	kclar. 9 .jpg	kclar
30	kclar. 12 .jpg	kclar

Lampiran 16 Hasil Pengenalan Data Latihan sebanyak 25 terhadap dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 1 .jpg	unknown
2	anpage. 8 .jpg	anpage
3	anpage. 9 .jpg	anpage
4	anpage. 10 .jpg	unknown
5	anpage. 11 .jpg	anpage
6	asamma. 1 .jpg	asamma
7	asamma. 2 .jpg	asamma
8	asamma. 3 .jpg	asamma
9	asamma. 4 .jpg	asamma
10	asamma. 5 .jpg	asamma
11	asewil. 1 .jpg	asewil
12	asewil. 2 .jpg	asewil
13	asewil. 8 .jpg	asewil
14	asewil. 9 .jpg	asewil
15	asewil. 10 .jpg	asewil
16	astefa. 1 .jpg	unknown
17	astefa. 8 .jpg	unknown
18	astefa. 9 .jpg	unknown
19	astefa. 10 .jpg	astefa
20	astefa. 11 .jpg	astefa
21	kclar. 1 .jpg	kclar
22	kclar. 3 .jpg	kclar
23	kclar. 6 .jpg	kclar
24	kclar. 8 .jpg	kclar
25	kclar. 9 .jpg	kclar

Lampiran 17 Hasil Pengenalan Data Latihan sebanyak 20 terhadap dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 1 .jpg	astefa
2	anpage. 8 .jpg	anpage
3	anpage. 9 .jpg	anpage
4	anpage. 10 .jpg	anpage
5	asamma. 1 .jpg	asamma
6	asamma. 2 .jpg	asamma
7	asamma. 3 .jpg	asamma
8	asamma. 4 .jpg	asamma
9	asewil. 1 .jpg	asewil
10	asewil. 2 .jpg	asewil
11	asewil. 8 .jpg	asewil
12	asewil. 9 .jpg	asewil
13	astefa. 1 .jpg	astefa
14	astefa. 8 .jpg	unknown
15	astefa. 9 .jpg	anpage
16	astefa. 10 .jpg	astefa
17	kclar. 1 .jpg	kclar
18	kclar. 3 .jpg	kclar
19	kclar. 6 .jpg	kclar
20	kclar. 8 .jpg	kclar

Lampiran 18 Hasil Pengenalan Data Latihan sebanyak 15 terhadap dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 1 jpg	anpage
2	anpage. 8 jpg	anpage
3	anpage. 9 jpg	anpage
4	asamma. 1 jpg	asamma
5	asamma. 2 jpg	asamma
6	asamma. 3 jpg	asamma
7	asewil. 1 jpg	asewil
8	asewil. 2 jpg	asewil
9	asewil. 8 jpg	asewil
10	astefa. 1 jpg	astefa
11	astefa. 8 jpg	astefa
12	astefa. 9 jpg	unknown
13	kclar. 1 jpg	kclar
14	kclar. 3 jpg	kclar
15	kclar. 6 jpg	kclar

Lampiran 19 Hasil Pengenalan Data Latihan sebanyak 10 terhadap dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 1 .jpg	unknown
2	anpage. 8 .jpg	unknown
3	asamma. 1 .jpg	asamma
4	asamma. 2 .jpg	asamma
5	asewil. 1 .jpg	asewil
6	asewil. 2 .jpg	asewil
7	astefa. 1 .jpg	astefa

8	astefa.	8	.jpg	astefa
9	kclar.	1	.jpg	unknown
10	kclar.	3	.jpg	unknown

Lampiran 20 Hasil Pengenalan Data Latihan sebanyak 5 terhadap dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	1	jpg	anpage
2	asamma.	1	jpg	asamma
3	asewil.	1	jpg	asewil
4	astefa.	1	jpg	astefa
5	kclar.	1	jpg	kclar

Lampiran 21 Hasil Pengujian dengan Jumlah Eigenface 50, 45, 40, 35, dan 30

No	Nama Data Uji	Jumlah Eigenface				
		50	45	40	35	30
1	anpage. 12 .jpg	anpage	anpage	anpage	anpage	anpage
2	anpage. 14 .jpg	anpage	anpage	anpage	anpage	anpage
3	anpage. 15 .jpg	anpage	anpage	anpage	anpage	anpage
4	anpage. 19 .jpg	anpage	anpage	anpage	anpage	anpage
5	anpage. 20 .jpg	anpage	anpage	anpage	anpage	anpage
6	asamma. 12 .jpg	unknown	unknown	unknown	unknown	unknown
7	asamma. 14 .jpg	unknown	unknown	unknown	unknown	unknown
8	asamma. 17 .jpg	unknown	unknown	unknown	unknown	unknown
9	asamma. 19 .jpg	unknown	unknown	unknown	unknown	unknown
10	asamma. 20 .jpg	unknown	unknown	unknown	unknown	unknown
11	asewil. 12 .jpg	asewil	asewil	asewil	asewil	asewil
12	asewil. 14 .jpg	asewil	asewil	asewil	asewil	asewil
13	asewil. 17 .jpg	asewil	asewil	asewil	asewil	asewil
14	asewil. 19 .jpg	asewil	asewil	asewil	asewil	asewil
15	asewil. 20 .jpg	asewil	asewil	asewil	asewil	asewil
16	astefa. 12 .jpg	astefa	astefa	astefa	astefa	astefa
17	astefa. 14 .jpg	unknown	astefa	astefa	astefa	astefa
18	astefa. 17 .jpg	unknown	unknown	unknown	unknown	unknown
19	astefa. 19 .jpg	astefa	astefa	astefa	astefa	astefa
20	astefa. 20 .jpg	unknown	unknown	unknown	unknown	unknown
21	kclar. 12 .jpg	kclar	kclar	kclar	kclar	kclar
22	kclar. 14 .jpg	kclar	kclar	kclar	kclar	kclar
23	kclar. 17 .jpg	kclar	kclar	kclar	kclar	kclar
24	kclar. 19 .jpg	kclar	kclar	kclar	kclar	kclar
25	kclar. 20 .jpg	kclar	kclar	kclar	kclar	kclar

26	9336923.	1	.jpg	unknown	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	unknown	unknown	unknown	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	astefa	astefa	astefa	astefa	astefa
35	sbains.	18	.jpg	asewil	asewil	asewil	asewil	asewil
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	unknown	unknown	unknown	unknown

Lampiran 22 Hasil Pengujian dengan Jumlah Eigenface 25, 20, 15, 10, dan 5

No	Nama Data Uji			Jumlah Eigenface				
				25	20	15	10	5
1	anpage.	12	.jpg	anpage	anpage	anpage	anpage	anpage
2	anpage.	14	.jpg	anpage	anpage	anpage	anpage	anpage
3	anpage.	15	.jpg	anpage	anpage	anpage	anpage	unknown
4	anpage.	19	.jpg	anpage	anpage	anpage	anpage	anpage
5	anpage.	20	.jpg	unknown	unknown	unknown	unknown	unknown
6	asamma.	12	.jpg	unknown	unknown	unknown	unknown	unknown
7	asamma.	14	.jpg	unknown	unknown	unknown	unknown	unknown
8	asamma.	17	.jpg	unknown	unknown	unknown	unknown	unknown
9	asamma.	19	.jpg	unknown	unknown	unknown	unknown	unknown
10	asamma.	20	.jpg	unknown	unknown	unknown	unknown	unknown
11	asewil.	12	.jpg	asewil	asewil	asewil	asewil	asewil

12	asewil.	14	.jpg	asewil	asewil	asewil	asewil	asewil
13	asewil.	17	.jpg	asewil	asewil	asewil	asewil	asewil
14	asewil.	19	.jpg	asewil	asewil	asewil	asewil	asewil
15	asewil.	20	.jpg	asewil	asewil	asewil	asewil	asewil
16	astefa.	12	.jpg	astefa	astefa	astefa	astefa	unknown
17	astefa.	14	.jpg	unknown	unknown	unknown	unknown	unknown
18	astefa.	17	.jpg	unknown	unknown	unknown	unknown	unknown
19	astefa.	19	.jpg	astefa	astefa	astefa	astefa	astefa
20	astefa.	20	.jpg	unknown	unknown	unknown	unknown	unknown
21	kclar.	12	.jpg	kclar	kclar	kclar	kclar	kclar
22	kclar.	14	.jpg	kclar	kclar	kclar	kclar	kclar
23	kclar.	17	.jpg	kclar	kclar	kclar	kclar	kclar
24	kclar.	19	.jpg	kclar	kclar	kclar	kclar	kclar
25	kclar.	20	.jpg	kclar	kclar	kclar	kclar	kclar
26	9336923.	1	.jpg	unknown	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	unknown	unknown	unknown	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	astefa	astefa	astefa	astefa	astefa
35	sbains.	18	.jpg	asewil	asewil	asewil	asewil	asewil
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	unknown	asamma	asamma	asamma

Lampiran 23 Hasil Pengujian dengan Jumlah Data Latihan 50, 45, 40, 35, dan 30

No	Nama Data Uji			Jumlah Data Latihan				
				50	45	40	35	30
1	anpage.	12	.jpg	anpage	anpage	anpage	anpage	anpage
2	anpage.	14	.jpg	anpage	anpage	anpage	anpage	anpage
3	anpage.	15	.jpg	anpage	anpage	anpage	anpage	anpage
4	anpage.	19	.jpg	anpage	anpage	anpage	anpage	anpage
5	anpage.	20	.jpg	anpage	anpage	kclar	anpage	astefa
6	asamma.	12	.jpg	unknown	unknown	unknown	unknown	unknown
7	asamma.	14	.jpg	unknown	unknown	unknown	unknown	unknown
8	asamma.	17	.jpg	unknown	unknown	unknown	unknown	unknown
9	asamma.	19	.jpg	unknown	unknown	unknown	unknown	unknown
10	asamma.	20	.jpg	unknown	unknown	unknown	unknown	unknown
11	asewil.	12	.jpg	asewil	asewil	asewil	asewil	asewil
12	asewil.	14	.jpg	asewil	asewil	asewil	asewil	asewil
13	asewil.	17	.jpg	asewil	asewil	asewil	asewil	asewil
14	asewil.	19	.jpg	asewil	asewil	asewil	asewil	asewil
15	asewil.	20	.jpg	asewil	asewil	asewil	asewil	asewil
16	astefa.	12	.jpg	astefa	astefa	unknown	astefa	anpage
17	astefa.	14	.jpg	unknown	astefa	unknown	astefa	unknown
18	astefa.	17	.jpg	unknown	astefa	unknown	astefa	unknown
19	astefa.	19	.jpg	astefa	astefa	unknown	astefa	unknown
20	astefa.	20	.jpg	unknown	astefa	unknown	astefa	unknown
21	kclar.	12	.jpg	kclar	kclar	kclar	kclar	kclar
22	kclar.	14	.jpg	kclar	kclar	kclar	kclar	kclar
23	kclar.	17	.jpg	kclar	kclar	kclar	kclar	kclar
24	kclar.	19	.jpg	kclar	kclar	kclar	kclar	kclar
25	kclar.	20	.jpg	kclar	kclar	kclar	kclar	kclar

26	9336923.	1	.jpg	astefa	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	astefa	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	unknown	unknown	asewil	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	anpage	astefa	astefa	astefa	anpage
35	sbains.	18	.jpg	unknown	asewil	unknown	asewil	unknown
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	unknown	unknown	unknown
38	yfhsie.	11	.jpg	unknown	unknown	unknown	unknown	unknown

Lampiran 24 Hasil Pengujian dengan Jumlah Data Latihan 25, 20, 15, 10, dan 5

No	Nama Data Uji			Jumlah Data Latihan				
				25	20	15	10	5
1	anpage.	12	.jpg	anpage	anpage	anpage	anpage	unknown
2	anpage.	14	.jpg	anpage	anpage	anpage	anpage	unknown
3	anpage.	15	.jpg	anpage	anpage	anpage	unknown	unknown
4	anpage.	19	.jpg	anpage	anpage	anpage	anpage	unknown
5	anpage.	20	.jpg	anpage	anpage	anpage	unknown	unknown
6	asamma.	12	.jpg	unknown	unknown	unknown	unknown	unknown
7	asamma.	14	.jpg	unknown	unknown	unknown	unknown	unknown
8	asamma.	17	.jpg	unknown	unknown	unknown	unknown	unknown
9	asamma.	19	.jpg	unknown	unknown	unknown	unknown	unknown
10	asamma.	20	.jpg	unknown	unknown	unknown	unknown	unknown

11	asewil.	12	.jpg	asewil	asewil	asewil	asewil	unknown
12	asewil.	14	.jpg	asewil	asewil	asewil	unknown	unknown
13	asewil.	17	.jpg	asewil	asewil	asewil	unknown	unknown
14	asewil.	19	.jpg	asewil	asewil	asewil	asewil	unknown
15	asewil.	20	.jpg	asewil	asewil	asewil	asewil	unknown
16	astefa.	12	.jpg	astefa	unknown	astefa	unknown	unknown
17	astefa.	14	.jpg	unknown	unknown	astefa	unknown	unknown
18	astefa.	17	.jpg	unknown	unknown	astefa	unknown	unknown
19	astefa.	19	.jpg	unknown	unknown	astefa	unknown	unknown
20	astefa.	20	.jpg	unknown	unknown	unknown	unknown	unknown
21	kclar.	12	.jpg	kclar	kclar	anpage	unknown	unknown
22	kclar.	14	.jpg	kclar	kclar	anpage	unknown	unknown
23	kclar.	17	.jpg	kclar	kclar	kclar	unknown	unknown
24	kclar.	19	.jpg	kclar	kclar	kclar	unknown	unknown
25	kclar.	20	.jpg	kclar	kclar	kclar	unknown	unknown
26	9336923.	1	.jpg	unknown	unknown	unknown	unknown	unknown
27	9338535.	20	.jpg	unknown	unknown	unknown	unknown	unknown
28	drbost.	10	.jpg	unknown	unknown	unknown	unknown	unknown
29	elduns.	12	.jpg	unknown	unknown	unknown	unknown	unknown
30	kaknig.	14	.jpg	unknown	unknown	unknown	unknown	unknown
31	ksunth.	11	.jpg	unknown	asewil	asewil	unknown	unknown
32	lfso.	19	.jpg	unknown	unknown	unknown	unknown	unknown
33	mbutle.	20	.jpg	unknown	unknown	unknown	unknown	unknown
34	phuge.	11	.jpg	unknown	astefa	astefa	unknown	unknown
35	sbains.	18	.jpg	asewil	asewil	asewil	unknown	unknown
36	slbirc.	14	.jpg	unknown	unknown	unknown	unknown	unknown
37	vstros.	20	.jpg	unknown	unknown	kclar	unknown	unknown
38	yfhsie.	11	.jpg	unknown	unknown	unknown	unknown	unknown

Lampiran 25 Hasil Pengenalan dengan Data Latih 50 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	1	.jpg	anpage
2	anpage.	2	.jpg	anpage
3	anpage.	3	.jpg	anpage
4	anpage.	4	.jpg	anpage
5	anpage.	5	.jpg	anpage
6	anpage.	6	.jpg	anpage
7	anpage.	7	.jpg	anpage
8	anpage.	8	.jpg	anpage
9	anpage.	9	.jpg	anpage
10	anpage.	10	.jpg	anpage
11	asamma.	1	.jpg	asamma
12	asamma.	2	.jpg	asamma
13	asamma.	3	.jpg	asamma
14	asamma.	4	.jpg	asamma
15	asamma.	5	.jpg	asamma
16	asamma.	6	.jpg	asamma
17	asamma.	7	.jpg	asamma
18	asamma.	8	.jpg	asamma
19	asamma.	9	.jpg	asamma
20	asamma.	10	.jpg	asamma
21	asewil.	1	.jpg	asewil
22	asewil.	2	.jpg	asewil
23	asewil.	3	.jpg	asewil
24	asewil.	4	.jpg	asewil
25	asewil.	5	.jpg	asewil
26	asewil.	6	.jpg	asewil
27	asewil.	7	.jpg	asewil
28	asewil.	8	.jpg	asewil
29	asewil.	9	.jpg	asewil
30	asewil.	10	.jpg	asewil
31	astefa.	1	.jpg	astefa
32	astefa.	2	.jpg	astefa
33	astefa.	3	.jpg	astefa
34	astefa.	4	.jpg	astefa
35	astefa.	5	.jpg	astefa
36	astefa.	6	.jpg	astefa
37	astefa.	7	.jpg	astefa
38	astefa.	8	.jpg	astefa
39	astefa.	9	.jpg	unknown
40	astefa.	10	.jpg	astefa
41	kclar.	1	.jpg	kclar
42	kclar.	2	.jpg	kclar
43	kclar.	3	.jpg	kclar
44	kclar.	4	.jpg	kclar
45	kclar.	5	.jpg	kclar
46	kclar.	6	.jpg	kclar
47	kclar.	7	.jpg	kclar
48	kclar.	8	.jpg	kclar
49	kclar.	9	.jpg	kclar
50	kclar.	10	.jpg	kclar

Lampiran 26 Hasil Pengenalan dengan Data Latih 45 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	2	.jpg	anpage
2	anpage.	3	.jpg	anpage
3	anpage.	4	.jpg	anpage
4	anpage.	5	.jpg	anpage
5	anpage.	6	.jpg	anpage
6	anpage.	7	.jpg	anpage
7	anpage.	8	.jpg	anpage
8	anpage.	9	.jpg	anpage
9	anpage.	10	.jpg	anpage
10	asamma.	2	.jpg	asamma
11	asamma.	3	.jpg	asamma
12	asamma.	4	.jpg	asamma
13	asamma.	5	.jpg	asamma
14	asamma.	6	.jpg	asamma
15	asamma.	7	.jpg	asamma
16	asamma.	8	.jpg	asamma
17	asamma.	9	.jpg	asamma
18	asamma.	10	.jpg	asamma
19	asewil.	2	.jpg	asewil
20	asewil.	3	.jpg	asewil
21	asewil.	4	.jpg	asewil
22	asewil.	5	.jpg	asewil
23	asewil.	6	.jpg	asewil
24	asewil.	7	.jpg	asewil
25	asewil.	8	.jpg	asewil
26	asewil.	9	.jpg	asewil
27	asewil.	10	.jpg	asewil
28	astefa.	2	.jpg	astefa
29	astefa.	3	.jpg	astefa
30	astefa.	4	.jpg	astefa
31	astefa.	5	.jpg	astefa
32	astefa.	6	.jpg	astefa
33	astefa.	7	.jpg	astefa
34	astefa.	8	.jpg	astefa
35	astefa.	9	.jpg	anpage
36	astefa.	10	.jpg	unknown
37	kclar.	2	.jpg	kclar
38	kclar.	3	.jpg	kclar
39	kclar.	4	.jpg	kclar
40	kclar.	5	.jpg	kclar
41	kclar.	6	.jpg	kclar
42	kclar.	7	.jpg	kclar
43	kclar.	8	.jpg	kclar
44	kclar.	9	.jpg	kclar
45	kclar.	10	.jpg	kclar

Lampiran 27 Hasil Pengenalan dengan Data Latih 40 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	3	.jpg	anpage
2	anpage.	4	.jpg	anpage
3	anpage.	5	.jpg	anpage
4	anpage.	6	.jpg	anpage
5	anpage.	7	.jpg	anpage
6	anpage.	8	.jpg	anpage
7	anpage.	9	.jpg	anpage
8	anpage.	10	.jpg	kclar
9	asamma.	3	.jpg	unknown
10	asamma.	4	.jpg	asamma
11	asamma.	5	.jpg	asamma
12	asamma.	6	.jpg	asamma
13	asamma.	7	.jpg	asamma
14	asamma.	8	.jpg	asamma
15	asamma.	9	.jpg	asamma
16	asamma.	10	.jpg	asamma
17	asewil.	3	.jpg	asewil
18	asewil.	4	.jpg	asewil
19	asewil.	5	.jpg	asewil
20	asewil.	6	.jpg	asewil
21	asewil.	7	.jpg	asewil
22	asewil.	8	.jpg	asewil
23	asewil.	9	.jpg	asewil
24	asewil.	10	.jpg	asewil
25	astefa.	3	.jpg	astefa
26	astefa.	4	.jpg	astefa
27	astefa.	5	.jpg	astefa
28	astefa.	6	.jpg	astefa
29	astefa.	7	.jpg	astefa
30	astefa.	8	.jpg	astefa
31	astefa.	9	.jpg	unknown
32	astefa.	10	.jpg	astefa
33	kclar.	3	.jpg	kclar
34	kclar.	4	.jpg	kclar

35	kclar.	5	.jpg	kclar
36	kclar.	6	.jpg	kclar
37	kclar.	7	.jpg	kclar
38	kclar.	8	.jpg	kclar
39	kclar.	9	.jpg	kclar
40	kclar.	10	.jpg	kclar

Lampiran 28 Hasil Pengenalan dengan Data Latih 35 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	4	.jpg	anpage
2	anpage.	5	.jpg	anpage
3	anpage.	6	.jpg	anpage
4	anpage.	7	.jpg	anpage
5	anpage.	8	.jpg	anpage
6	anpage.	9	.jpg	anpage
7	anpage.	10	.jpg	anpage
8	asamma.	4	.jpg	asamma
9	asamma.	5	.jpg	asamma
10	asamma.	6	.jpg	asamma
11	asamma.	7	.jpg	asamma
12	asamma.	8	.jpg	asamma
13	asamma.	9	.jpg	asamma
14	asamma.	10	.jpg	asamma
15	asewil.	4	.jpg	asewil
16	asewil.	5	.jpg	asewil
17	asewil.	6	.jpg	asewil
18	asewil.	7	.jpg	asewil
19	asewil.	8	.jpg	asewil
20	asewil.	9	.jpg	asewil
21	asewil.	10	.jpg	asewil
22	astefa.	4	.jpg	astefa
23	astefa.	5	.jpg	astefa
24	astefa.	6	.jpg	astefa
25	astefa.	7	.jpg	astefa
26	astefa.	8	.jpg	astefa
27	astefa.	9	.jpg	unknown
28	astefa.	10	.jpg	astefa
29	kclar.	4	.jpg	kclar
30	kclar.	5	.jpg	kclar
31	kclar.	6	.jpg	kclar
32	kclar.	7	.jpg	kclar
33	kclar.	8	.jpg	kclar
34	kclar.	9	.jpg	kclar
35	kclar.	10	.jpg	kclar

Lampiran 29 Hasil Pengenalan dengan Data Latih 30 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	5	.jpg	anpage
2	anpage.	6	.jpg	anpage
3	anpage.	7	.jpg	anpage
4	anpage.	8	.jpg	anpage
5	anpage.	9	.jpg	anpage
6	anpage.	10	.jpg	anpage
7	asamma.	5	.jpg	asamma
8	asamma.	6	.jpg	asamma
9	asamma.	7	.jpg	asamma
10	asamma.	8	.jpg	asamma
11	asamma.	9	.jpg	asamma
12	asamma.	10	.jpg	asamma
13	asewil.	5	.jpg	asewil
14	asewil.	6	.jpg	asewil
15	asewil.	7	.jpg	asewil
16	asewil.	8	.jpg	asewil
17	asewil.	9	.jpg	asewil
18	asewil.	10	.jpg	asewil
19	astefa.	5	.jpg	astefa
20	astefa.	6	.jpg	astefa
21	astefa.	7	.jpg	astefa
22	astefa.	8	.jpg	astefa
23	astefa.	9	.jpg	unknown
24	astefa.	10	.jpg	astefa
25	kclar.	5	.jpg	kclar
26	kclar.	6	.jpg	kclar
27	kclar.	7	.jpg	kclar
28	kclar.	8	.jpg	kclar
29	kclar.	9	.jpg	kclar
30	kclar.	10	.jpg	kclar

Lampiran 30 Hasil Pengenalan dengan Data Latih 25 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	6	.jpg	anpage
2	anpage.	7	.jpg	anpage

3	anpage.	8	.jpg	anpage
4	anpage.	9	.jpg	anpage
5	anpage.	10	.jpg	anpage
6	asamma.	6	.jpg	asamma
7	asamma.	7	.jpg	asamma
8	asamma.	8	.jpg	asamma
9	asamma.	9	.jpg	asamma
10	asamma.	10	.jpg	asamma
11	asewil.	6	.jpg	asewil
12	asewil.	7	.jpg	asewil
13	asewil.	8	.jpg	asewil
14	asewil.	9	.jpg	asewil
15	asewil.	10	.jpg	asewil
16	astefa.	6	.jpg	astefa
17	astefa.	7	.jpg	astefa
18	astefa.	8	.jpg	astefa
19	astefa.	9	.jpg	astefa
20	astefa.	10	.jpg	astefa
21	kclar.	6	.jpg	kclar
22	kclar.	7	.jpg	kclar
23	kclar.	8	.jpg	kclar
24	kclar.	9	.jpg	kclar
25	kclar.	10	.jpg	kclar

Lampiran 31 Hasil Pengenalan dengan Data Latih 20 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	7	.jpg	anpage
2	anpage.	8	.jpg	anpage
3	anpage.	9	.jpg	anpage
4	anpage.	10	.jpg	anpage
5	asamma.	7	.jpg	asamma
6	asamma.	8	.jpg	asamma
7	asamma.	9	.jpg	asamma
8	asamma.	10	.jpg	asamma
9	asewil.	7	.jpg	asewil
10	asewil.	8	.jpg	asewil
11	asewil.	9	.jpg	asewil
12	asewil.	10	.jpg	asewil
13	astefa.	7	.jpg	astefa
14	astefa.	8	.jpg	unknown
15	astefa.	9	.jpg	unknown
16	astefa.	10	.jpg	astefa
17	kclar.	7	.jpg	kclar
18	kclar.	8	.jpg	kclar
19	kclar.	9	.jpg	kclar
20	kclar.	10	.jpg	kclar

Lampiran 32 Hasil Pengenalan dengan Data Latih 15 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	8	.jpg	anpage
2	anpage.	9	.jpg	anpage
3	anpage.	10	.jpg	anpage
4	asamma.	8	.jpg	asamma
5	asamma.	9	.jpg	asamma
6	asamma.	10	.jpg	asamma
7	asewil.	8	.jpg	asewil
8	asewil.	9	.jpg	asewil
9	asewil.	10	.jpg	asewil
10	astefa.	8	.jpg	unknown
11	astefa.	9	.jpg	astefa
12	astefa.	10	.jpg	astefa
13	kclar.	8	.jpg	kclar
14	kclar.	9	.jpg	kclar
15	kclar.	10	.jpg	kclar

Lampiran 33 Hasil Pengenalan dengan Data Latih 10 dengan dirinya sendiri

No.	Nama Data Wajah			Dikenali sebagai
1	anpage.	9	.jpg	anpage
2	anpage.	10	.jpg	anpage
3	asamma.	9	.jpg	asamma
4	asamma.	10	.jpg	asamma
5	asewil.	9	.jpg	asewil
6	asewil.	10	.jpg	asewil
7	astefa.	9	.jpg	unknown
8	astefa.	10	.jpg	unknown
9	kclar.	9	.jpg	kclar
10	kclar.	10	.jpg	kclar

Lampiran 34 Hasil Pengenalan dengan Data Latih 5 dengan dirinya sendiri

No.	Nama Data Wajah	Dikenali sebagai
1	anpage. 9 .jpg	anpage
2	asamma. 9 .jpg	asamma
3	asewil. 9 .jpg	asewil
4	astefa. 9 .jpg	astefa
5	kclar. 9 .jpg	kclar

:

Lampiran 35 Jarak bobotInput dan bobotTiapKelas Data Latihan 1 sejumlah 10, diuji terhadap Data Uji 1

No.	Input	Kelas Wajah	Jarak bobotInput dan bobotTiapKelas
1	anpage6.jpg	1	441771.41
		2	43268332.87
		3	134349080.49
		4	15540625.07
		5	26099465.72
2	anpage16.jpg	1	6155056.15
		2	49154900.73
		3	140436768.74
		4	11102009.43
		5	19820561.27
3	anpage18.jpg	1	7259187.34
		2	50259650.79
		3	141564514.49
		4	10364684.76
		5	18737785.21
4	anpage19.jpg	1	4881466.19
		2	48021438.14
		3	139269515.36
		4	11644644.90
		5	21115863.51
5	anpage20.jpg	1	8488384.81
		2	51388349.39
		3	142725248.03
		4	9822670.61
		5	17480450.87
6	asamma6.jpg	1	36380987.06
		2	7129355.72
		3	99004289.99
		4	50099452.15
		5	60930340.17
7	asamma16.jpg	1	32272350.39
		2	11206852.10
		3	102972663.38
		4	45992291.64
		5	56921922.31
8	asamma18.jpg	1	25838714.10
		2	17683633.89
		3	109329994.77
		4	39529859.26
		5	50580273.83
9	asamma19.jpg	1	20387660.83
		2	23206248.65
		3	114813029.14
		4	34056485.47
		5	45111620.05
10	asamma20.jpg	1	22553732.33
		2	20978893.05
		3	112382677.44
		4	36279964.77
		5	47563259.11
11	asewil3.jpg	1	131393778.71
		2	88793109.86
		3	3121754.09
		4	145027034.60
		5	156364052.17
12	asewil11.jpg	1	127230438.81
		2	84585378.57
		3	7444336.84
		4	140890420.62
		5	152139129.24
13	asewil18.jpg	1	131293883.87
		2	88652297.51
		3	3406147.92
		4	144947653.45
		5	156218753.66

14	asewil19.jpg	1	129313204.76			3	152183031.50
		2	86666574.81			4	9749789.52
		3	5384357.65			5	7863583.09
		4	142972020.19				
		5	154226640.26	23	kclar17.jpg	1	20974671.09
						2	63547235.63
15	asewil20.jpg	1	130140187.82			3	155009471.90
		2	87498224.35			4	11155854.21
		3	4544749.38			5	5459643.09
		4	143796163.81				
		5	155060318.73	24	kclar19.jpg	1	20723302.02
						2	63298016.68
16	astefa7.jpg	1	15002617.91			3	154741592.89
		2	57023800.01			4	11265210.52
		3	148061060.82			5	5512367.93
		4	452168.43				
		5	16148936.66	25	kclar20.jpg	1	21771366.23
						2	64358118.04
17	astefa10.jpg	1	9948584.71			3	155806047.93
		2	52174254.49			4	11892045.46
		3	143517486.50			5	4667475.71
		4	7108812.57				
		5	17138858.41	26	9336923.1.jpg	1	13618521.12
						2	38507976.34
18	astefa17.jpg	1	5501860.69			3	129602332.53
		2	38885578.49			4	23963493.11
		3	130061225.18			5	30953053.86
		4	18750536.62				
		5	30043362.34	27	9338535.20.jpg	1	54611028.08
						2	12763295.68
19	astefa19.jpg	1	3863962.86			3	80351167.38
		2	44038697.00			4	68300084.95
		3	135197775.05			5	79237719.18
		4	13501751.81				
		5	25398297.97	28	drbost.10.jpg	1	70279673.51
						2	27660301.31
20	astefa20.jpg	1	4834209.18			3	64577223.57
		2	43854495.22			4	84017228.26
		3	134836057.49			5	94983900.71
		4	13451326.04				
		5	26378141.88	29	elduns.12.jpg	1	17974511.63
						2	25684919.43
21	kclar11.jpg	1	19011427.23			3	116615620.18
		2	61346230.57			4	32193449.61
		3	152843871.65			5	43130822.85
		4	10364986.74				
		5	7139403.35	30	kaknig.14.jpg	1	43586362.29
						2	85452190.18
22	kclar14.jpg	1	18268218.19			3	177035994.31
		2	60717684.50			4	31593337.69

		5	18140031.56
31	ksunth.11.jpg	1	135746075.28
		2	93200179.81
		3	3534615.80
		4	149392081.26
		5	160826991.87
32	lfso.19.jpg	1	93799872.73
		2	51674364.48
		3	40907829.14
		4	107409940.36
		5	119015202.35
33	mbutle.20.jpg	1	91026210.31
		2	48699731.63
		3	43555337.70
		4	104673291.72
		5	116159867.81
34	phuge.11.jpg	1	3222614.73
		2	44548642.77
		3	135799315.86
		4	13218300.81
		5	24617283.19
35	sbains.18.jpg	1	131350696.49
		2	88833100.19
		3	3947327.07
		4	144978401.86
		5	156418106.68
36	slbirc.14.jpg	1	69874163.82
		2	28152947.38
		3	64727131.78
		4	83476066.87
		5	95108433.33
37	vstros.20.jpg	1	41773688.77
		2	82781807.28
		3	174165536.03
		4	30595623.85
		5	16584929.51
38	yfhsie.11.jpg	1	29684972.49
		2	14432704.53
		3	105084535.29
		4	43214479.29
		5	54861302.34

- Data Latihan : 10 data latih
- Threshold : 2266247,95
- Jarak minimal data latihan terhadap tiap kelas wajah:

Kelas Wajah	Nama individu	Jarak Minimal
1	Anpage	2832809,94
2	Asamma	468676,37
3	Asewil	824142,37
4	Astefa	1333341,57
5	kclar	2790275,11

Lampiran 36 Jarak bobotInput dan bobotTiapKelas Data Latihan 1 sejumlah 15, diuji terhadap Data Uji 1

No.	Input	Kelas Wajah	Jarak bobotInput dan bobotTiapKelas
1	anpage6.jpg	1	402117.83
		2	89727582.75
		3	221600522.94
		4	29683884.63
		5	11488954.78
2	anpage16.jpg	1	1848844.11
		2	91142415.88
		3	223031829.46
		4	31080773.11
		5	10040918.09
3	anpage18.jpg	1	3212619.43
		2	92539818.67
		3	224438565.45
		4	32354583.56
		5	8736870.72
4	anpage19.jpg	1	1356915.04
		2	90601746.77
		3	222508635.36
		4	30400295.27
		5	10696693.90
5	anpage20.jpg	1	2881325.55
		2	92182476.59
		3	224085018.50

		4	32005660.70				
		5	9039661.30	14	Asewil19.jpg	1	213634260.84
						2	124475778.31
6	asamma6.jpg	1	80336641.94			3	7796698.69
		2	9133457.63			4	186868447.57
		3	141210187.77			5	224946064.17
		4	53730520.10				
		5	91724236.79	15	Asewil20.jpg	1	215056887.38
						2	125892858.78
7	asamma16.jpg	1	75136989.00			3	6370192.39
		2	14439666.17			4	188280250.79
		3	146465835.57			5	226370313.70
		4	48420644.15				
		5	86551248.69	16	Astefa7.jpg	1	42696490.66
						2	53203271.54
8	Asamma18.jpg	1	67956832.22			3	183703805.40
		2	21769463.93			4	13878419.88
		3	153738656.75			5	54023619.42
		4	41097245.77				
		5	79406049.80	17	Astefa10.jpg	1	13491224.72
						2	77168233.18
9	Asamma19.jpg	1	59630539.68			3	208808501.62
		2	30128375.30			4	18610897.35
		3	162107744.18			5	24671084.34
		4	32794887.81				
		5	71101066.63	18	Astefa17.jpg	1	32180384.09
						2	57781424.99
10	Asamma20.jpg	1	65488495.58			3	189386243.51
		2	24394602.77			4	11921709.18
		3	156280368.25			5	43606525.20
		4	38473944.05				
		5	76966078.53	19	Astefa19.jpg	1	30589109.37
						2	59383963.44
11	Asewil3.jpg	1	219309417.13			3	191120881.60
		2	130102358.58			4	8406550.98
		3	2009875.43			5	42157622.10
		4	192440830.12				
		5	230639202.26	20	Astefa20.jpg	1	40462473.28
						2	50252659.63
12	Asewil11.jpg	1	210459265.41			3	181896020.34
		2	121305826.97			4	12750370.50
		3	10938102.67			5	52083761.64
		4	183700333.63				
		5	221770966.36	21	Kclar11.jpg	1	12640644.51
						2	101908551.34
13	Asewil18.jpg	1	216885275.19			3	233833787.39
		2	127718214.97			4	41151658.65
		3	4585139.24			5	3129530.43
		4	190103782.10				
		5	228198925.79	22	Kclr14.jpg	1	10618539.66

		2	99863576.45			4	76780825.35
		3	231794021.59			5	37562601.50
		4	39070561.21				
		5	3376524.06	31	ksunth.11.jpg	1	222945200.12
						2	133745135.06
23	Kclar17.jpg	1	13978265.30			3	2114298.03
		2	103245396.77			4	196137507.35
		3	235180882.48			5	234266911.73
		4	42304092.79				
		5	4301630.37	32	lfso.19.jpg	1	156283914.29
						2	67240695.35
24	kclar19.jpg	1	11727480.10			3	65031873.56
		2	101005541.66			4	129615066.18
		3	232928581.73			5	167623248.68
		4	40248829.10				
		5	2978092.81	33	mbutle.20.jpg	1	161137995.29
						2	71891805.77
25	kclar20.jpg	1	13972632.46			3	60254469.46
		2	103258284.46			4	134215518.40
		3	235185177.83			5	172507133.57
		4	42389238.72				
		5	4044481.09	34	phuge.11.jpg	1	15464401.64
						2	74614911.71
26	9336923.1.jpg	1	57000278.57			3	206548550.51
		2	58302080.36			4	14008677.67
		3	153650787.08			5	27051462.75
		4	36305313.97				
		5	65481800.45	35	sbains.18.jpg	1	223027984.89
						2	133790820.58
27	9338535.20.jpg	1	36836986.58			3	2369242.91
		2	52724409.71			4	196105013.40
		3	184597283.80			5	234369243.76
		4	14293640.40				
		5	48141760.47	36	slbirc.14.jpg	1	127939336.62
						2	38802007.50
28	drbost.10.jpg	1	113623591.96			3	93475690.86
		2	24605960.82			4	100994668.32
		3	107743545.08			5	139343931.14
		4	86832947.70				
		5	124998891.22	37	vstros.20.jpg	1	38741795.39
						2	127971525.27
29	elduns.12.jpg	1	39159308.39			3	259903234.88
		2	50420515.60			4	66652502.37
		3	182153824.21			5	27509939.27
		4	16681255.54				
		5	50520544.23	38	yfhsie.11.jpg	1	80185825.06
						2	11223856.35
30	kaknig.14.jpg	1	48831899.41			3	141791397.41
		2	138128183.54			4	52725780.35
		3	270049449.06			5	91693784.92

- Data Latihan : 15 data latih
- Threshold : 12321876,6679277
- Jarak minimal data latihan terhadap tiap kelas wajah:

Kelas Wajah	Nama individu	Jarak Minimal
1	Anpage	1049579,65
2	Asamma	447596,94
3	Asewil	3693239,64
4	Astefa	8472244,30
5	kclar	1377661,02

Keterangan warna:

Jarak euclidean terkecil <=θ

Jarak euclidean terkecil > θ

Jarak euclidean terkecil >θ karena merupakan individu yang tak dikenal

Jarak euclidean terkecil <=θ, hasil pengenalan salah