

**RANCANG BANGUN APLIKASI PENGENALAN
WAJAH MENGGUNAKAN METODE *EIGENFACE*
YANG BERORIENTASI PADA *PRINCIPAL
COMPONENT ANALYSIS (PCA)***

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Informatika

Oleh :

SUTRISNO AFFANDI
10451026435



FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU
PEKANBARU
2011

RANCANG BANGUN APLIKASI PENGENALAN WAJAH MENGUNAKAN METODE *EIGENFACE* YANG BERORIENTASI PADA *PRINCIPAL COMPONENT ANALYSIS (PCA)*

**SUTRISNO AFFANDI
NIM : 10451026435**

Tanggal Sidang : 22 Juni 2011
Tanggal Wisuda : November 2011

Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Sultan Syarif Kasim Riau
Jl. Soebrantas No. 155 Pekanbaru

ABSTRAK

Kemampuan manusia dalam mengenali wajah merupakan kemampuan yang luar biasa. Dengan kemampuan ini, manusia mampu mengenali sampai ribuan wajah dan masih bisa mengingat dan mengenalinya walaupun setelah bertahun-tahun kemudian. Kemampuan dalam mengenali wajah yang diimplementasikan pada suatu alat atau sistem akan memberikan banyak manfaat pada kehidupan saat ini. Berbagai aplikasi dari alat dengan kemampuan seperti ini terbentang luas dari pencarian penjahat, kriminalitas, sistem akses keruangan, sampai interaksi manusia dengan komputer.

Dalam tugas akhir ini akan dilaksanakan perancangan sistem pengenalan identitas manusia dengan identifikasi wajah manusia sebagai media pengenalnya atau yang lebih dikenal sebagai *face recognition*. *Face recognition* sendiri merupakan suatu cabang ilmu *biometric*, yaitu suatu bidang keilmuan yang menggunakan karakteristik fisik dari seseorang untuk menentukan atau mengungkapkan identitasnya. Metode pengenalan yang dipakai adalah metode *eigenface*. *Eigenface* menggunakan metode *Principal Component Analysis (PCA)* yaitu suatu metode matematika untuk merepresentasikan sebuah objek, mengekstraksi ciri-ciri sebuah objek dan mereduksi sebuah objek dengan cara mentransformasikannya menggunakan *eigenvalue* dan *eigenvector* secara linier.

Kata Kunci: *Biometric, Eigenface, Face Recognition, Principal Component Analysis (PCA)*.

DAFTAR ISI

	Halaman
LEMBAR PERSETUJUAN.....	ii
LEMBAR PENGESAHAN	iii
LEMBARAN HAK ATAS KEKAYAAN INTELEKTUAL.....	iv
LEMBARAN PERNYATAAN	v
LEMBARAN PERSEMBAHAN	vi
ABSTRAK	vii
<i>ABSTRACT</i>	viii
KATA PENGANTAR	ix
DAFTAR ISI.....	xi
DAFTAR GAMBAR	xv
DAFTAR TABEL.....	xvi
DAFTAR LAMPIRAN.....	xvii
DAFTAR ISTILAH	xviii
BAB I PENDAHULUAN.....	I-1
1.1 Latar Belakang.....	I-1
1.2 Rumusan Masalah.....	I-3
1.3 Batasan Masalah	I-3
1.4 Tujuan Penelitian	I-3
1.5 Sistematika Penulisan	I-3
BAB II LANDASAN TEORI	II-1
2.1 Citra	II-1
2.2 Refresentasi Citra <i>Digital</i>	II-2
2.3 Pengenalan Wajah	II-4
2.4 Vektor dan Ruang Wajah	II-5
2.5 <i>Principal Component Analysis (PCA)</i>	II-6
2.6 <i>Eigenface</i>	II-9

2.7 <i>Euclidean Distance</i>	II-16
2.8 Pemograman MATLAB.....	II-18
BAB III METODOLOGI PENELITIAN.....	III-1
3.1 Identifikasi Masalah	III-2
3.2 Penetapan Tujuan	III-2
3.3 Pengumpulan Data.....	III-2
3.4 Analisa Sistem	III-2
3.5 Perancangan Sistem.....	III-3
3.6 Implementasi dan Pengujian.....	III-4
3.7 Kesimpulan dan Saran	III-4
BAB IV ANALISA dan PERANCANGAN.....	IV-1
4.1 Analisa Sistem	IV-1
4.1.1 Analisa Metode Pengenalan Wajah	IV-1
4.1.2 Perhitungan Nilai <i>Eigenface</i>	IV-2
4.1.2.1 Ekstraksi Ciri Citra Referensi	IV-2
4.1.2.2 Ekstraksi Ciri Citra Uji	IV-9
4.1.2.3 Proses Pengenalan Citra Menggunakan <i>Euclidean Distance</i>	IV-10
4.2 Penggambaran Perancangan	IV-11
4.2.1 Penggambaran Metode <i>Eigenface</i>	IV-11
4.2.2 Penggambaran Metode Pengenalan Citra.....	IV-12
4.2.3 Perancangan <i>Interface</i>	IV-12
BAB V IMPLEMENTASI dan PENGUJIAN.....	V-1
5.1 Implementasi	V-1
5.1.1 Batasan Implementasi.....	V-1
5.1.2 Lingkungan Implementasi	V-1
5.1.2.1 Lingkungan Perangkat Keras.....	V-2
5.1.2.2 Lingkungan Perangkat Lunak.....	V-2

5.1.3 Implementasi Antarmuka	V-2
5.1.3.1 Tampilan Proses Hitung <i>Eigenface</i>	V-3
5.1.3.2 Tampilan <i>Input</i> Citra Yang Akan Dikenali	V-5
5.1.3.3 Tampilan <i>Update Database</i>	V-7
5.1.3.4 Tampilan Hapus <i>Database</i>	V-8
5.1.3.5 Tampilan Keluar Program	V-9
5.2 Pengujian	V-10
5.2.1 Pengujian Terhadap Citra	V-10
5.2.2 Pengujian <i>Blackbox</i>	V-19
BAB VI PENUTUP	VI-1
6.1 Kesimpulan	VI-1
6.2 Saran.....	VI-1
DAFTAR PUSTAKA	
LAMPIRAN	
DAFTAR RIWAYAT HIDUP	

DAFTAR TABEL

Tabel	Halaman
5.1 Butir Uji Pengujian Modul Hitung Nilai <i>Eigenface</i> Citra.....	V-19
5.2 Butir Uji Pengujian Modul <i>Input</i> Citra Yang Akan Dikenali	V-20
5.3 Butir Uji Pengujian Modul <i>Update Database</i>	V-21
5.4 Butir Uji Pengujian Modul Hapus <i>Database</i>	V-22

DAFTAR ISTILAH

- Citra* : Dapat dijelaskan sebagai 2 dimensi dari $f(x,y)$ sebagai koordinat spasial yang dapat ditentukan nilainya dan objeknya berupa gambar (*Image*).
- Covariance Matrix* : Merupakan perhitungan dalam matrix yang menggunakan dimensi $M \times N^2$ dengan mengalikan kelompok dari jarak perbedaan masing-masing citra dengan rata-rata masing-masing piksel citra.
- Eigenface* : Dapat dijelaskan sebagai metode dengan cara mengambil informasi yang terdapat pada citra wajah menggunakan teknik PCA yang mana hasil tersebut diurutkan dari besar sampai kekecil untuk mendapatkan nilai vektor yang paling menonjolkan ciri dari sebuah citra
- Eigenspace* : Merupakan proyeksi nilai *eigenface* dengan semua citra yang ada sudah dikurangi dengan nilai rata-rata seluruh wajah.
- Eigenvalue & Eigenvektor* : Merupakan perhitungan yang dilakukan untuk mendapatkan ciri yang paling terbesar pada suatu citra sebelum nilai *eigenface* didapatkan.
- Euclidean Distance* : Merupakan teknik dalam pencocokan citra yang nilai perbedaan ukuran akan memberitahukan kesamaan yang terdapat pada dua buah gambar yang dijadikan sebagai pembandingan.
- Face Recognition* : Merupakan proses penganalisa karakteristik dari bentuk muka yang tidak berubah.

<i>Grayscale</i>	: Dapat dijelaskan sebagai nilai RGB (<i>Red, Green, Blue</i>) pada tiap piksel dari sebuah citra.
<i>Interface</i>	: Merupakan tampilan antar muka sistem pengenalan citra wajah terhadap <i>user</i> .
JPEG	: Merupakan jenis tipe file dari citra yang digunakan dalam pengenalan citra wajah ini.
Matlab	: Merupakan sebuah piranti lunak pembantu yang digunakan dalam sistem pengenalan wajah ini.
<i>Mean Subtracted Image</i>	: Merupakan hasil pengurangan rata-rata piksel citra terhadap nilai rata-rata perbedaan setiap citra tersebut.
Piksel	: Merupakan nilai intensitas warna yang dimiliki oleh suatu citra digital.
<i>Principle Component Analysis (PCA)</i>	: Merupakan teknik statistikal yang digunakan dalam lingkungan seperti <i>face recognition</i> dan <i>image compression</i> dan merupakan teknik yang biasa dipakai untuk menemukan pola dalam data dengan dimensi yang besar.
<i>Resize</i>	: Bisa dikatakan sebagai men-seting kembali ukuran asli dari sebuah citra menjadi ukuran yang dibutuhkan.
RGB (<i>Red, Green, Blue</i>)	: Merupakan warna paling dasar dari segala warna yang ada.
<i>Training</i>	: Merupakan suatu tahapan proses untuk pencarian wajah dari beberapa image sebagai inputan untuk mendapatkan hasil yang lebih akurat sebelum masuk pada proses pengenalan wajah.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi informasi yang kian berkembang dewasa ini telah banyak menghasilkan berbagai aplikasi yang menggunakan citra wajah sebagai sumber informasi. Hal ini dikarenakan secara umum sebuah citra wajah dapat memberikan informasi khusus yang berkaitan dengan identifikasi *personal* berbasis pengenalan wajah yang dapat dimanfaatkan dalam suatu sistem pengamanan elektronik. Keuntungan yang dimiliki dari sistem pengamanan berbasis pengenalan wajah adalah kemampuan pengamanannya yang relatif sulit untuk ditembus.

Pengenalan wajah merupakan salah satu pendekatan pengenalan pola untuk keperluan identifikasi *personal* selain pendekatan *biometrik* lainnya seperti pengenalan sidik jari, tanda tangan, retina mata dan sebagainya. Pengenalan citra wajah berhubungan dengan obyek yang tidak pernah sama, karena adanya bagian-bagian yang dapat berubah. Perubahan ini dapat disebabkan oleh ekspresi wajah, intensitas cahaya dan sudut pengambilan gambar, atau perubahan asesoris pada wajah. Dalam kaitan ini, obyek yang sama dengan beberapa perbedaan tersebut harus dikenali sebagai satu obyek yang sama.

Pengenalan wajah manusia mendapat banyak perhatian dalam beberapa tahun terakhir, hal ini karena banyak aplikasi yang menerapkannya antara lain dalam pengamanan gedung, alat identifikasi, ATM, *Tele-Conference*, dan alat bantu dalam pelacakan pelaku kriminal.

Secara umum sistem pengenalan citra wajah dibagi menjadi 2 jenis, yaitu sistem *feature based* dan sistem *image based*. Pada sistem pertama digunakan fitur yang diekstraksi dari komponen citra wajah seperti mata, hidung, dan mulut yang kemudian hubungan antara fitur-fitur tersebut dimodelkan secara *geometris*. Sedangkan sistem kedua menggunakan informasi mentah dari piksel citra yang kemudian direpresentasikan dalam metode tertentu, misalnya *Principal*

Component Analysis (PCA), *Transformasi Wavelet* yang kemudian digunakan untuk klasifikasi identitas citra (Hanif Al Fatta, 2009).

Eigenface merupakan suatu metode yang digunakan untuk mentransformasikan dan mereduksi dimensi dari suatu citra. *Eigenface* menggunakan metode *Principal Component Analysis (PCA)* yaitu suatu metode matematika untuk merepresentasikan sebuah objek, mengekstraksi ciri-ciri sebuah objek dan mereduksi sebuah objek dengan cara mentransformasikannya menggunakan *eigenvalue* dan *eigenvector* secara linier.

Eigenface dapat diperoleh dengan terlebih dahulu merepresentasikan setiap matriks wajah menjadi matriks linier, yang kemudian akan ditentukan vektor wajah rata-ratanya. Dari vektor wajah rata-rata tersebut akan dicari matriks *covariance*-nya yang selanjutnya akan diperoleh *eigenvalue* dan *eigenvector*-nya. Melalui *eigenvector* matriks *covariance* inilah akan kita peroleh *eigenface* yang selanjutnya akan digunakan untuk mendapatkan objek pada penelitian ini.

Pada sistem *feature based* cukup mustahil untuk dapat melakukan ekstraksi terhadap komponen mata ketika ada kacamata hitam yang digunakan pada citra wajah. Selain itu, kontur wajah yang aktif juga dapat terganggu jika pada citra wajah terdapat jenggot ketika menentukan lokasi kontur wajah. Algoritma *eigenface* yang menggunakan *image based* unggul dalam aspek pengenalan wajah ini. Perubahan kecil pada wajah seperti kacamata, jenggot, atau kumis tidak menyebabkan pengurangan performa pengenalan wajah.

Berdasarkan uraian diatas, maka pada tugas akhir ini akan dibuat perangkat lunak pengenalan wajah menggunakan metode *eigenface* yang berorientasi pada *Principal Component Analysis (PCA)*.

1.2 Rumusan Masalah

Dari latar belakang permasalahan tersebut diatas maka masalah yang dirumuskan yaitu bagaimana membangun perangkat lunak pengenalan wajah menggunakan metode *eigenface* yang berorientasi pada *Principal Component Analysis* (PCA).

1.3 Batasan Masalah

Dalam penyusunan tugas akhir ini, untuk mengatasi permasalahan yang ada maka penulis membatasi permasalahan antara lain:

1. Tidak membahas tahapan *pre-processing* pengambilan citra wajah.
2. Ukuran citra wajah yang akan dijadikan citra *referensi* dan citra uji berukuran sama. Dalam tugas akhir ini dibatasi sebesar 180 x 200 *pixel*.

1.4 Tujuan Tugas Akhir

Tujuan penyusunan tugas akhir ini adalah membangun perangkat lunak pengenalan wajah menggunakan metode *eigenface* yang berorientasi pada *Principal Component Analysis* (PCA).

1.5 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir terbagi dalam 6 (enam) bab. Berikut penjelasan dari masing-masing bab.

BAB I : PENDAHULUAN

Menjelaskan dasar-dasar dari penulisan laporan tugas akhir ini, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, serta sistematika penulisan laporan tugas akhir.

BAB II : LANDASAN TEORI

Menjelaskan teori-teori tentang citra, representasi citra *digital*, pengenalan wajah, *vector* ruang dan wajah, *Principal Component Analysis* (PCA), *eigenface*, *euclidean distance* serta metode-metode dan teori pendukung lainnya yang berkaitan dengan tugas akhir yang dibuat.

BAB III : METODOLOGI PENELITIAN

Metodologi penelitian membahas langkah sistematis dan logis yang disusun secara tahap demi tahap pengerjaan selama pembuatan sistem. Seperti penelitian pendahuluan, identifikasi masalah, menetapkan tujuan, mengumpulkan data, analisa dan perancangan sistem, implementasi dan pengujian, serta kesimpulan dan saran.

BAB IV : ANALISA DAN PERANCANGAN

Bab ini membahas hasil analisa dan perancangan yang meliputi pembahasan mengenai analisa sistem, analisa metode pengenalan wajah, perhitungan nilai *eigenface*, penggambaran perancangan, penggambaran metode *eigenface*, penggambaran metode pengenalan wajah, dan perancangan *interface*.

BAB V : IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas implementasi dan pengujian sistem yang meliputi tahapan implementasi, batasan implementasi, lingkungan implementasi, implementasi antar muka, pengujian, pengujian terhadap citra dan pengujian terhadap *blackbox*.

BAB VI : PENUTUP

Bab ini berisi kesimpulan dan saran sebagai hasil akhir dari penelitian tugas akhir yang telah dilakukan.

BAB II

LANDASAN TEORI

2.1 Citra

Citra adalah representasi dua dimensi untuk bentuk *fisik* nyata tiga dimensi. Citra dalam perwujudannya dapat bermacam-macam, mulai dari gambar hitam-putih pada sebuah foto (yang tidak bergerak) sampai pada gambar berwarna yang bergerak pada pesawat televisi. Proses transformasi dari bentuk tiga dimensi ke bentuk dua dimensi untuk menghasilkan citra akan dipengaruhi oleh bermacam-macam faktor yang mengakibatkan penampilan citra suatu benda tidak sama persis dengan bentuk fisik nyatanya. Faktor-faktor tersebut merupakan efek degradasi atau penurunan kualitas yang dapat berupa rentang kontras benda yang terlalu sempit atau terlalu lebar, *distorsi geometrik*, keaburan (*blur*), keaburan akibat obyek yang bergerak (*motion blur*), *noise* atau gangguan yang disebabkan oleh *interferensi* peralatan pembuat citra, baik berupa *transduser*, peralatan elektronik ataupun peralatan optik (Rodiyanayah, 2010).

Teknik dan proses untuk mengurangi atau menghilangkan efek degradasi pada citra *digital* meliputi perbaikan/peningkatan citra (*image enhancement*), restorasi citra (*image restoration*), dan transformasi spasial (*spasial transformation*). Subyek lain dari pengolahan citra *digital* diantaranya adalah pengkodean citra (*image coding*), segmentasi citra (*image segmentation*), representasi dan deskripsi citra (*image representation and description*).

Pengolahan citra dilakukan dengan komputer *digital* maka citra yang akan diolah terlebih dahulu ditransformasikan ke dalam bentuk besaran-besaran *diskrit* dari nilai tingkat keabuan pada titik-titik elemen citra. Bentuk citra ini disebut citra *digital*. Setiap citra *digital* memiliki beberapa karakteristik, antara lain ukuran citra, resolusi dan format lainnya. Umumnya citra *digital* berbentuk persegi panjang yang memiliki lebar dan tinggi tertentu, yang biasanya dinyatakan dalam banyaknya titik atau *piksel* (*picture elemen/piksel*).

Pengolahan citra dapat dibagi kedalam tiga kategori yakni kategori rendah, menengah dan tinggi. Kategori rendah melibatkan operasi-operasi sederhana seperti prapengolahan citra untuk mengurangi derau, pengaturan kontras dan pengaturan ketajaman citra. Pengolahan kategori menengah melibatkan operasi-operasi seperti segmentasi dan klasifikasi citra. Proses pengolahan citra menengah ini melibatkan *input* berupa citra dan *output* berupa atribut (*fitur*) citra yang dipisahkan dari citra *input*. Pengolahan citra kategori tinggi melibatkan proses pengenalan dan deskripsi citra (Darma Putra, 2010).

Ukuran citra dapat juga dinyatakan secara *fisik* dalam satuan panjang (misalnya mm atau inch). Dalam hal ini tentu saja harus ada hubungan antara ukuran titik penyusun citra dengan satuan panjang. Hal tersebut dinyatakan dengan resolusi yang merupakan ukuran banyaknya titik untuk setiap satuan panjang. Biasanya satuan yang digunakan adalah dpi (*dot per inch*). Makin besar resolusi makin banyak titik yang terkandung dalam citra dengan ukuran fisik yang sama. Hal ini memberikan efek penampakan citra menjadi semakin halus.

Format citra *digital* ada bermacam-macam. Karena sebenarnya citra merepresentasikan informasi tertentu, sedangkan informasi tersebut dapat dinyatakan secara bervariasi, maka citra yang mewakilinya dapat muncul dalam berbagai format. Citra yang merepresentasikan informasi yang hanya bersifat *biner* untuk membedakan dua keadaan tentu tidak sama citra dengan informasi yang lebih kompleks sehingga memerlukan lebih banyak keadaan yang diwakilinya. Pada citra *digital* semua informasi tadi disimpan dalam bentuk angka, sedangkan penampilan angka tersebut biasanya dikaitkan dengan warna (Rodiyansyah, 2010).

Setiap titik juga memiliki nilai berupa angka *digital* yang merepresentasikan informasi yang diwakili titik tersebut. Format nilai *piksel* sama dengan format citra keseluruhan. Pada kebanyakan sistem pencitraan, nilai ini biasanya berupa bilangan bulat positif.

2.2 Representasi Citra *Digital*

Komputer dapat mengolah isyarat-isyarat *elektronik digital* yang merupakan kumpulan sinyal *biner* (bernilai dua: 0 dan 1). Untuk itu, citra *digital* harus mempunyai format tertentu yang sesuai sehingga dapat merepresentasikan obyek pencitraan dalam bentuk kombinasi data *biner*.

Citra yang tidak berwarna atau hitam putih dikenal sebagai citra dengan derajat abu-abu (citra *graylevel/grayscale*). Derajat abu-abu yang dimiliki ini bisa beragam mulai dari 2 derajat abu-abu (yaitu 0 dan 1) yang dikenal juga sebagai citra *monochrome*, 16 derajat keabuan dan 256 derajat keabuan.

Dalam sebuah citra *monochrome*, sebuah *piksel* diwakili oleh 1 bit data yang berisikan data tentang derajat keabuan yang dimiliki *piksel* tersebut. Data akan berisi 0 bila *piksel* berwarna hitam dan 1 bila *piksel* berwarna putih. Citra yang memiliki 16 derajat keabuan (mulai dari 0 yang mewakili warna hitam sampai dengan 15 yang mewakili warna putih) direpresentasikan oleh 4 bit data. Sedangkan citra dengan 256 derajat keabuan (nilai dari 0 yang mewakili warna hitam sampai dengan 255 yang mewakili warna putih) direpresentasikan oleh 8 bit data.

Dalam citra berwarna, jumlah warna bisa beragam mulai dari 16, 256, 65536 atau 16 juta warna yang masing-masing direpresentasikan oleh 4,8,16 atau 24 bit data untuk setiap *piksel*-nya. Warna yang ada terdiri dari 3 komponen utama yaitu nilai merah (*red*), nilai hijau (*green*) dan nilai biru (*blue*). Paduan ketiga komponen utama pembentuk warna tersebut dikenal sebagai *RGB color* yang nantinya akan membentuk citra warna (Rodiyansyah, 2010).

2.3 Pengenalan Wajah

Proses pengenalan wajah yang dilakukan oleh komputer tidak semudah dan secepat dibandingkan dengan proses pengenalan yang dilakukan oleh manusia. Manusia dengan mudah dapat mengenali wajah seseorang dengan sangat cepat tanpa rasanya harus berfikir. Manusia juga tidak terpengaruh oleh orientasi wajah orang tersebut, misalnya orang tersebut dalam keadaan agak menoleh, menunduk, menengadah asal dalam batas-batas yang masih dapat dilihat.

Sedangkan komputer selain lambat dalam pengenalan, juga kesulitan pada orientasi wajah yang berlainan, pencahayaan, latar belakang yang berbeda, potongan rambut, kumis atau jenggot, kacamata dan lain sebagainya. Memang otak manusia lebih memiliki keuntungan dalam mengatasi masalah dimana aturan eksplisit tidak dapat dengan mudah diformulasikan, sedangkan komputer mempunyai keuntungan pada bidang seperti matematika dimana aturan-aturan mudah diformulasikan.

Oleh karena itu banyak dilakukan penelitian untuk mencari algoritma-algoritma yang tepat bagi komputer agar dapat mengenali suatu wajah yang diinputkan dengan memperhatikan faktor kecepatan dan akurasinya.

Banyak penelitian yang dilakukan oleh para ahli komputer untuk menganalisa wajah manusia. Penelitian tentang wajah ini terdiri dari beberapa bidang. Misalnya bidang pendeteksian wajah, maksudnya penelitian dilakukan untuk mencari lokasi wajah yang ada dalam *image*. Jadi dimasukkan sebuah foto kedalam program komputer melalui *scanner*, dan nanti program akan menentukan lokasi-lokasi mana saja yang terdapat wajah.

Bidang yang lain yaitu pencarian *feature* dari wajah. Yang dimaksud *feature* dari wajah adalah hidung, mata, alis, telinga, mulut, dan lain sebagainya. Jadi program akan menerima *input* wajah gambar, dan setelah diproses akan menghasilkan lokasi-lokasi mana saja yang terdapat *feature-feature* tersebut.

Ada juga penelitian tentang ekspresi wajah. Dengan *training* yang cukup, maka program akan mengenali ekspresi *input testing* berupa wajah gambar yang dimasukkan, apakah wajah tersebut marah, sedih, tertawa, dan lain sebagainya.

Selain pengklasifikasian ekspresi, para ahli juga berusaha mengklasifikasikan jenis kelamin. Program akan dapat mengenali jenis kelamin dari foto orang yang di-*input*-kan kedalamnya.

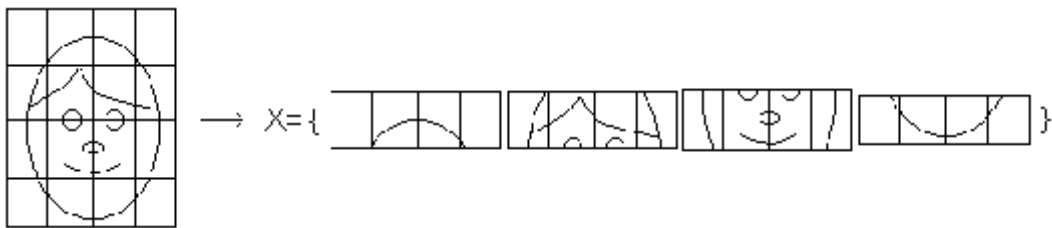
Beberapa contoh aplikasi dari pengenalan wajah oleh komputer adalah :

1. Pengenalan *credit card*, Surat Izin Mengemudi (SIM), *passport*.
2. Pengenalan wajah untuk sekuritas sebagai pengganti tanda tangan atau sidik jari, misal untuk verifikasi *credit card*.
3. Pengenalan pasien yang tidak sadarkan diri.

4. Pengenalan orang hilang atau seorang kriminal.
5. Pengontrolan masyarakat, seperti kamera di bank sehingga dapat mengidentifikasi bila ada orang jahat yang masuk ke bank.
6. Rekonstruksi wajah sesuai dengan bayangan dari saksi pada suatu peristiwa kejahatan.
7. Klasifikasi jenis kelamin.

2.4 *Vector* dan Ruang wajah

Sebuah wajah, yang juga merupakan sebuah citra, dapat dipandang sebagai sebuah *vector*. Apabila citra berukuran $p \times l$ *piksel*, maka banyaknya komponen dari *vector* ini adalah $p \times l$. Konstruksi *vector* dari sebuah citra dibentuk oleh penggabungan sederhana, yaitu baris dari sebuah citra diletakkan saling bersebelahan dengan baris-baris yang lain, seperti yang terlihat pada Gambar 1 di bawah ini:



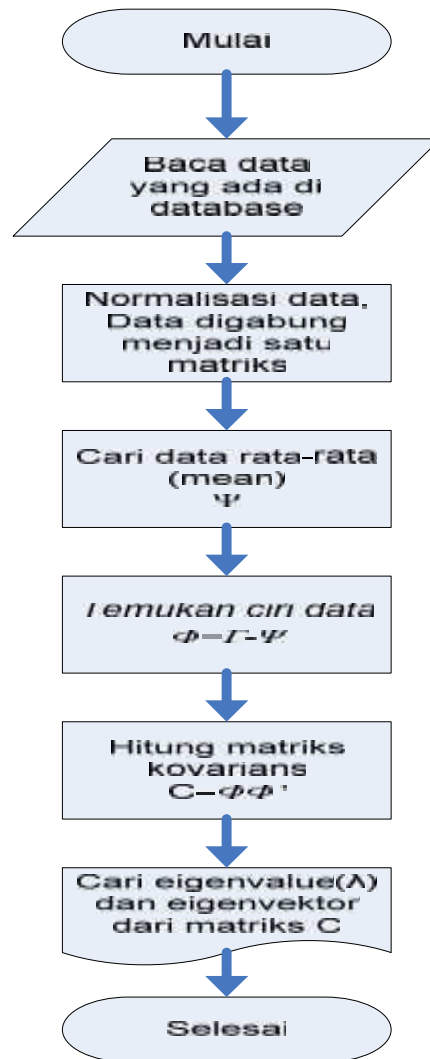
Gambar 2.1 Formasi *Vector* dari Sebuah Gambar

Vector wajah yang telah dideskripsikan sebelumnya merupakan bagian dari sebuah ruang citra yaitu ruang dari keseluruhan citra yang mempunyai dimensi $p \times l$ piksel. Semua wajah mirip satu sama lain dimana wajah-wajah ini mempunyai dua mata, satu hidung, satu mulut, dua telinga, dan lain sebagainya yang terletak pada tempat yang sama. Akibatnya semua *vector* wajah terletak pada tempat-tempat yang amat berdekatan dalam ruang citra. *Vector* basis dari ruang wajah disebut sebagai komponen utama (*principal component*).

2.5 *Principal Component Analysis (PCA)*

Metode PCA dikenal juga dengan nama *Karhunen-Loeve Transformation* (KLT), yang telah dikenal sejak 30 tahun dalam dunia pengenalan pola. PCA memberikan transformasi *ortogonal* yang disebut dengan ‘*eigenimage*’ yang mana sebuah *image* direpresentasikan kedalam bentuk proyeksi linier searah dengan *eigenimage* yang bersesuaian dengan nilai *eigen* terbesar dari *matrix covariance* (atau *scatter matrix*). Secara praktis *matrix covariance* ini dibangun dari sekumpulan *image training* yang diambil dari berbagai obyek/kelas.

PCA termasuk dalam bidang *multivariate analysis* pada ilmu statistik. Metode PCA ini mungkin teknik yang terlama dan terpopuler dalam bidang *multivariate analysis*. *Multivariate analysis* secara sederhana dapat diartikan sebagai metode yang berhubungan dengan *variabel* dalam jumlah besar pada satu atau banyak percobaan.



Gambar 2.2 Flowchart PCA Secara Umum

Prinsip PCA adalah memproyeksikan citra ke dalam ruang *eigen*-nya dengan cara mencari *eigen vector* yang dimiliki setiap citra dan memproyeksikan kedalam ruang *eigen* yang didapat tersebut. Besar ruang *eigen* tergantung dari jumlah citra referensi yang dimiliki.

Proses pengambilan ciri khusus menggunakan PCA secara umum dapat dilihat dari diagram alur pada Gambar 2.2. Prosesnya dimulai dengan menormalisasi setiap data ke bentuk *matrix* satu dimensi. Dalam hal ini dapat berupa *matrix* baris atau *matrix* kolom. Selanjutnya membentuk satu *matrix* baru

yang akan menampung seluruh data yang ada di *database*. Dapat dinyatakan dalam pola :

$$I_m = [\Gamma_1, \Gamma_2, \dots, \Gamma_M] \quad (2.1)$$

Keterangan :

I_m = *matrix* baru yang berisi nilai dari seluruh data

Γ_i = data ke- i

Setiap data akan diubah ke bentuk *matrix* satu dimensi yang panjangnya tergantung dari jumlah data tersebut. Dalam hal ini akan terbentuk suatu *matrix* besar yang berisi seluruh data referensi. Selanjutnya merata-rata data (Ψ). Rata-rata dapat dihitung dengan menggunakan persamaan :

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n \quad (2.2)$$

Keterangan :

Ψ = data rerata (*mean*)

M = banyaknya data di referensi

Γ_n = data ke- n

Kemudian cari *feature* PCA atau ciri datanya (Φ) dengan cara mengurangi data (Γ) dengan reratanya (Ψ).

$$\Phi_i = \Gamma_i - \Psi \quad (2.3)$$

Keterangan :

Φ_i = Pola hasil ekstraksi data ke i

Γ_i = data ke i

Ψ = data rerata (*mean*)

Kemudian dari ciri data ini dicari *matrix* kovariansnya (C). Berikut persamaannya.

$$C = AA^T \quad (2.4)$$

dengan $A = \{\Phi_1, \Phi_2, \dots, \Phi_N\}$ dan A^T adalah *transpose* dari *matrix* A . *Matrix* A sendiri adalah *matrix* yang berisi informasi pola hasil ekstraksi dari seluruh data yang ada (dari $n=1$ sampai citra ke- M). Setelah didapatkan *matrik kovarian* maka nilai *eigen* dan *eigenvector*-nya dapat dicari dengan persamaan dasar *eigen*:

$$AX = \lambda X \quad (2.5)$$

Dengan λ adalah suatu skalar yang dinamakan nilai *eigen*. X adalah *vectoreigen*. Dari *matrix kovarians* yang berisi ciri utama data inilah nantinya didapatkan nilai *eigen* dan *vectoreigen* yang selanjutnya disebut dengan *eigenfile*. Khusus pada penelitian ini karena menyangkut tentang wajah (*face*), maka disebut dengan *eigenface*.

Jadi PCA merupakan algoritma untuk mengambil ciri penting dari sekumpulan *dataset* dengan mereduksi data tersebut menjadi data yang tidak saling berkorelasi. Sasaran PCA adalah menangkap variasi total dan menjelaskannya dengan variabel yang lebih sederhana.

2.6 *Eigenface*

Pemrosesan awal pada citra dua dimensi yang dipakai perlu dilakukan. Tujuan pemrosesan awal adalah untuk mempercepat kinerja dan memperkecil ukuran memori yang digunakan. Pemrosesan awal dilakukan dengan mengekstrak ciri dari citra.

Citra dua dimensi yang akan dikenali dan dikumpulkan untuk mewakili obyek tersebut sebagai citra acuan. Ekstraksi ciri dilakukan pada kumpulan tersebut untuk mendapat informasi objek. Hasil ekstraksi ciri kemudian digunakan untuk proses pengenalan objek.

Berdasarkan persamaan (2.5) pengertian nilai *eigen* dan *vectoreigen* dapat lebih dipahami jika ditinjau dalam bentuk persoalan fisis. Misal terdapat selembar membran elastik dua dimensi yang dapat dinyatakan dalam koordinat x dan y . Membran tersebut mendapat perlakuan *fisik* yakni dapat ditekan, ditarik maupun dirotasi terhadap titik asal. Seandainya membran tersebut mendapat perlakuan di atas maka setelah deformasi titik mula-mula pada bidang membran (x,y) berubah posisi menjadi (X,Y) dan dapat dikatakan terdapat sebuah *matrix* M yang menggambarkan bentuk deformasi tersebut [10].

Perubahan *vector* posisi dapat dinyatakan sebagai $R=\lambda r$, λ =konstanta dengan *vector* R sebagai *eigenvector* dan λ disebut sebagai nilai *eigen* dari *matrix* transformasi M . Untuk mengilustrasikan penentuan nilai *eigen*, dapat dilihat pada contoh di bawah ini:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Dengan syarat *vectoreigen* $R=\lambda I$, bentuk di atas dapat dituliskan sebagai :

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 5 & -2 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \lambda x \\ \lambda y \end{bmatrix}$$

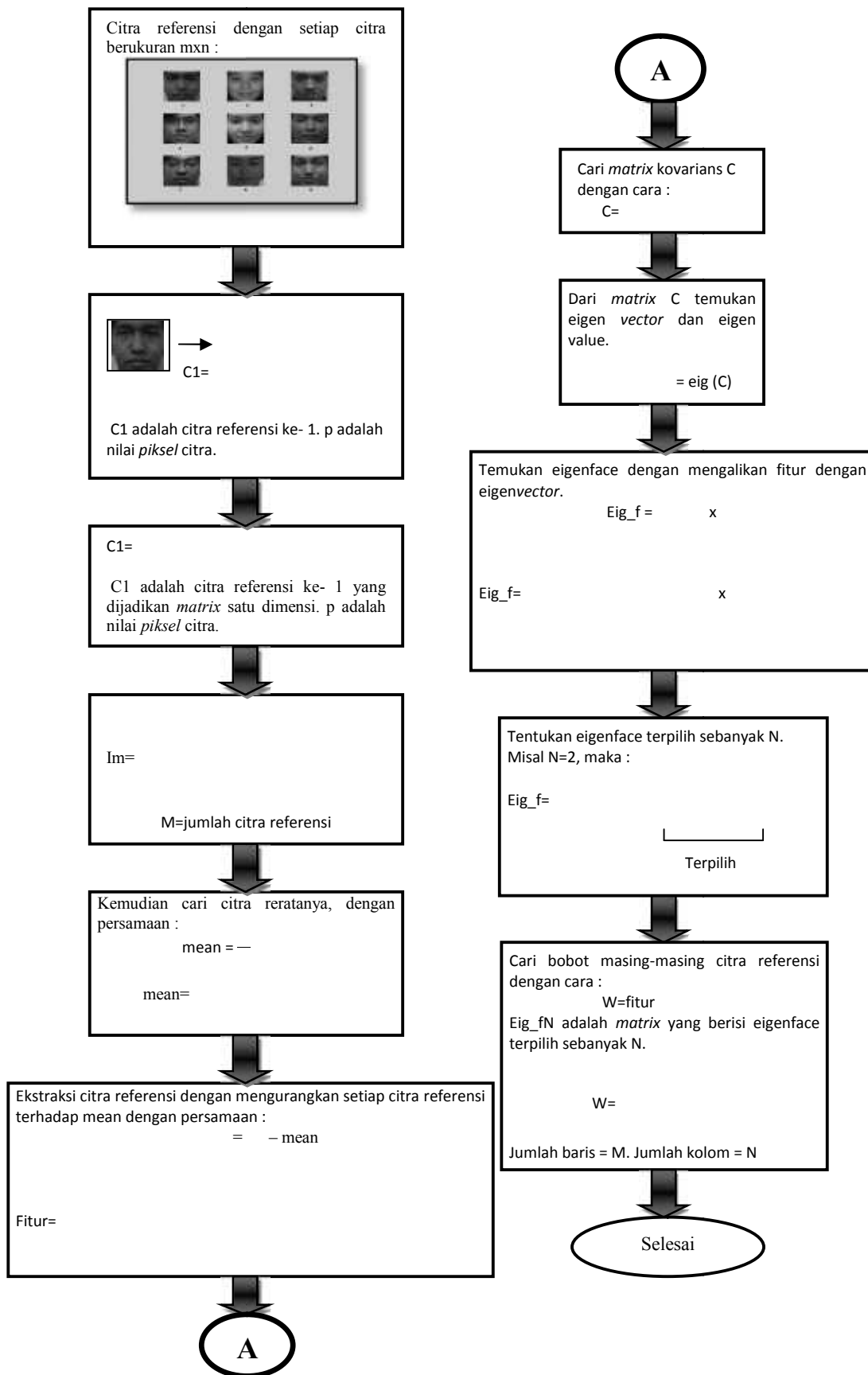
Atau dapat juga ditulis sebagai :

$$(5 - \lambda)x - 2y = 0$$

$$-2x + (2 - \lambda)y = 0$$

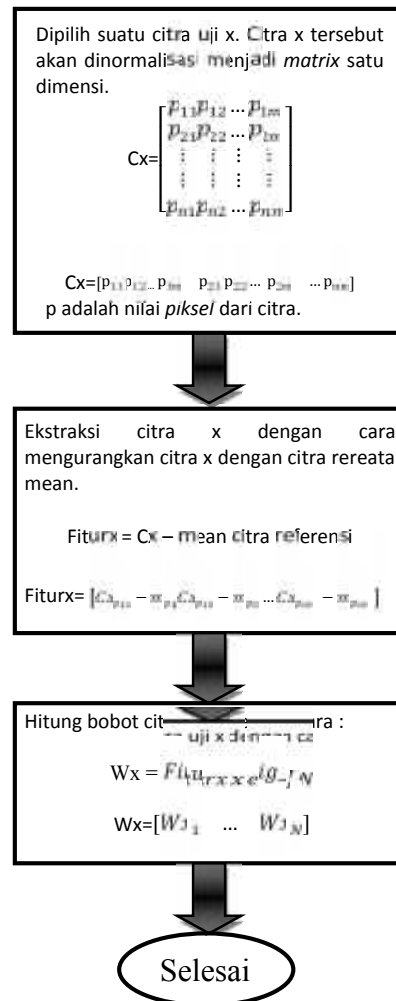
Interpretasi dari bentuk tersebut adalah nilai *eigen* dari *matrix* transformasi memberikan informasi seberapa besar deformasi bidang diberikan. Sedangkan *eigenvector* dari *matrix* transformasi memberikan informasi arah perubahan deformasi tersebut. Nilai *eigen* dan *eigenvector* itulah yang nantinya akan berisi informasi tentang citra wajah tertentu sehingga disebut sebagai *eigenface*.

Nilai *eigen* dan *vectoreigen* didapat dari ekstraksi ciri citra berdasarkan PCA sebagaimana terlihat pada Gambar 2.2. Pada penelitian ini proses pencarian *eigenface* sampai ditemukan nilai bobot dari citra referensi maupun citra uji dapat dilihat pada blok diagram Gambar 2.3 dan Gambar 2.4.



Gambar 2.3 Diagram Alur Proses Ekstraksi Ciri Citra Referensi

Setelah bobot citra referensi ditemukan langkah selanjutnya adalah menemukan bobot citra yang diujikan. Langkahnya dapat dilihat pada Gambar 2.4.



Gambar 2.4 Diagram Alur Ekstraksi Ciri Citra Uji

Gambar 2.3 dan Gambar 2.4 menunjukkan bagaimana proses awal sampai ditemukannya bobot dari citra referensi dan citra uji. Bobot tersebut berisi ciri utama dari setiap citra. Berdasarkan Gambar 2.3 proses awal menunjukkan pembacaan citra yang ada di *database* (citra *referensi*). Setiap citra terdiri dari *pixel-pixel* dengan ukuran $m \times n$ *pixel*. m adalah jumlah kolom dan n adalah jumlah baris.

Setiap citra referensi akan dinormalisasi menjadi *matrix* satu dimensi. Pada penelitian ini setiap citra akan dirubah menjadi *matrix* baris.

$$C1 = [p_{11} p_{12} \dots p_{1n} \quad p_{21} p_{22} \dots p_{2n} \quad \dots \quad p_{m1} \dots p_{mn}]$$

C1 adalah citra referensi ke-1 yang dijadikan *matrix* satu dimensi. p adalah nilai *pixel* citra. Kemudian *matrix* tersebut digabung menjadi satu *matrix* besar Im.

$$Im = \begin{bmatrix} C1_{p_{11}} & C1_{p_{12}} & \dots & C1_{p_{nn}} \\ C2_{p_{11}} & C2_{p_{12}} & \dots & C2_{p_{nn}} \\ C3_{p_{11}} & C3_{p_{12}} & \dots & C3_{p_{nn}} \\ \vdots & \vdots & \ddots & \vdots \\ CM_{p_{11}} & CM_{p_{12}} & \dots & CM_{p_{nn}} \end{bmatrix}$$

Setiap baris dari *matrix* Im adalah satu citra referensi. Jumlah baris *matrix* Im adalah M dengan M adalah jumlah citra referensi. Jumlah kolom dari *matrix* Im adalah sebanyak m x n. Langkah selanjutnya adalah mencari citra reratanya. Citra rerata dapat dicari dengan menggunakan persamaan (2.2). *Matrix* citra rerata ini akan berupa *matrix* baris dengan panjang m x n.

$$mean = [n_{p_{11}} \quad n_{p_{12}} \quad \dots \quad n_{p_{nn}}]$$

Setelah ditemukan citra rerata langkah selanjutnya adalah mencari fitur berdasarkan persamaan (3). *Matrix* fitur ini dapat dituliskan sebagai berikut.

$$Fitur = \begin{bmatrix} C1_{p_{11}} - n_{p_{11}} & C1_{p_{12}} - n_{p_{12}} & \dots & C1_{p_{nn}} - n_{p_{nn}} \\ C2_{p_{11}} - n_{p_{11}} & C2_{p_{12}} - n_{p_{12}} & \dots & C2_{p_{nn}} - n_{p_{nn}} \\ C3_{p_{11}} - n_{p_{11}} & C3_{p_{12}} - n_{p_{12}} & \dots & C3_{p_{nn}} - n_{p_{nn}} \\ \vdots & \vdots & \ddots & \vdots \\ CM_{p_{11}} - n_{p_{11}} & CM_{p_{12}} - n_{p_{12}} & \dots & CM_{p_{nn}} - n_{p_{nn}} \end{bmatrix} \quad (2.6)$$

Setiap elemen dari *matrix* Im akan dikurangkan dengan reratanya. Ukuran *matrix* fitur ini sama dengan ukuran *matrix* Im, yaitu M baris dan m x n kolom. Selanjutnya cari *matrix covarian* C sesuai dengan persamaan (4). Dalam hal ini *matrix* C dapat dituliskan sebagai berikut.

$$C = Fitur \times Fitur^T \quad (2.7)$$

Setelah itu cari nilai *eigen* dan *vectoreigen* dari *matrix* C. Pencarian nilai *eigen* dan *vectoreigen* ini dapat dituliskan dengan :

$$C \times eval = ev \times eval \quad (2.8)$$

ev adalah *vectoreigen* dan eval adalah eigen value. Kemudian *vectoreigen* dikalikan dengan *matrix* $Fitur^T$ sehingga didapatkan *eigenface* dari citra referensi.

$$\text{Eig}_f = \text{Fitur}^T \times \text{ev} \quad (2.9)$$

$$\text{Eig}_f = \begin{bmatrix} fC1_1 & fC2_1 & \dots & fCM_1 \\ fC1_2 & fC2_2 & \dots & fCM_2 \\ fC1_3 & fC2_3 & \dots & fCM_3 \\ \vdots & \vdots & \ddots & \vdots \\ fC1_n & fC2_n & \dots & fCM_n \end{bmatrix} \times \begin{bmatrix} ev_1 & \dots & ev_m \\ \vdots & \ddots & \vdots \\ ev_n & \dots & ev_{nm} \end{bmatrix}$$

Eig_f adalah *matrix eigenface* yang mempunyai ukuran yang berkebalikan dengan ukuran *matrix* Im. *Matrix eigenface* ini akan mempunyai jumlah kolom sebanyak M dengan M adalah jumlah citra referensi, dan mempunyai jumlah baris sebanyak m x n. *Matrix eigenface* ini akan berisi nilai *eigenface* setiap citra referensi yang tersusun dari kecil ke besar. Langkah berikutnya adalah menentukan nilai N dengan N adalah jumlah *eigenface* terpilih yang mempunyai nilai yang signifikan dari *eigenface* lainnya. Nilai *eigenface* terpilih ini akan mewakili seluruh *eigenface* citra referensi. Artinya untuk menghitung bobot tidak perlu menggunakan *eigenface* sebesar 100%, cukup sebagian saja. Nilai N didapat setelah pengujian. Misal N=2, maka :

$$\text{Eig}_f = \begin{bmatrix} ef1_1 & \dots & efM - 1_{n-1} & efM_m \\ \vdots & \ddots & \vdots & \vdots \\ ef1_n & \dots & efM - 1_{n-1} & efM_{nm} \end{bmatrix}$$

└──────────────────┘
Terpilih

Kemudian bobot masing-masing citra referensi dapat dihitung dengan cara mengalikan *matrix* Fitur dengan *matrix eigenface* terpilih.

$$W = \text{Fitur} \times \text{Eig}_f N \quad (2.10)$$

$$W = \begin{bmatrix} W_1 & \dots & W_N \\ \vdots & \ddots & \vdots \\ W_M & \dots & W_{MN} \end{bmatrix}$$

$\text{Eig}_f N$ adalah *matrix* yang berisi *eigenface* terpilih sebanyak N. *Matrix* W adalah *matrix* yang berisi bobot dari seluruh citra referensi. Ukuran *matrix* W adalah jumlah kolom sebesar N dan jumlah baris sebesar M. Artinya setiap baris dari *matrix* W adalah bobot satu citra referensi. Jumlah kolom *matrix* W adalah jumlah banyaknya ciri utama yang diambil.

Setelah menemukan nilai bobot (*matrix* W) ini maka proses ekstraksi ciri utama dari citra referensi selesai. Bobot tersebut disimpan sehingga proses perhitungannya hanya akan berlangsung satu kali selama program ini berjalan. Hal ini tentunya akan menghemat waktu komputasi program.

Gambar 2.4 menggambarkan proses perhitungan bobot dari citra uji. Bobot ini juga berisi ciri utama dari citra uji. Citra uji harus memiliki ukuran *piksel* yang sama dengan citra referensi, yaitu $m \times n$. Prosesnya dimulai dengan menormalisasi citra uji menjadi *matrix* baris.

$$C_x = \begin{bmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & \dots & p_{nn} \end{bmatrix}$$



$$C_x = [p_{11} p_{12} \dots p_{1n} \quad p_{21} p_{22} \dots p_{2n} \quad \dots p_{n1} \dots p_{nn}]$$

C_x adalah citra uji x dengan p adalah nilai *piksel* dari citra. Citra uji juga dihitung fiturnya sesuai persamaan (3). Prosesnya dapat dituliskan:

$$\text{Fitur}_x = C_x - \text{mean citra referensi}$$

$$\text{Fitur}_x = [(C_{x_{p_{11}}} - m_{p_1}) \quad (C_{x_{p_{12}}} - m_{p_2}) \quad \dots \quad (C_{x_{p_{nn}}} - m_{p_{nn}})]$$

Fitur_x adalah *matrix* yang berisi ciri dari citra uji x . Hal tersebut dilakukan dengan mengurangi setiap elemen *matrix* citra uji dengan elemen *matrix* *mean* dari citra referensi. Hasilnya berupa *matrix* baris dengan panjang $m \times n$ *piksel*. Langkah berikutnya adalah menghitung bobot citra uji dengan cara :

$$W_x = \text{Fitur}_x \times \text{eig_fN} \tag{2.11}$$

$$W_x = [W_{\lambda_1} \quad \dots \quad W_{\lambda_N}]$$

W_x adalah *matrix* yang berisi bobot dari citra uji x . Cara menghitungnya adalah dengan mengalikan *matrix* Fitur_x dengan *matrix* eig_fN . *Matrix* eig_fN adalah *matrix* yang berisi *eigenface* terpilih dari citra referensi. Ukuran *matrix* W_x adalah satu baris dan N kolom dengan N adalah jumlah *eigenface* terpilih dari citra referensi.

Dengan ditemukannya nilai bobot citra uji ini (W_x) maka proses ekstraksi ciri utama dari citra uji selesai. Kedua bobot ini (W dan W_x) akan digunakan pada proses pengenalan. Pada penelitian ini proses pengenalan atau *recognition* menggunakan konsep jarak. Semakin kecil jarak diantara keduanya maka akan semakin mirip. Sebaliknya, semakin besar jarak maka akan semakin tidak mirip.

2.7 *Euclidean Distance*

Konsep jarak sebagai indikasi pengenalan yang digunakan adalah *euclidean distance*. Metode *euclidean* adalah metode pengukuran jarak garis lurus (*straight line*) antara dua titik, misal titik X (X_1, X_2, \dots, X_n) dan titik Y (Y_1, Y_2, \dots, Y_n). Metode *Euclidean* sendiri memiliki rumusan (*formula*) pengembangannya sesuai dengan keadaan ruang. Dalam hal ini akan digunakan ruang satu dimensi. Berikut persamaan umumnya (Rodiyansyah, 2010).

$$d(x,y) = \|x - y\|^2 = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (2.13)$$

Keterangan :

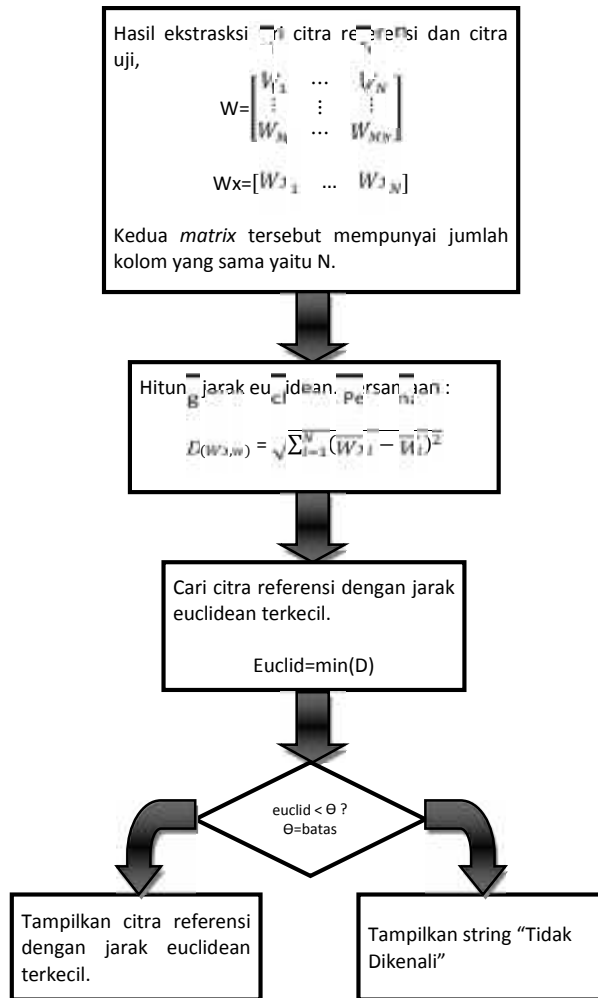
$d(x,y)$ = jarak *euclidean*

x_1 = data uji

y_1 = data referensi

Khusus pada penelitian ini data uji adalah bobot dari citra uji dan data referensi adalah bobot dari citra referensi. *Euclidean distance* inilah yang nantinya digunakan dalam mengambil keputusan apakah citra uji dikenali atau tidak.

Jadi setelah melalui proses klasifikasi menggunakan algoritma PCA dan menemukan *eigenface*, kemudian menghitung bobot dari citra uji dan citra wajah referensi, baru kemudian dilakukan proses pengenalan menggunakan jarak *Euclidean* ini. *Euclidean distance* ini akan melihat kedekatan antara *eigenface* citra wajah referensi dan citra wajah uji. Semakin dekat nilai bobot citra wajah referensi dengan nilai bobot citra wajah uji maka akan dikenali sebagai citra wajah yang ada di *database*. Proses yang menggambarkan perhitungan jarak *euclidean* ini dapat dilihat pada Gambar 2.5.



Gambar 2.5 Diagram Alur Proses Perhitungan Jarak Euclidean

Matrix Wx yang berisi bobot citra uji akan dibandingkan dengan setiap baris dari *matrix* W . Setiap satu baris *matrix* W adalah bobot dari satu citra referensi. Artinya bobot citra uji akan dibandingkan dengan bobot masing-masing citra referensi. Besarnya nilai bobot akan bergantung pada jumlah *eigenface* terpilih dari citra referensi. Berdasarkan persamaan (6) maka perbandingannya dapat dituliskan:

$$D_{(W, Wx)} = \sqrt{\sum_{i=1}^N (Wx_i - W_i)^2} \quad (2.14)$$

D adalah variabel yang berisi seluruh perbandingan bobot citra uji terhadap masing-masing citra referensi. Jumlah D akan sama dengan jumlah M atau jumlah citra referensi. N adalah jumlah *eigenface* terpilih dari citra referensi.

Dari seluruh bobot akan dicari bobot yang terkecil. Hal tersebut dapat dituliskan dengan:

$$\text{Euclid}=\min(D)$$

Euclid adalah suatu *variabel* yang digunakan untuk menyimpan jarak *euclidean* terkecil. Setelah bobot terkecil ditemukan maka bobot tersebut akan dibandingkan dengan suatu nilai batas (Θ) sebagai indikasi apakah citra uji dikenali atau tidak. Besarnya nilai Θ akan didapat setelah pengujian dilakukan. Jika bobot citra uji lebih kecil dari Θ maka citra tersebut dikenali sebagai salah satu citra referensi, kemudian citra referensi tersebut akan ditampilkan. Dan jika bobot citra uji lebih besar dari Θ maka citra uji akan dianggap tidak dikenali, kemudian dimunculkan string “Citra Tidak Dikenali”. Pada tahap ini seluruh proses penelitian ini akan selesai. Dan ketika ada citra uji baru maka perhitungan bobot citra referensi tidak perlu diulang selama program ini masih berjalan (Rodiyansyah, 2010).

2.8 Pemrograman Matlab

MATLAB adalah sebuah bahasa dengan (*high-performance*) kinerja tinggi untuk komputasi masalah teknik. Matlab mengintegrasikan komputasi, visualisasi, dan pemrograman dalam suatu model yang sangat mudah untuk pakai dimana masalah-masalah dan penyelesaiannya diekspresikan dalam notasi matematika yang familiar. Penggunaan Matlab meliputi bidang–bidang:

1. Matematika dan Komputasi
2. Pembentukan *Algorithm*
3. Akusisi Data
4. Pemodelan, simulasi, dan pembuatan prototipe
5. Analisa data, explorasi, dan visualisasi
6. Grafik Keilmuan dan bidang Rekayasa

MATLAB merupakan suatu sistem interaktif yang memiliki elemen data dalam suatu *array* sehingga tidak lagi kita dipusingkan dengan masalah dimensi. Hal ini memungkinkan kita untuk memecahkan banyak masalah teknis yang terkait dengan komputasi, khususnya yang berhubungan dengan *matrix* dan formulasi *vector*, yang mana masalah tersebut merupakan momok apabila kita

harus menyelesaikannya dengan menggunakan bahasa *level* rendah seperti Pascall, C dan Basic.

Nama MATLAB merupakan singkatan dari *matrix laboratory*. MATLAB pada awalnya ditulis untuk memudahkan akses perangkat lunak *matrix* yang telah dibentuk oleh LINPACK dan EISPACK. Saat ini perangkat MATLAB telah menggabung dengan LAPACK dan BLAS *library*, yang merupakan satu kesatuan dari sebuah seni tersendiri dalam perangkat lunak untuk komputasi *matrix*.

Dalam lingkungan perguruan tinggi teknik, Matlab merupakan perangkat standar untuk memperkenalkan dan mengembangkan penyajian materi matematika, rekayasa dan kelimuan. Di industri, MATLAB merupakan perangkat pilihan untuk penelitian dengan produktifitas yang tinggi, pengembangan dan analisisnya.

Fitur-fitur MATLAB sudah banyak dikembangkan, dan lebih kita kenal dengan nama *toolbox*. Sangat penting bagi seorang pengguna Matlab, *toolbox* mana yang mendukung untuk *learn* dan *apply teknologi* yang sedang dipelajarinya. *Toolbox* ini merupakan kumpulan dari fungsi-fungsi MATLAB (*M-files*) yang telah dikembangkan ke suatu lingkungan kerja MATLAB untuk memecahkan masalah dalam kelas *particular*. Area-area yang sudah bisa dipecahkan dengan *toolbox* saat ini meliputi pengolahan sinyal, sistem kontrol, *neural networks*, *fuzzy logic*, *wavelets*, dan lain-lain (Romy Fadly, “*Dasar Pengolahan Citra Dengan Matlab*”).

Sebagai sebuah sistem, MATLAB tersusun dari 5 bagian utama:

1. *Development Environment*.

Merupakan sekumpulan perangkat dan fasilitas yang membantu untuk menggunakan fungsi-fungsi dan *file-file* MATLAB. Beberapa perangkat ini merupakan sebuah *graphical user interfaces* (GUI). Termasuk didalamnya adalah MATLAB *desktop* dan *Command Window*, *command history*, sebuah editor dan *debugger*, dan *browsers* untuk melihat *help*, *workspace*, *files*, dan *search path*.

2. *MATLAB Mathematical Function Library.*

Merupakan sekumpulan algoritma komputasi mulai dari fungsi-fungsi dasar seperti: *sum*, *sin*, *cos*, dan *complex arithmetic*, sampai dengan fungsi fungsi yang lebih kompek seperti *matrix inverse*, *matrix eigenvalues*, *Bessel functions*, dan *fast Fourier transforms*.

3. *MATLAB Language.*

Merupakan suatu *high-level matrix/array language* dengan *control flow statements*, *functions*, *data structures*, *input/output*, dan *fitur-fitur object-oriented programming*. Ini memungkinkan bagi kita untuk melakukan kedua hal baik "pemrograman dalam lingkup sederhana " untuk mendapatkan hasil yang cepat, dan "pemrograman dalam lingkup yang lebih besar" untuk memperoleh hasil-hasil dan aplikasi yang kompleks.

4. *Graphics.*

MATLAB memiliki fasilitas untuk menampilkan *vector* dan *matrix* sebagai suatu *grafik*. Didalamnya melibatkan *high-level functions* (fungsi-fungsi level tinggi) untuk visualisasi data dua dimensi dan data tiga dimensi, *image processing*, *animation*, dan *presentation graphics*. Ini juga melibatkan fungsi level rendah yang memungkinkan bagi anda untuk membiasakan diri untuk memunculkan grafik mulai dari bentuk yang sederhana sampai dengan tingkatan *graphical user interfaces* pada aplikasi MATLAB anda.

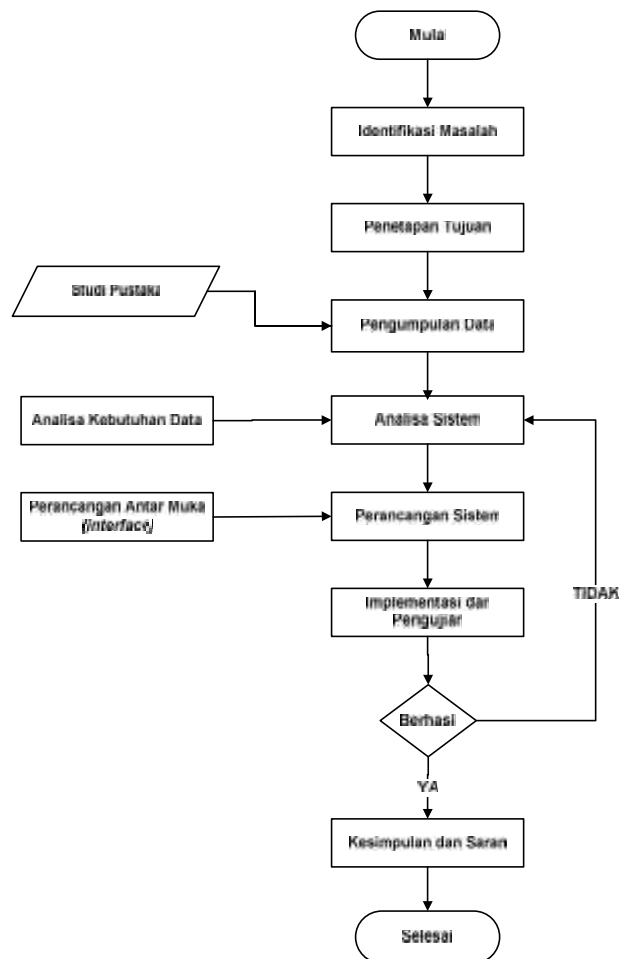
5. *MATLAB Application Program Interface (API).*

Merupakan suatu *library* yang memungkinkan program yang telah di tulis dalam bahasa C dan Fortran mampu berinteraksi dengan MATLAB. Ini melibatkan fasilitas untuk pemanggilan *routines* dari MATLAB (*dynamic linking*), pemanggilan MATLAB sebagai sebuah *computational engine*, dan untuk membaca dan menuliskan MAT-files.

BAB III

METODOLOGI PENELITIAN

Metodologi adalah tata cara yang disusun secara pasti, sistematis dan logis sebagai landasan untuk suatu kegiatan tertentu. Metodologi yang diperlukan untuk tugas akhir ini terdiri dari beberapa tahap seperti: tahap pengumpulan data, tahap analisa dan perancangan, tahap implementasi dan tahap pengujian. Gambar 3.1 menunjukkan kerangka keseluruhan metodologi pengembangan perangkat lunak pengenalan wajah berdasarkan penampilan (*appearance based*) menggunakan metode *eigenface* yang berorientasi pada *Principal Component Analysis* (PCA).



Gambar 3.1 Langkah-Langkah Penelitian

3.1 Identifikasi Masalah

Pada tugas akhir ini, masalah penelitian secara umum bisa kita temukan lewat studi literatur atau lewat pengamatan lapangan, selanjutnya dilakukan pengidentifikasian masalah. Masalah yang akan diidentifikasi adalah yaitu bagaimana merancang dan membangun aplikasi pengenalan wajah menggunakan metode *eigenface* yang berorientasi pada *Principal Component Analysis* (PCA).

3.2 Penetapan Tujuan

Penetapan tujuan sangat diperlukan untuk menjawab permasalahan yang ada. Penetapan tujuan dilakukan setelah mengidentifikasi masalah. Tujuan akan ditetapkan dengan cara mengetahui dan menentukan apa saja yang perlu dipertahankan, ditingkatkan, dihilangkan, dievaluasi dan diperbarui dari suatu masalah yang ada dapat teratasi.

3.3 Pengumpulan Data

Tahapan ini merupakan langkah-langkah yang dilakukan untuk mendapatkan data yang dibutuhkan, yaitu dengan membaca buku-buku yang berhubungan dengan pengolahan citra, metode *eigenface*, *Principal Component Analysis* (PCA), metode *euclidean distance*, serta mencari informasi, artikel dan literatur dari internet.

3.4 Analisa Sistem

Pada tahap ini dilakukan analisa data dan permasalahan yang telah dirumuskan, kemudian merancang sebuah sistem yang dapat menjawab permasalahan dan kendala yang ada. Analisa yang dilakukan adalah:

Analisa setelah data yang dikumpulkan telah lengkap agar selanjutnya mulai merancang sebuah sistem untuk pengenalan wajah dengan menerapkan metode *eigenface*. Pada saat menganalisa data, ada beberapa tahap yang harus dilakukan, yaitu mengidentifikasi kebutuhan sistem, fungsi sistem, memodelkan sistem yang akan dibangun, karakteristik pengguna, merancang lingkungan implementasi, serta merancang antar muka pengguna sistem yang akan dibangun.

Perancangan antarmuka pengguna sistem digunakan untuk memudahkan tahap implementasi pada saat membangun antar muka pengguna.

Tahapan implementasi merupakan tahapan penerjemah hasil analisa kedalam bentuk *coding* sesuai dengan hasil perancangan sistem yang telah dibuat. Bahasa pemrograman yang dibuat untuk membangun sistem ini menggunakan Matlab.

Selanjutnya dilakukan pengujian terhadap perangkat lunak yang telah dibangun agar dapat diketahui hasilnya. Jika terdapat *error*, maka proses akan kembali ketahap analisa sistem, perancangan sistem dan implementasi untuk dilakukan pengecekan ulang.

Tahapan analisa menentukan bagaimana implementasi harus dikerjakan, dan pada tahapan perancangan menentukan bagaimana pemecahan masalah akan dikerjakan atau bagaimana melakukannya

3.5 Perancangan Sistem

Tujuan dari perancangan sistem ini adalah bagaimana mengimplementasikan permasalahan yang ada kedalam sebuah program dan memberikan gambaran komponen-komponen sistem secara umum kepada pengguna sistem tentang sistem yang akan dibuat.

Tahap-tahap yang dilakukan dalam perancangan ini adalah sebagai berikut:

1. Perancangan antarmuka (*input/output*)
2. Perancangan menu

3.6 Implementasi dan Pengujian

Tahapan implementasi merupakan tahap dimana sistem siap dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat benar-benar dapat menghasilkan tujuan yang ingin dicapai.

Dalam implementasi dan pengujian ini akan digunakan:

1. Perangkat lunak dan perangkat keras.

Lingkungan implementasi sistem ada dua, yaitu lingkungan perangkat keras (*Hardware*) dan perangkat lunak (*Software*).

2. *Coding*.

Pembuatan *coding* program dilakukan menggunakan Matlab.

Kumpulan dari semua program yang telah diintegrasikan perlu dites kembali untuk melihat apakah suatu program dapat menerima *input* data dengan baik, dapat memprosesnya dengan baik dan dapat memberikan *output* kepada program lainnya.

3.7 Kesimpulan dan Saran

Kesimpulan yang diambil dapat bersifat *positif* maupun *negatif* yang ditinjau dari berbagai aspek, baik aspek *performance* atau *interface* sistem yang bersangkutan. Sementara saran merupakan sesuatu yang diharapkan di masa mendatang bagi perkembangan sistem selanjutnya.

BAB IV

ANALISA DAN PERANCANGAN

Analisa memegang peranan yang penting dalam membuat rincian perangkat lunak pada komputer. Analisa merupakan langkah pemahaman persoalan sebelum mengambil tindakan atau keputusan penyelesaian hasil utama, sedangkan tahap perancangan sistem adalah membuat rincian hasil dari analisa menjadi bentuk perancangan agar dapat dipahami dalam menjelaskan analisisnya dalam dunia nyata sehingga mendapatkan gambaran tentang analisa dan mudah dimengerti.

4.1 Analisa Sistem

Analisa sistem yang akan dibahas dalam bab ini adalah analisa metode pengenalan wajah, dan penggambaran perancangan perangkat lunak yang akan dibangun.

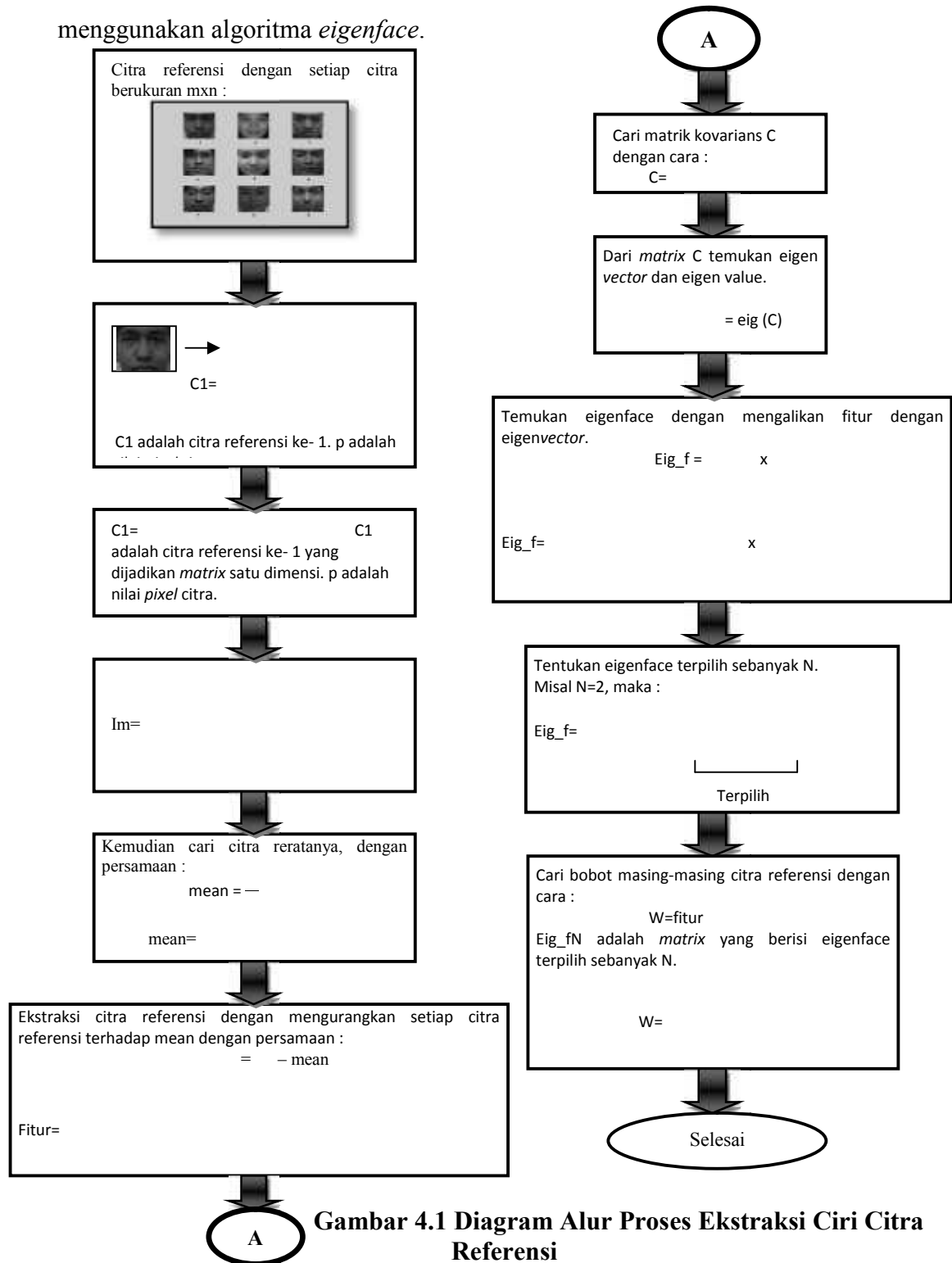
4.1.1 Analisa Metode Pengenalan Wajah

Proses pengenalan wajah diawali dengan pengambilan citra wajah yang nantinya akan dikenali oleh sistem dan dijadikan citra referensi, Kemudian tahap selanjutnya wajah tersebut dikumpulkan kedalam satu folder *image* untuk dihitung nilai *eigen*-nya. Setelah nilai *eigen* didapat maka tahap selanjutnya yakni mengambil citra wajah yang akan diujikan ke sistem pengenalan wajah, citra uji ini terlebih dahulu dihitung juga nilai *eigen*-nya. Kemudian dengan menggunakan metode *euclidean distance* akan ditelusuri citra uji tersebut apakah bisa dikenali oleh sistem. Caranya dengan membandingkan nilai *eigen* dari citra uji dengan semua citra referensi yang berada dalam sistem. Dan perbandingan nilai *eigen* yang paling kecil dianggap yang paling mirip.

4.1.2 Perhitungan Nilai *Eigenface*

4.1.2.1 Ekstraksi Ciri Citra Referensi

Berikut ini merupakan alur proses ekstraksi ciri citra referensi dengan menggunakan algoritma *eigenface*.



Gambar 4.1 Diagram Alur Proses Ekstraksi Ciri Citra Referensi

Untuk lebih mudah memahami perhitungan rumusnya berikut diberikan contoh perhitungan yang lebih sederhana dengan mengambil 3 buah citra wajah dengan ukuran *pixel* 2 x 2. Perhitungan ini dibuat hanya untuk gambaran umum dari tiap-tiap rumus yang ada dan bukan merupakan perhitungan sebenarnya karena perhitungan sebenarnya menggunakan citra dengan *pixel* berukuran 180 x 200 sehingga perhitungannya terlalu besar untuk dijabarkan.

Matrix 2 x 2 diasumsikan sebagai suatu citra sederhana yang setiap elemennya dianggap sebagai nilai *pixel* dari citra. Hal ini dilakukan untuk menggambarkan metode *eigenface* secara sederhana. Misal ada citra A, citra B dan citra C sebagai citra referensi. $M = 3$. M adalah jumlah citra referensi.

$$A = \begin{bmatrix} 7 & 4 \\ 2 & 8 \end{bmatrix} \quad B = \begin{bmatrix} 9 & 1 \\ 5 & 4 \end{bmatrix} \quad C = \begin{bmatrix} 3 & 7 \\ 2 & 6 \end{bmatrix}$$

Kemudian masing-masing citra akan dirubah menjadi *matrix* baris dengan ukuran satu baris dan $2 \times 2 = 4$ kolom.

$$A = [7 \quad 4 \quad 2 \quad 8]$$

$$B = [9 \quad 1 \quad 5 \quad 4]$$

$$C = [3 \quad 7 \quad 2 \quad 6]$$

Selanjutnya seluruh citra akan digabungkan ke dalam satu *matrix* besar I_m dengan ukuran M ($M=3$) baris dan 4 (2×2) kolom.

$$I_m = \begin{bmatrix} 7 & 4 & 2 & 8 \\ 9 & 1 & 5 & 4 \\ 3 & 7 & 2 & 6 \end{bmatrix}$$

Baris pertama adalah citra A, baris kedua adalah citra B dan baris ketiga adalah citra C. Langkah berikutnya adalah mencari citra rerata. Hal ini dilakukan sesuai dengan persamaan (2.2). Setiap kolom dari *matrix* I_m akan dihitung reratanya.

$$\text{Kolom 1} = \begin{bmatrix} 7 \\ 9 \\ 3 \end{bmatrix} = \frac{1}{3}(7 + 9 + 3) = \frac{19}{3} = 6,33 = 6$$

$$\text{Kolom 2} = \begin{bmatrix} 4 \\ 1 \\ 7 \end{bmatrix} = \frac{1}{3}(4 + 1 + 7) = \frac{12}{3} = 4$$

$$\text{Kolom 3} = \begin{bmatrix} 2 \\ 5 \\ 2 \end{bmatrix} = \frac{1}{3}(2 + 5 + 2) = \frac{9}{3} = 3$$

$$\text{Kolom 4} = \begin{bmatrix} 8 \\ 4 \\ 6 \end{bmatrix} = \frac{1}{3}(8 + 4 + 6) = \frac{18}{3} = 6$$

Hasilnya adalah *matrix* baris:

$$\text{Mean} = [6 \quad 4 \quad 3 \quad 6]$$

Selanjutnya mencari ciri setiap citra referensi sesuai dengan persamaan (2.3). Proses perhitungannya dapat dituliskan sebagai berikut.

$$\text{Fitur} = \begin{bmatrix} 7-6 & 4-4 & 2-3 & 8-6 \\ 9-6 & 1-4 & 5-3 & 4-6 \\ 3-6 & 7-4 & 2-3 & 6-6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 2 \\ 3 & -3 & 2 & -2 \\ -3 & 3 & -1 & 0 \end{bmatrix}$$

Untuk menghemat komputasi maka nilai yang diambil adalah bilangan bulat positif saja. Maka *matrix* fitur yang dihasilkan menjadi:

$$\text{Fitur} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix}$$

Berikutnya cari *matrix covarians* berdasarkan persamaan (2.4). Pada percobaan ini persamaannya dapat dituliskan:

$$\text{Cov} = \text{Fitur} \times \text{Fitur}^T$$

$$\text{Cov} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

$$\text{Cov} = \begin{bmatrix} 1+0+0+4 & 3+0+0+0 & 0+0+0+0 \\ 3+0+0+0 & 9+0+4+0 & 0+0+0+0 \\ 0+0+0+0 & 0+0+0+0 & 0+9+0+0 \end{bmatrix} = \begin{bmatrix} 5 & 3 & 0 \\ 3 & 13 & 0 \\ 0 & 0 & 9 \end{bmatrix}$$

Setelah menemukan *matrix covarians* maka akan dicari nilai *eigen* dan *vector eigen*. Nilai *eigen* (λ) dan *vector eigen* dapat dihitung berdasarkan persamaan (2.5). Berikut perhitungan nilainya.

Nilai *eigen* dari *matrix covarians* didapatkan dari persamaan:

$$\text{Det } [\lambda I - \text{Cov}] = 0$$

$$\det \begin{bmatrix} \lambda - 5 & -3 & 0 \\ -3 & \lambda - 13 & 0 \\ 0 & 0 & \lambda - 9 \end{bmatrix} = 0$$

Persamaan karakteristik :

$$(\lambda - 9)\{(\lambda - 5)(\lambda - 13) - 9\} = 0$$

$$(\lambda - 9)\{\lambda^2 - 18\lambda + 56\} = 0$$

$$(\lambda - 9)(\lambda - 4)(\lambda - 14)$$

Maka nilai λ_1 adalah :

$$\lambda - 4 = 0$$

$$\lambda_1 = 4$$

$$\lambda - 9 = 0$$

$$\lambda_2 = 9$$

$$\lambda - 14 = 0$$

$$\lambda_3 = 14$$

Maka nilai *eigen* atau *eigenvalue* dari *matrix covarians* adalah: $\begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 14 \end{bmatrix}$

Kemudian *vector eigen* akan dicari berdasarkan persamaan (2.5). *Vector eigen* didapatkan dengan perhitungan:

$$\begin{bmatrix} 5 - \lambda & 3 & 0 \\ 3 & 13 - \lambda & 0 \\ 0 & 0 & 9 - \lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Untuk $\lambda = 4$

Persamaan linearnya dituliskan:

$$x_1 + 3x_2 + 0 = 0$$

$$3x_1 + 9x_2 + 0 = 0$$

$$0 + 0 + 5x_3 = 0$$

$$x_1 = -3x_2$$

$$x_3 = 0$$

Vector eigen yang sesuai adalah $X = \begin{bmatrix} -3x_2 \\ x_2 \\ 0 \end{bmatrix}$

Misalkan $x_2 = t$ maka *vector eigen*-nya menjadi $X = \begin{bmatrix} -3t \\ t \\ 0 \end{bmatrix}$

Untuk $\lambda = 9$

Persamaan linearnya dituliskan:

$$-4x_1 + 3x_2 + 0 = 0$$

$$3x_1 + 4x_2 + 0 = 0$$

$$0 + 0 + 0 = 0$$

$$4x_1 = 3x_2$$

$$x_1 = \frac{3}{4}x_2$$

$$3x_1 = -4x_2$$

$$x_2 = -\frac{3}{4}x_1$$

Tidak ada solusi, maka *vector eigen* dicari dengan persamaan $AX = \lambda X$ dengan A adalah *matrix covarians*, X adalah *eigen vector* dan λ adalah nilai *eigen*.

$$\begin{bmatrix} 5 & 3 & 0 \\ 3 & 13 & 0 \\ 0 & 0 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 9 \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Dipilih nilai $X = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

$$\text{Maka : } \begin{bmatrix} 5 & 3 & 0 \\ 3 & 13 & 0 \\ 0 & 0 & 9 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 9 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 9 \end{bmatrix}$$

Untuk $\lambda = 9$ *eigen vector*nya adalah $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

Untuk $\lambda = 14$

Persamaan linearnya dituliskan:

$$-9x_1 - 3x_2 + 0 = 0$$

$$3x_1 - x_2 + 0 = 0$$

$$0 + 0 - 5x_3 = 0$$

$$3x_1 = x_2$$

Vector *eigen* yang sesuai adalah $X = \begin{bmatrix} x_1 \\ 3x_1 \\ 0 \end{bmatrix}$

Misalkan $x_1 = t$ maka *vector eigen*-nya menjadi $X = \begin{bmatrix} t \\ 3t \\ 0 \end{bmatrix}$

Untuk penyelesaian diambil suatu nilai $t = 0,3$, maka pasangan nilai *eigen* dan *vector eigen*-nya adalah:

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 14 \end{bmatrix} \text{ dan } \begin{bmatrix} -0,9 & 0 & 0,3 \\ 0,3 & 0 & 0,9 \\ 0 & 1 & 0 \end{bmatrix}$$

Selanjutnya mencari *eigenface* dari citra referensi dengan cara mengalikan Fitur^T dengan *vector eigen* dari *matrix Cov*.

$$\text{Eig}_f = \text{Fitur}^T \times \text{vector eigen}$$

$$\text{Eig}_f = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 3 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -0,9 & 0 & 0,3 \\ 0,3 & 0 & 0,9 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\text{Eig}_f = \begin{bmatrix} -0,9 + 0,9 + 0 & 0 + 0 + 0 & 0,3 + 2,7 + 0 \\ 0 + 0 + 0 & 0 + 0 + 3 & 0 + 0 + 0 \\ 0 + 0,6 + 0 & 0 + 0 + 0 & 0 + 1,8 + 0 \\ -1,8 + 0 + 0 & 0 + 0 + 0 & 0,6 + 0 + 0 \end{bmatrix}$$

$$\text{Eig}_f = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 3 & 0 \\ 0,6 & 0 & 1,8 \\ -1,8 & 0 & 0,6 \end{bmatrix}$$

Dapat diperhatikan nilai setiap kolom dari *matrix Eig_f* adalah nilai *eigenface* masing-masing citra referensi. Nilai *eigenface* terbesar selalu berada paling kanan, semakin ke kiri semakin kecil. Langkah berikutnya adalah menentukan nilai N, N adalah *eigenface* terbesar yang akan mewakili seluruh citra referensi. Misal $N=2$, maka:

$$\text{Eig}_{fN} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \\ 1,8 & 0 \\ 0,6 & 0 \end{bmatrix}$$

Langkah selanjutnya mencari bobot dari masing-masing citra referensi. Hal tersebut dilakukan dengan cara:

$$W = \text{Fitur} \times \text{Eig_fN}$$

$$W = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 3 & 0 \\ 0 & 3 \\ 1,8 & 0 \\ 0,6 & 0 \end{bmatrix}$$

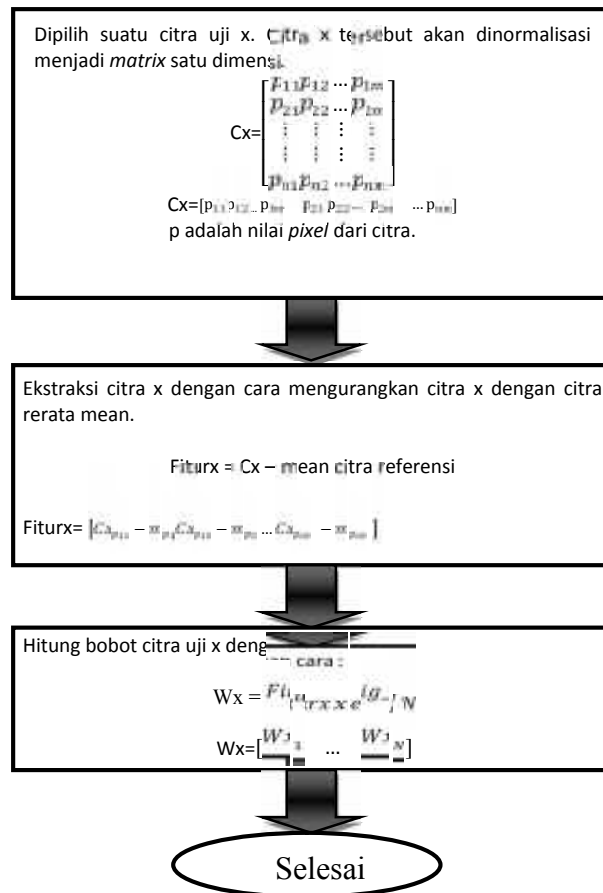
$$W = \begin{bmatrix} 3 + 0 + 0 + 1,2 & 0 + 0 + 0 + 0 \\ 9 + 0 + 3,6 + 0 & 0 + 0 + 0 + 0 \\ 0 + 0 + 0 + 0 & 0 + 9 + 0 + 0 \end{bmatrix}$$

$$W = \begin{bmatrix} 4,2 & 0 \\ 12,6 & 0 \\ 0 & 9 \end{bmatrix}$$

Setiap baris dari *matrix* W adalah bobot dari satu citra referensi, yang urutannya baris pertama adalah bobot citra referensi A, baris kedua adalah bobot dari citra referensi B dan baris ketiga adalah bobot dari citra referensi C. Dengan ditemukannya *matrix* W ini maka ekstraksi ciri utama citra referensi selesai.

4.1.2.2 Ekstraksi Ciri Citra Uji

Berikut ini merupakan alur proses ekstraksi ciri citra uji dengan menggunakan algoritma *eigenface*.



Gambar 4.2 Diagram Alur Proses ekstraksi Ciri Citra Uji

Berikutnya adalah pengujian dengan memberi citra uji X . Citra X juga harus memiliki ukuran yang sama dengan citra referensi. Misal citra X :

$$X = \begin{bmatrix} 8 & 2 \\ 5 & 5 \end{bmatrix}$$

Citra X akan dirubah ke bentuk *matrix* baris.

$$X = [8 \ 2 \ 5 \ 5]$$

Langkah selanjutnya adalah mencari fitur dari citra X dengan cara:

$$\text{Fiturx} = X - \text{mean citra referensi}$$

$$\text{Fiturx} = [8-6 \ 2-4 \ 5-3 \ 5-6]$$

$$\text{Fiturx} = [2 \ -2 \ 2 \ -1]$$

Untuk memudahkan komputasi maka nilai yang diambil selalu bilangan bulat positif. Maka:

$$\text{Fitur}_x = [2 \ 0 \ 2 \ 0]$$

Berikutnya mencari bobot citra uji X dengan cara mengalikan fitur citra uji X dengan *eigenface* terpilih.

$$W_x = \text{Fitur}_x \times \text{eig}_f N$$

$$W_x = [2 \ 0 \ 2 \ 0] \times \begin{bmatrix} 3 & 0 \\ 0 & 3 \\ 1,8 & 0 \\ 0,6 & 0 \end{bmatrix}$$

$$W_x = [6+0+3,6+0 \ 0+0+0+0]$$

$$W_x = [9,6 \ 0]$$

Dengan ditemukannya *matrix* W_x yang berisi bobot dari citra uji X maka ekstraksi ciri utama dari citra uji selesai.

4.1.2.3 Proses Pengenalan Citra Menggunakan *Euclidean Distance*

Dari dua perhitungan di atas dihasilkan dua buah *matrix* yang berisi bobot masing-masing citra yaitu *matrix* W dan W_x . Konsep jarak digunakan sebagai indikasi pengenalan. Perhitungan jarak sesuai persamaan (2.13). Citra yang mempunyai jarak terdekat diasumsikan mirip.

$$W = \begin{bmatrix} 4,2 & 0 \\ 12,6 & 0 \\ 0 & 9 \end{bmatrix} \text{ dan } W_x = [9,6 \ 0]$$

Elemen *matrix* W_x akan dibandingkan dengan setiap baris dari *matrix* W yang merupakan bobot dari masing-masing citra referensi. Berikut perhitungan jarak *euclidean* antara citra uji dengan citra-citra referensi.

Jarak pertama (D_1) adalah jarak antara citra uji X terhadap citra referensi A.

$$D_1 = \sqrt{(9,6 - 4,2)^2 + (0 - 0)^2} = 5,4$$

Jarak kedua (D_2) adalah jarak antara citra uji X terhadap citra referensi B.

$$D_2 = \sqrt{(9,6 - 12,6)^2 + (0 - 0)^2} = 3$$

Jarak ketiga (D_3) adalah jarak antara citra uji X terhadap citra referensi C.

$$D_3 = \sqrt{(9,6 - 0)^2 + (0 - 9)^2} = 13,15$$

Jarak *euclidean* terkecil adalah D_2 yaitu sebesar 3. Berikut perbandingan citranya:

$$X = \begin{bmatrix} 8 & 2 \\ 5 & 5 \end{bmatrix}, B = \begin{bmatrix} 9 & 1 \\ 5 & 4 \end{bmatrix}$$

Kedua *matrix* tersebut yang diasumsikan sebagai sebuah citra memang memiliki elemen yang nilainya berdekatan dibandingkan dengan *matrix* lainnya.

4.2 Penggambaran Perancangan

Penerapan dari pengenalan wajah ini akan digambarkan sebagai berikut:

4.2.1 Penggambaran Metode *Eigenface*

Data *input* yang disimpan kedalam matrik dari hasil proses deteksi wajah akan dirata-ratakan nilainya dengan seluruh nilai matrik per tiap *pixel*. Karena matrik ini menyimpan data citra wajah yang sudah terdeteksi dan dijadikan sebagai data *input*. Setiap citra ini selanjutnya akan diproyeksikan terhadap nilai rata-rata untuk mendapatkan perbedaan setiap citra kepada nilai rata-ratanya. Hasil dari pengurangan ini bisa disebut sebagai *mean subtracted image* berdimensi $M \times N^2$. *Mean subtracted image* akan menjadi data input untuk menghitung *covariance matrix* dengan mengalikan masing-masing *mean subtracted image* dengan nilai transposnya sendiri. Dimensi dari *covariance matrix* ini akan sebesar $M \times N^2$.

Menggunakan matlab, *covariance matrix* ini digunakan untuk menghitung nilai *eigen* dan *eigenvector*. Hasil dari *eigenvalue* dan *eigenvector* ini selanjutnya akan diurutkan dari terbesar sampai ke yang terkecil. Pengurutan ini bertujuan untuk mendapatkan nilai *eigenface* yang jelas menonjol dan lebih besar dari yang lain.

Proses selanjutnya mengalikan nilai *eigenvector* dengan nilai citra yang didapat dari proses *face detection* yang sudah dikurangi dengan rata-rata. Nilai dari hasil perkalian ini perlu normalisasi agar tidak melebihi dari batas warna hitam yaitu 0, dengan mengubah yang lebih kecil dari 0 akan diset menjadi 0. Hasilnya berupa *eigenface*. Dengan mengalikan nilai *eigenface* dengan nilai citra yang didapat dari proses *face detection* yang sudah dikurangi dengan rata-rata kembali maka akan didapat nilai *eigenface*-nya. *Eigenface* ini yang akan menjadi pembanding dengan nilai *input* baru untuk mencari kecocokan dari objek yang dibandingkan.

4.2.2 Penggambaran Metode Pengenalan Citra

Proses pembandingan akan menggunakan metode *euclidean distance*. Dan hasil yang paling minimal dari hasil perbandingan *eigenface* bisa dikatakan mempunyai beda yang paling kecil, dan dapat dianggap sebagai wajah yang sama.

4.2.3 Perancangan Interface

Berikut ini *form interface* pengenalan wajah yang akan dirancang dalam pembuatan perangkat lunak pengenalan wajah berdasarkan penampilan (*appearance based*) menggunakan metode *eigenface* yang berorientasi pada *principal component analysis* (PCA).

<p>Perangkat Lunak Pengenalan Wajah Menggunakan Metode Eigenface dan Euclidean Distance Teknik Informatika 2011</p> <hr/>
<p>Menu Utama</p> <p>*****</p>
<p>Pilih Perintah</p> <p>*****</p>
<p>Hitung Eigenface.....[1] Input Gambar Untuk Pengenalan.....[2] Update Database.....[3] Hapus Database.....[4] Keluar Program.....[Q]</p>
<p>Pilih dan tekan perintah yang ingin dijalankan.....</p>

Gambar 4.3 Rancangan *Form Interface* Pengenalan Wajah

Perangkat lunak pengenalan wajah ini dikembangkan berdasarkan perintah-perintah. Jadi apabila ingin menjalankan suatu perintah tinggal menekan tombol sesuai dengan perintah tersebut.

Berikut merupakan penjelasan dari tiap perintah yang dapat dijalankan.

1. Hitung Eigenface [1]

Merupakan perintah untuk menghitung nilai dari *eigenface* dari gambar gambar yang berada pada *database*. Untuk menjalankan perintah ini tinggal menekan angka 1 pada *keyboard*.

2. Input Gambar Untuk Pengenalan [2]

Perintah untuk memilih gambar yang akan dikenali oleh sistem. Untuk menjalankan perintah ini tinggal menekan angka 2 pada *keyboard*.

3. Update Database [3]

Perintah untuk meng-*update* gambar pada *database* perangkat lunak pengenalan wajah. Untuk menjalankan perintah ini tinggal menekan angka 3 pada *keyboard*.

4. Hapus Database [4]

Merupakan perintah untuk menghapus seluruh gambar yang berada pada *database*. Untuk menjalankan perintah ini tinggal menekan angka 4 pada *keyboard*.

5. Keluar Program [Q]

Perintah untuk keluar dari perangkat lunak pengenalan wajah. Untuk menjalankan perintah ini tinggal menekan tombol Q pada *keyboard*.

BAB V

IMPLEMENTASI DAN PENGUJIAN

Setelah melakukan beberapa tahap pengembangan perangkat lunak pada bab analisa dan perancangan, maka tahap pengembangan perangkat lunak pengenalan wajah selanjutnya adalah Implementasi dan Pengujian.

5.1 Implementasi

Implementasi merupakan lanjutan dari tahap perancangan yaitu aplikasi siap dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah aplikasi yang dibuat telah menghasilkan tujuan yang diinginkan. Pada bagian ini diberikan gambaran mengenai implementasi perangkat lunak pengenalan wajah berdasarkan hasil perancangan yang telah dibuat pada bab sebelumnya. Pada bab ini meliputi batasan implementasi, lingkungan implementasi, serta implementasi antarmuka hasil perancangan.

5.1.1 Batasan Implementasi

Batasan untuk implementasi perangkat lunak pengenalan wajah adalah sebagai berikut:

1. Tidak membahas tahapan *pre-processing* pengambilan citra wajah.
2. Ukuran citra wajah yang akan dijadikan citra *referensi* dan citra uji berukuran sama. Dalam tugas akhir ini dibatasi sebesar 180 x 200 *pixel*.
3. Hanya citra wajah yang memiliki *format* .JPG saja yang bisa di *input*-kan ke dalam sistem.

5.1.2 Lingkungan Implementasi

Lingkungan implementasi sistem ada dua yaitu lingkungan perangkat keras dan lingkungan perangkat lunak.

5.1.2.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan mempunyai spesifikasi sebagai berikut:

1. *Processor* Pentium IV 2.4 GHz
2. Memori RAM minimal 512 MB
3. *Hard Disk* minimal 80 GB
4. Monitor
5. *Keyboard*
6. *Mouse*

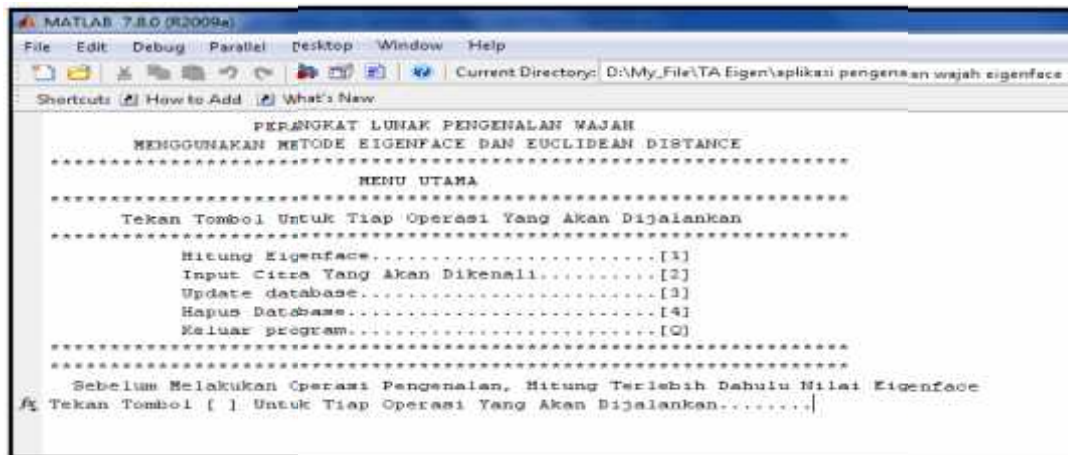
5.1.2.2 Lingkungan Perangkat Lunak

Perangkat lunak Pengenalan wajah dikembangkan dengan menggunakan:

1. Sistem operasi *Microsoft Windows Xp Professional Service Pack 2*.
2. Bahasa pemrograman Matlab 7.8.0
3. Adobe Photoshop CS3.

5.1.3 Implementasi Antarmuka

Rancangan antarmuka perangkat lunak pengenalan wajah diimplementasikan dengan menggunakan Matlab 7.8.0. Berikut ini akan dijelaskan secara umum seluruh *interface* yang terdapat pada perangkat lunak pengenalan wajah yang meliputi:



Gambar 5.1 Tampilan Menu Utama Aplikasi Pengenalan Wajah

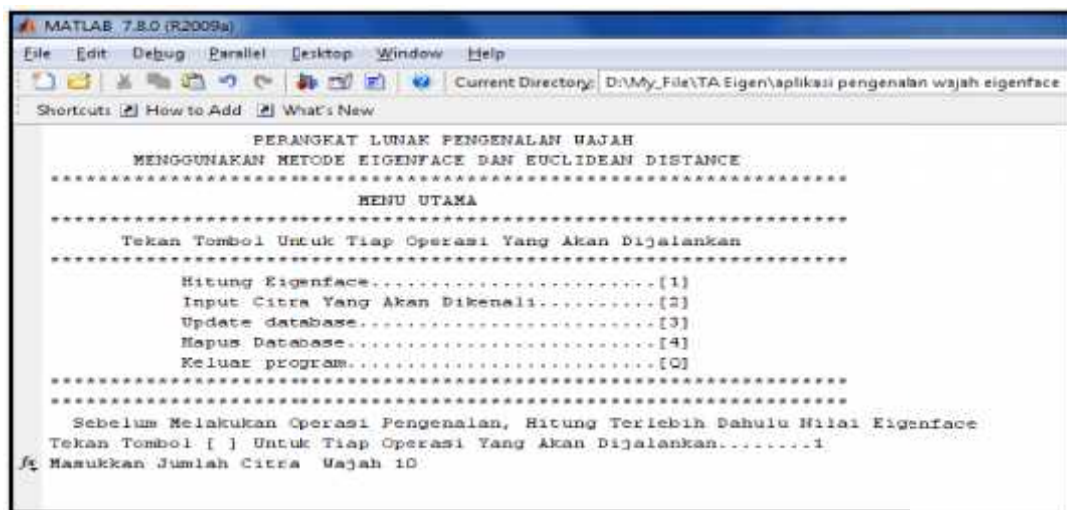
Form ini digunakan sebagai *form* utama dalam proses pengenalan wajah. Apabila ingin menjalankan operasi-operasi yang tersedia pada menu program maka tinggal menekan tombol yang telah disediakan.

Berikut merupakan penjelasan tombol untuk tiap-tiap operasi yang siap dijalankan:

- | | | |
|----|--|---|
| 1. | Untuk operasi Hitung <i>Eigenface</i> tekan tombol | 1 |
| 2. | Untuk operasi <i>Input</i> Citra Yang Akan Dikenali tekan tombol | 2 |
| 3. | Untuk operasi <i>Update Database</i> tekan tombol | 3 |
| 4. | Untuk operasi Hapus <i>Database</i> tekan tombol | 4 |
| 5. | Untuk Keluar dari program tekan tombol | Q |

5.1.3.1 Tampilan Proses Hitung *Eigenface*

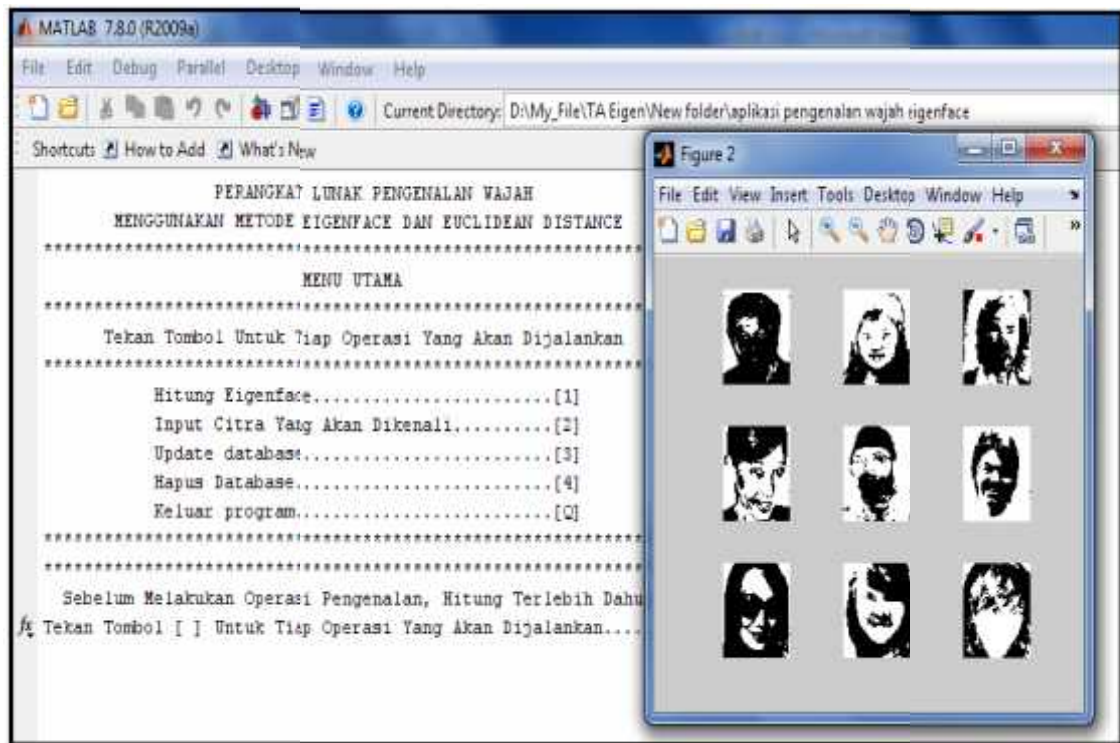
Apabila tombol 1 ditekan untuk proses hitung nilai *eigenface* maka akan muncul pertanyaan Masukkan Jumlah Citra Wajah yang akan kita hitung nilai *eigenface*-nya. Selanjutnya program akan membaca semua citra wajah yang berada dalam folder. Kemudian dari masing-masing citra wajah akan dimasukkan kedalam matrik untuk dicari nilai *eigenface*-nya. Nilai *eigen* ini disimpan dan akan dipergunakan untuk proses selanjutnya.



Gambar 5.2 Tampilan Menu Proses Hitung *Eigenface*

	120	24	12	7	19	17	59
	121	28	18	7	24	17	57
	121	35	26	8	26	15	56
	119	40	32	8	26	13	54
	118	51	42	7	22	15	56
	118	58	43	7	25	15	56
	117	67	45	9	33	16	56
	117	70	46	11	45	16	56
	117	66	45	13	56	15	56
	118	72	44	15	62	15	56
	119	97	44	16	61	13	56
	119	124	44	17	58	13	56
	117	104	47	19	57	14	56
	116	59	47	20	58	17	56
	115	27	47	21	59	21	56
	114	18	47	22	60	24	56
	110	29	54	13	22	108	68
	110	28	54	14	20	119	68
	110	28	54	15	16	128	68
	110	31	54	15	15	126	68

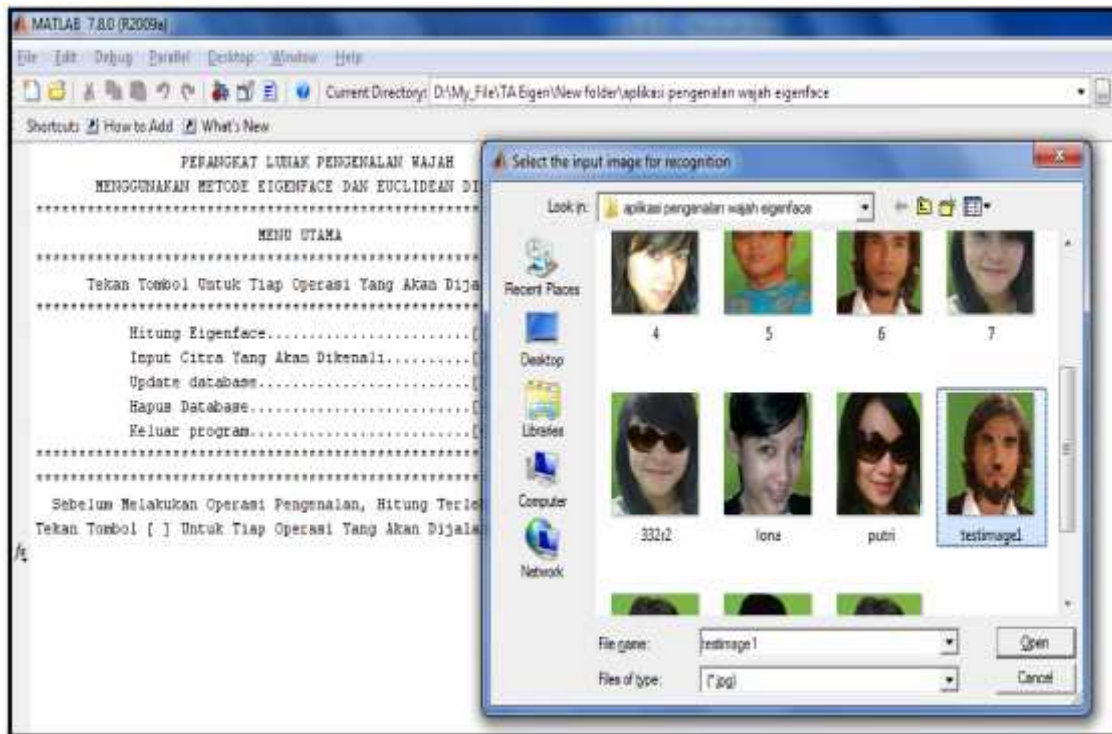
Gambar 5.3 Tampilan Jalannya Proses Perhitungan Nilai *Eigenface*



Gambar 5.4 Tampilan Citra Wajah Dalam Bentuk *Eigenface*

5.1.3.2 Tampilan Input Citra Yang Akan Dikenali

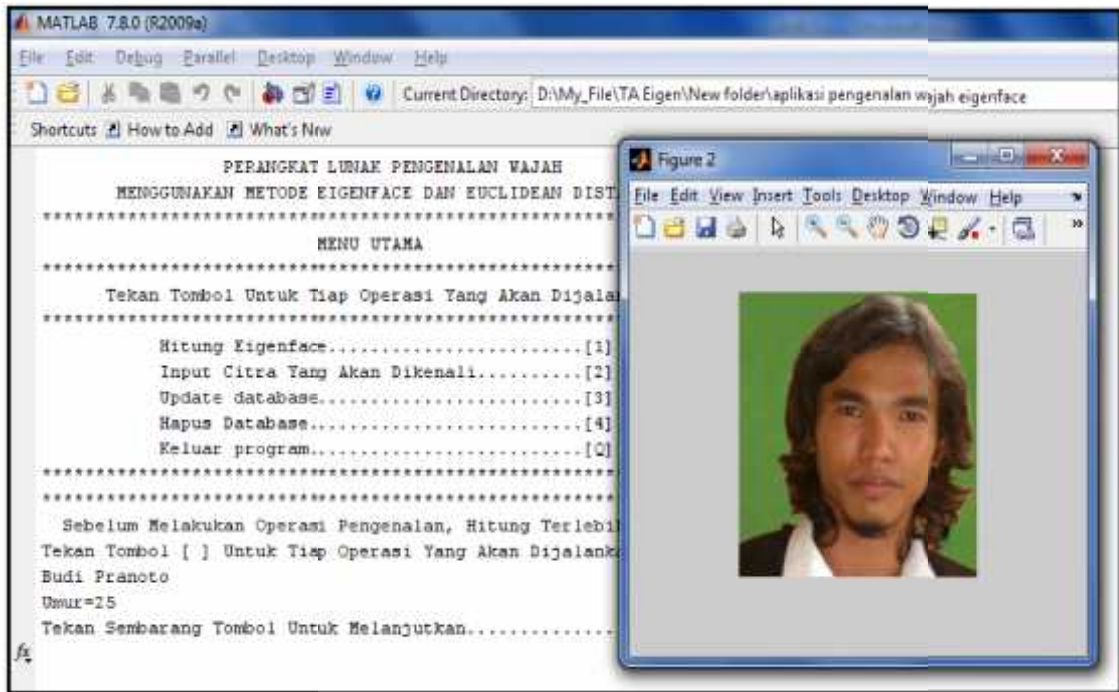
Setelah nilai *eigenface* dari citra wajah diketahui, maka operasi *input* citra yang akan dikenali baru bisa dijalankan. Untuk proses ini tekan tombol 2 pada *keyboard* selanjutnya akan muncul *dialog box* dan memilih citra yang dijadikan *testing image*.



Gambar 5.5 Tampilan Input Citra Yang Akan Dikenali

Setelah memilih *testing image* dari citra yang telah disediakan maka sistem akan menghitung nilai *euclidean distance* dari citra tersebut. Proses perhitungan nilai *euclidean distance* dilakukan dengan membandingkan nilai *eigenface* dari *testing image* dengan citra yang dijadikan *database*.

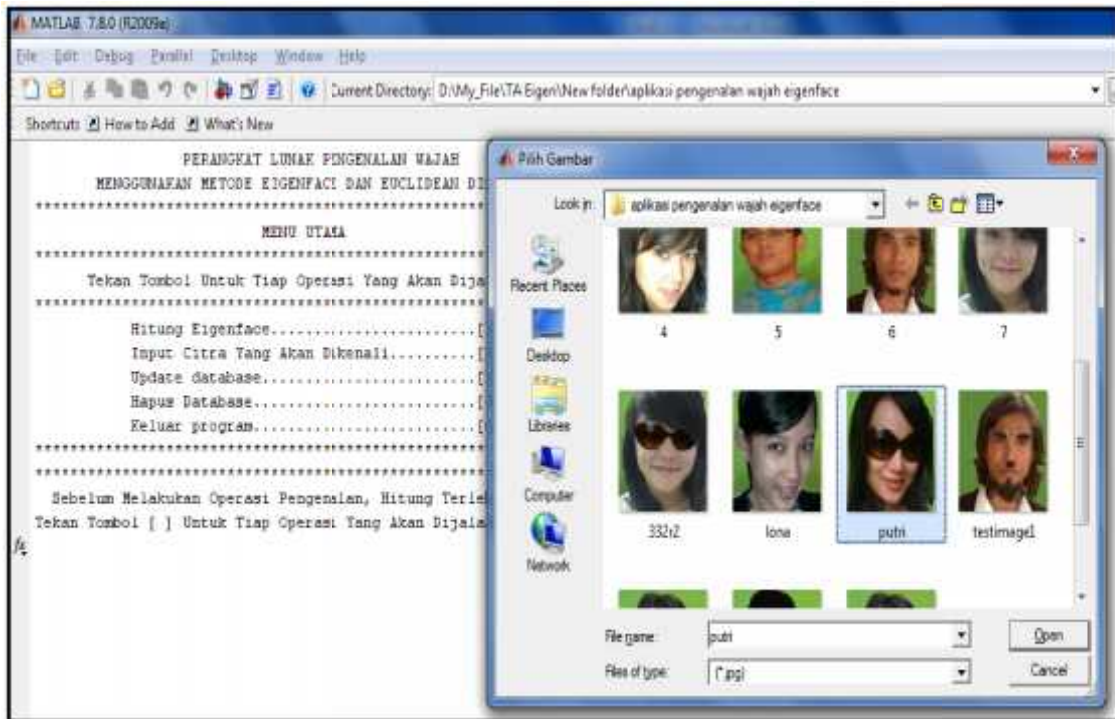
Apabila ditemukan kecocokan maka akan muncul foto citra yang berada pada *database* disertai dengan nama dan umur dari orang tersebut. Namun apabila tidak ditemukan kecocokan maka akan muncul pernyataan tidak ditemukan kecocokan, tekan sembarang tombol untuk melanjutkan.



Gambar 5.6 Tampilan Citra Telah Dikenali

5.1.3.3 Tampilan Update Database

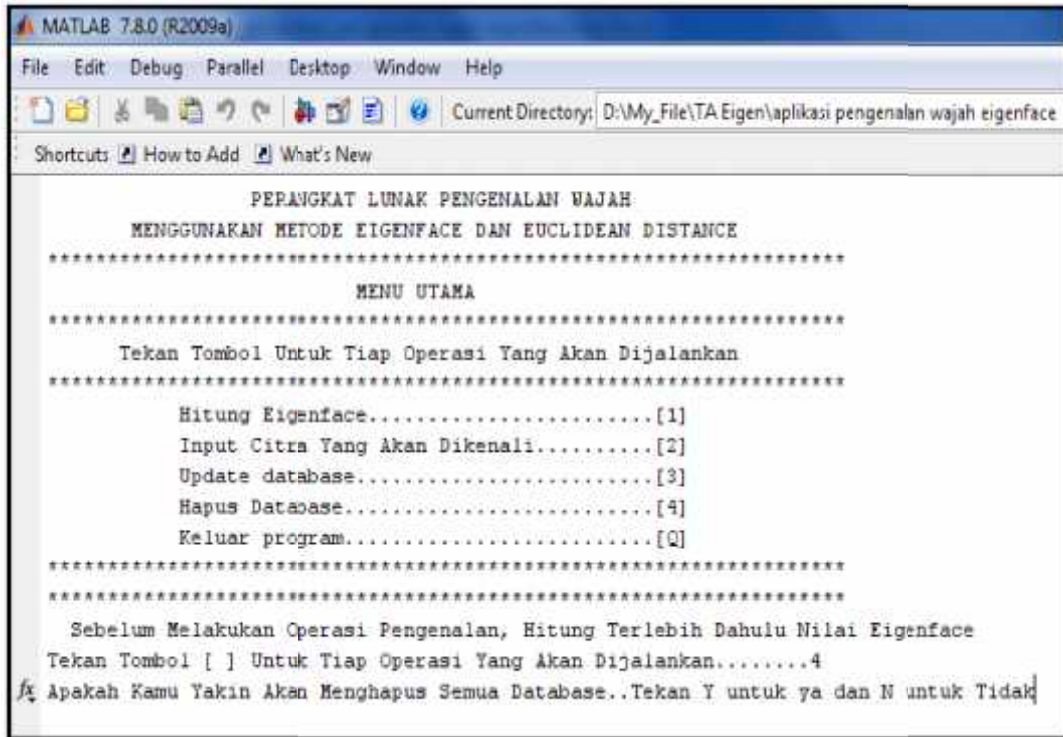
Untuk menjalankan operasi *update database* tekan tombol 3, selanjutnya akan muncul *dialog box* untuk memilih citra wajah yang akan dimasukkan ke dalam sistem. Selanjutnya sistem akan memberikan penomoran pada citra tersebut secara berurutan.



Gambar 5.7 Tampilan *Update Database*

5.1.3.4 Tampilan Hapus Database

Untuk menjalankan operasi hapus *database* kita memilih menekan tombol 4, selanjutnya akan muncul pertanyaan apakah kamu yakin akan menghapus *database*. tekan Y untuk ya dan N untuk tidak. *Database* akan terhapus jika ditekan tombol Y dan sistem akan kembali ke menu semula jika ditekan tombol N.



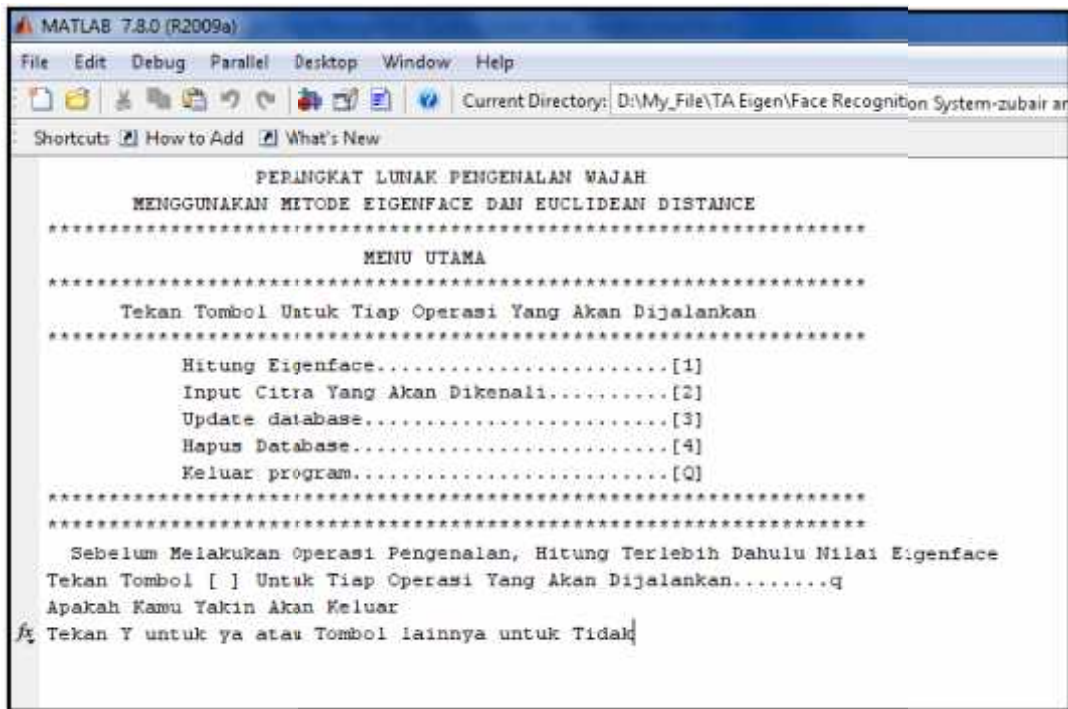
```
MATLAB 7.8.0 (R2009a)
File Edit Debug Parallel Desktop Window Help
Current Directory: D:\My_File\TA Eigen\aplikasi pengenalan wajah eigenface
Shortcuts: How to Add What's New

PEPANGKAT LUNAK PENGENALAN WAJAH
MENGGUNAKAN METODE EIGENFACE DAN EUCLIDEAN DISTANCE
*****
MENU UTAMA
*****
Tekan Tombol Untuk Tiap Operasi Yang Akan Dijalankan
*****
Hitung Eigenface.....[1]
Input Citra Yang Akan Dikenali.....[2]
Update database.....[3]
Hapus Database.....[4]
Keluar program.....[Q]
*****
*****
Sebelum Melakukan Operasi Pengenalan, Hitung Terlebih Dahulu Nilai Eigenface
Tekan Tombol [ ] Untuk Tiap Operasi Yang Akan Dijalankan.....4
fx Apakah Kamu Yakin Akan Menghapus Semua Database..Tekan Y untuk ya dan N untuk Tidak
```

Gambar 5.8 Tampilan Hapus *Database*

5.1.3.5 Tampilan Keluar Program

Untuk menjalankan operasi keluar program kita memilih menekan tombol Q, selanjutnya akan muncul pertanyaan apakah kamu yakin akan keluar. Tekan Y untuk ya tombol lainnya untuk tidak.



Gambar 5.9 Tampilan Keluar Program

5.2 Pengujian

Tahap *testing* dilakukan setelah selesai tahap pembuatan dan seluruh data telah dimasukkan. Suatu hal yang tidak kalah penting yaitu aplikasi harus dapat berjalan dengan baik dilingkungan pengguna. Pengguna merasakan manfaat serta kemudahan dari aplikasi tersebut dan dapat menggunakannya sendiri terutama untuk aplikasi interaktif. Pada tahap pengujian, aplikasi diuji melalui pengujian terhadap citra dan pengujian melalui *blackbox*.

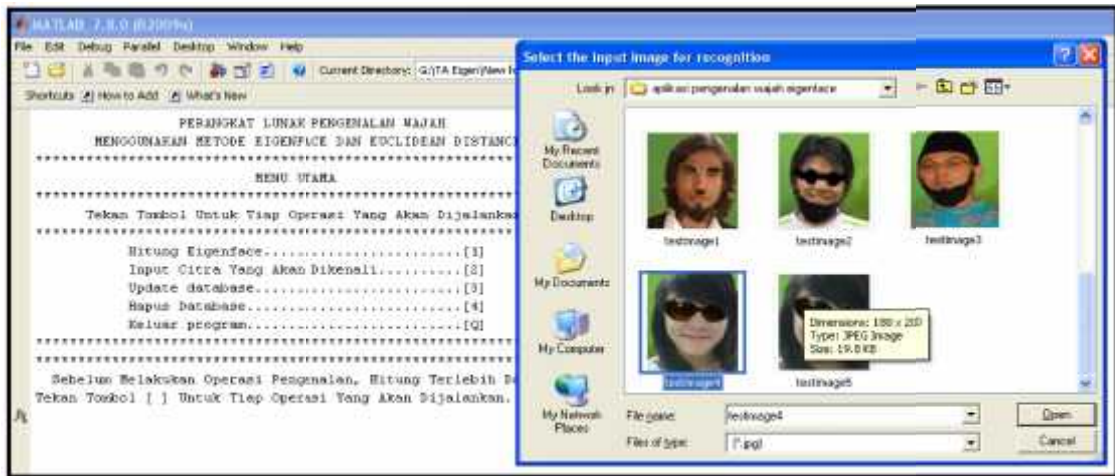
5.2.1. Pengujian Terhadap Citra

Pada aplikasi pengenalan wajah ini jumlah citra yang disediakan pada folder *image* berjumlah tiga belas citra wajah yang terdiri dari sepuluh citra referensi dan tiga citra uji. Ukuran dari semua citra tersebut harus sama yakni 180 x 200 *pixel*, hal ini karena jika ukuran dari citra uji berbeda dengan citra referensi maka tidak akan ditemukan kecocokan dari nilai *eigen* wajah tersebut. Selain ukurannya harus sama sudut pengambilan citra dan pencahayaan dari citra referensi dan citra uji tersebut juga harus sama karena nilai dari tiap *pixel* yang ada pada gambar akan berbeda dan akan mempengaruhi terhadap pengenalan wajah.

Jika ditemukan kesesuaian nilai *eigenface* dari citra uji dengan citra referensi maka sistem akan menampilkan nama dan umur dari citra tersebut beserta gambar wajah dari citra referensi yang telah tersimpan di *database*. Pada aplikasi ini penambahan *feature* pada wajah seperti pemakaian kacamata, munculnya jenggot dan kumis tetap akan dikenali oleh sistem. Namun apabila pencahayaan citra uji di bedakan dengan pencahayaan citra referensi atau sudut pengambilan gambar dan ukuran gambar wajah citra uji diubah maka sistem tidak bisa menemukan nama dan umur dari citra uji tersebut dan muncul pesan citra wajah tidak dikenali.

Berikut merupakan tampilan pengujian aplikasi pengenalan wajah terhadap beberapa citra wajah yang telah ditambahkan asesoris wajah seperti penambahan kacamata hitam, penambahan kumis, penambahan jenggot dan citra wajah yang telah diperbesar dan dirotasi beberapa derajat.

1. Pengujian Terhadap Citra Yang Menggunakan Kaca Mata Hitam



Gambar 5.10 Pengujian Terhadap Citra Yang Menggunakan Kaca Mata Hitam



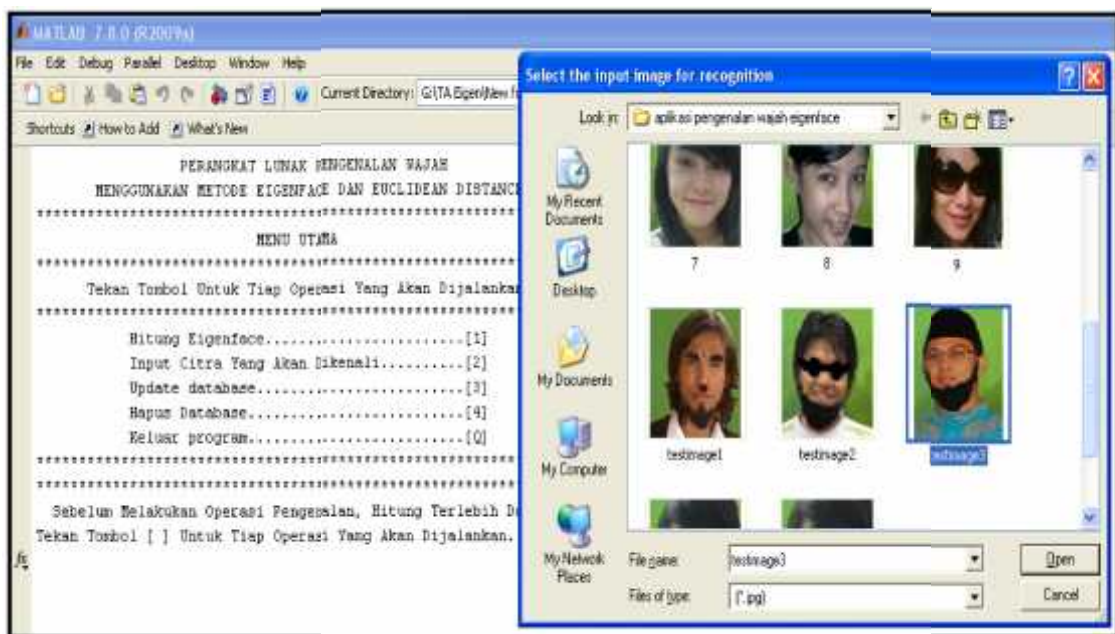
Gambar 5.11 Hasil Pengujian Terhadap Citra Yang Menggunakan Kaca Mata Hitam



Gambar 5.12 Perbandingan Citra Referensi Dan Citra Uji Yang Menggunakan Kaca Mata Hitam

Pada pengujian terhadap citra yang menggunakan kaca mata hitam, sistem masih bisa mengenali citra tersebut dan menampilkan nama dan umur citra wajah tersebut.

2. Pengujian Terhadap Citra Yang Telah Ditambahkan Jenggot



Gambar 5.13 Pengujian Terhadap Citra Yang Telah Ditambahkan Jenggot



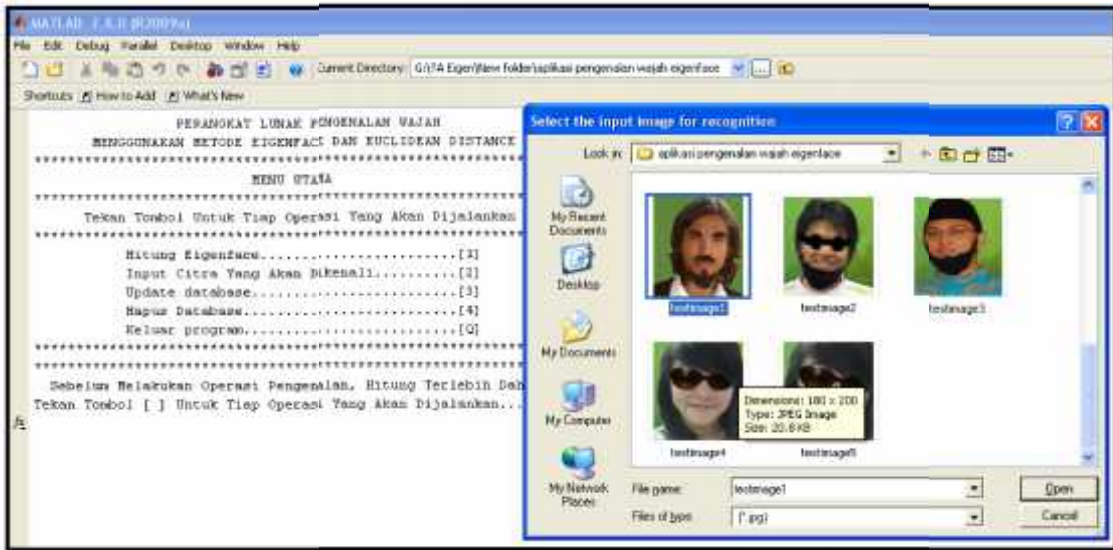
Gambar 5.14 Hasil Pengujian Terhadap Citra Yang Telah Ditambahkan Jenggot



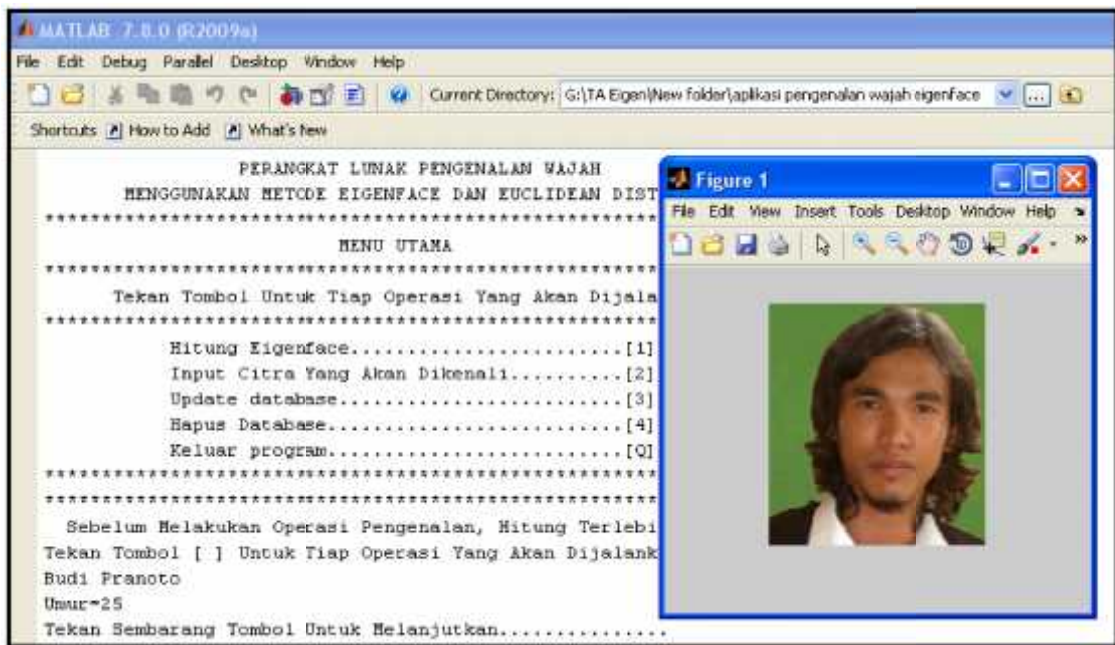
Gambar 5.15 Perbandingan Citra Referensi Dan Citra Uji Yang Telah Ditambahkan Jenggot

Pada pengujian terhadap citra yang telah ditambahkan jenggot, sistem masih bisa mengenali citra tersebut dan menampilkan nama dan umur citra wajah tersebut.

3. Pengujian Terhadap Citra Yang Telah Ditambahkan Kumis



Gambar 5.16 Pengujian Terhadap Citra Yang Telah Ditambahkan Kumis



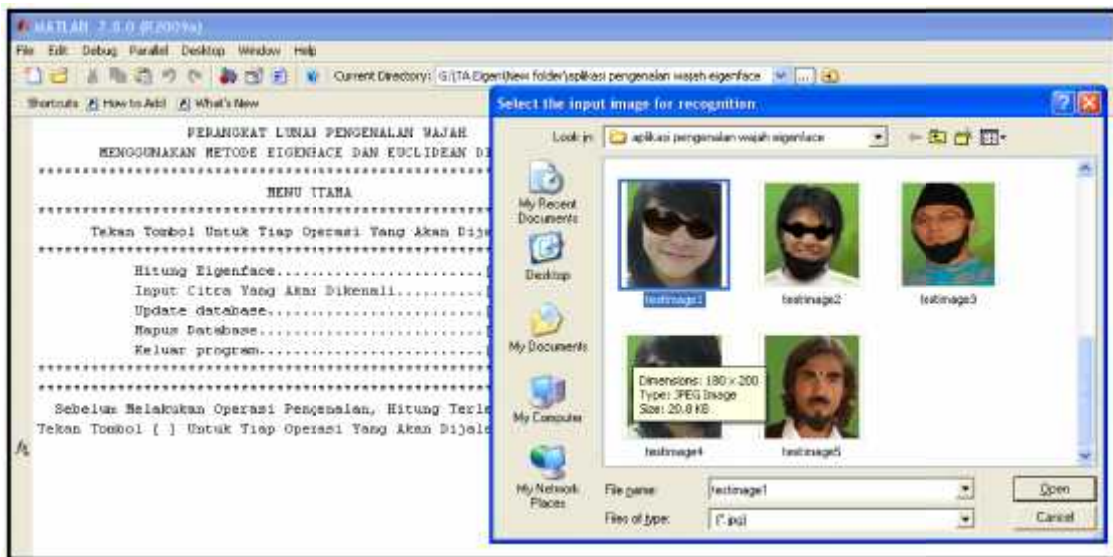
Gambar 5.17 Hasil Pengujian Terhadap Citra Yang Telah Ditambahkan Kumis



Gambar 5.18 Perbandingan Citra Referensi Dan Citra Uji Yang Telah Ditambahkan Kumis

Pada pengujian terhadap citra yang telah ditambahkan kumis, sistem masih bisa mengenali citra tersebut dan menampilkan nama dan umur citra wajah tersebut.

4. Pengujian Terhadap Citra Yang Telah Dirotasi dan Diperbesar



Gambar 5.19 Pengujian Terhadap Citra Yang Telah Dirotasi dan Diperbesar


```

MATLAB 7.8.0 (R2009a)
File Edit Debug Parallel Desktop Window Help
Current Directory: G:\TA Eigen\New folder\aplikasi pengenalan wajah eigenface
Shortcuts How to Add What's New

PERANGKAT LUNAK PENGENALAN WAJAH
MENGUNAKAN METODE EIGENFACE DAN EUCLIDEAN DISTANCE
*****
MENU UTAMA
*****
Tekan Tombol Untuk Tiap Operasi Yang Akan Dijalankan
*****
Hitung Eigenface.....[1]
Input Citra Yang Akan Dikenali.....[2]
Update database.....[3]
Hapus Database.....[4]
Keluar program.....[Q]
*****
Sebelum Melakukan Operasi Pengenalan, Hitung Terlebih Dahulu Nilai Eigenface
Tekan Tombol [ ] Untuk Tiap Operasi Yang Akan Dijalankan.....2
tidak ditemukan kecocokan
Tekan Sembarang Tombol Untuk Melanjutkan.....

```

Gambar 5.20 Hasil Pengujian Terhadap Citra Yang Telah Dirotasi dan Diperbesar



Gambar 5.21 Citra Uji Sebelum Dirotasi dan Diperbesar



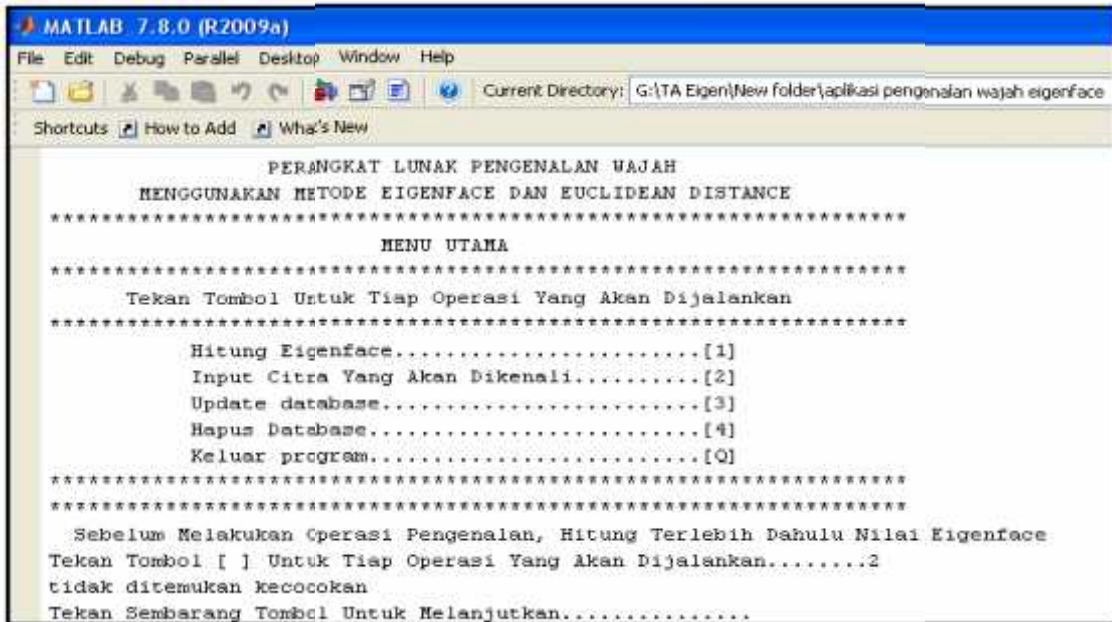
Gambar 5.22 Citra Uji Setelah Dirotasi dan Diperbesar

Pada pengujian terhadap citra yang telah dirotasi dan diperbesar, sistem tidak bisa lagi mengenali citra tersebut dan menampilkan pesan tidak ditemukan kecocokan.

5. Pengujian Terhadap Citra Yang Telah Dinaikkan Kecerahannya.



Gambar 5.23 Pengujian Terhadap Citra Yang Telah Dinaikkan Kecerahannya



Gambar 5.24 Hasil Pengujian Terhadap Citra Yang Telah Dinaikkan Kecerahannya



Gambar 5.25 Citra Uji Sebelum Dinaikkan Kecerahannya



Gambar 5.26 Citra Uji Setelah Dinaikkan Kecerahannya

Pada pengujian terhadap citra yang telah dinaikkan kecerahannya, sistem tidak bisa lagi mengenali citra tersebut dan menampilkan pesan tidak ditemukan kecocokan.

Dari beberapa pengujian terhadap citra pada gambar diatas dapat kita lihat bahwa citra uji yang ditambahkan kacamata, ditambahkan kumis dan jenggot masih dapat dikenali sebagai wajah yang sesuai dengan yang terdapat pada citra referensi oleh sistem. Sedangkan citra uji wajah yang telah mengalami proses perbesaran dan perotasian beberapa derajat dan peningkatan tingkat kecerahan tidak dapat lagi dikenali oleh sistem.

5.2.2. Pengujian *Blackbox*

Tabel 5.1. Butir Uji Pengujian Modul Hitung Nilai *Eigenface* Citra

Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian perhitungan nilai <i>eigenface</i> dari tiap citra yang berada pada folder <i>image</i>	<ol style="list-style-type: none"> 1. Tekan tombol 1 pada <i>keyboard</i>. 2. Masukkan berapa jumlah citra yang akan dihitung nilai <i>eigenface</i>-nya 	Data file wajah sudah tersimpan dalam folder yang telah disediakan.	Data berhasil diproses, tampil proses perhitungan <i>eigenface</i> dan tidak ada instruksi <i>error</i> .	Data berhasil diproses, tampil perhitungan <i>eigenface</i> dan tidak ada instruksi <i>error</i> .	Data berhasil diproses, tampil perhitungan <i>eigenface</i> dan tidak ada instruksi <i>error</i> .	Diterima

Tabel 5.2. Butir Uji Pengujian Modul *Input* Citra Yang Akan Dikenali

Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian modul input citra wajah yang akan dikenali	1.tekan tombol 2 pada <i>keyboard</i> . 2. pilih gambar yang akan dikenali oleh perangkat lunak.	Gambar yang akan dikenali oleh perangkat lunak	Data berhasil diproses tampil citra wajah orang yang berada dalam <i>database</i> beserta nama dan umur.	Data berhasil diproses tampil citra wajah orang yang berada dalam <i>database</i> beserta nama dan umur.	Data berhasil diproses tampil citra wajah orang yang berada dalam <i>database</i> beserta nama dan umur.	Diterima

Tabel 5.3. Butir Uji Pengujian Modul *Update Database*

Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian modul update <i>database</i>	<p>1. Tekan tombol 3 pada <i>keyboard</i>.</p> <p>2. Pilih gambar yang akan dikenali untuk dimasukkan ke <i>database</i> oleh perangkat lunak.</p>	Citra wajah yang akan di masukkan ke <i>database</i>	Citra berhasil dimasukkan, tidak ada <i>error</i> .	Citra berhasil dimasukkan, tidak ada <i>error</i> .	Citra berhasil dimasukkan, tidak ada <i>error</i> .	Diterima

Tabel 5.4. Butir Uji Pengujian Modul Hapus *Database*.

Deskripsi	Prosedur Pengujian	Masukan	Keluaran yang Diharapkan	Kriteria Evaluasi Hasil	Hasil yang didapat	Kesimpulan
Pengujian modul hapus <i>database</i>	1. tekan tombol 4 pada <i>keyboard</i> .	<i>Tombol Y</i> untuk menghapus dan tombol N untuk tidak	Data berhasil di hapus, dan tidak ada <i>error</i> .	Data berhasil di hapus, dan tidak ada <i>error</i> .	Data berhasil di hapus, dan tidak ada <i>error</i> .	Diterima

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan analisa, perancangan dan implementasi pada perangkat lunak pengenalan wajah menggunakan metode *eigenface* dan *euclidean distance* dapat diambil kesimpulan sebagai berikut:

1. Metode *eigenface* merupakan metode pengenalan wajah dengan membandingkan nilai karakteristik dari piksel citra uji dengan citra referensi.
2. Proses pengenalan wajah menggunakan metode *eigenface sensitif* terhadap perubahan cahaya dan orientasi wajah. Jika citra yang digunakan sebagai citra referensi dan sebagai citra uji memiliki intensitas cahaya yang berbeda dan tidak berada pada posisi yang sama maka proses tersebut tidak dapat memberikan hasil yang akurat.
3. Perangkat lunak pengenalan wajah ini masih banyak kekurangan, seperti keterbatasan format gambar. Dalam sistem pengenalan wajah ini baru bisa membaca gambar yang berformat .jpg. selain itu ukuran citra wajah juga dibatasi dengan ukuran 180 x 200 piksel.

6.2 Saran

Beberapa hal yang disarankan dalam pengembangan perangkat lunak pengenalan wajah menggunakan metode *eigenface* dan *euclidean distance* ini adalah sebagai berikut:

1. Pada aplikasi ini menggunakan berkas citra digital dengan format .JPG sebagai citra uji dan citra referensi namun tidak menutup kemungkinan dikembangkan menggunakan berkas citra digital dengan format lain.
2. Pada aplikasi ini citra wajah yang dijadikan referensi untuk tiap-tiap wajah hanya satu, sebaiknya citra referensi wajah seseorang itu ada banyak karena semakin banyak variasi wajah seseorang akan semakin besar pula kemungkinan suatu citra dapat dikenali.

3. Untuk mengatasi kompleksitas penghitungan dan untuk mengurangi resiko pencahayaan serta warna mempengaruhi nilai dari nilai eigen dari suatu citra wajah maka lebih baik citra referensi dan citra uji menggunakan citra dengan format *grayscale*.

DAFTAR PUSTAKA

- Al Fatta, Hanif, "*Rekayasa Sistem Pengenalan Wajah: Membangun Sistem Presensi Karyawan Menggunakan Microsoft Visual Basic 6.0 dan Microsoft Access*", Andi, Yogyakarta, 2009.
- Fadly, Romy, "*Dasar Pengolahan Citra Dengan Matlab*".
- Dewi, Agushinta R, "*Ekstraksi Fitur Dan Segmentasi Wajah Sebagai Semantik Pada Sistem Pengenalan Wajah*", Makalah Skripsi Universitas Gunadarma.
- Lim, Resmana, "*Face Recognition Menggunakan Metode Linear Discriminant Analysis (LDA)*", makalah skripsi Universitas kristen petra , Surabaya, 2002.
- Marti, Ni Wayan, "*Pemanfaatan GUI Dalam Pengembangan Perangkat Lunak Pengenalan Citra Wajah Manusia Menggunakan Metode Eigenface*", Makalah Skripsi Universitas Pendidikan Ganesha, Yogyakarta, 2010.
- Putra, Darma, "*Pengolahan Citra Digital*", Andi, Yogyakarta, 2010.
- Rodiyansyah, Sandy Fajar, "*Ekstraksi Histogram Citra Digital Untuk Mengukur Similarity dengan Menggunakan Metode Euclidian Distance*", 2010.
- Sarbini, "*Perbandingan Metode Eigen Pada Pengenalan Wajah*". Makalah skripsi ilmu komputer FMIPA IPB, Bogor, 2007.