FACE RECOGNITION BERDASAR *EIGENFACE* MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS (PCA)*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Teknik

Disusun Oleh:

GALIH CANDRA SETIAWAN

NIM. 0210630052

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2009

FACE RECOGNITION BERDASAR *EIGENFACE* MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS (PCA)*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Teknik

Disusun Oleh:

GALIH CANDRA SETIAWAN

NIM. 0210630052

Telah diperiksa dan disetujui oleh:

DOSEN PEMBIMBING

Sutrisno Ir., MT	**Suprapto, ST., MT.**
NIP. 19570325 198701 1 001	**NIP. 19701727 199603 1 001**

FACE RECOGNITION BERDASAR *EIGENFACE* MENGGUNAKAN METODE *PRINCIPAL COMPONENT ANALYSIS (PCA)*

Disusun Oleh:

GALIH CANDRA SETIAWAN

NIM. 0210630048

Skripsi ini telah diuji dan dinyatakan lulus pada

tanggal 7 Agustus 2009

DOSEN PENGUJI

Rudy Yuwono, ST, M.Sc
NIP. 19710615 199802 1 003

R. Arief Setyawan, ST., MT
NIP. 19750819 199903 1 001

Ir. Muhammad Aswin, MT
NIP. 19640626 199002 1 001

Mengetahui,
Ketua Jurusan Teknik Elektro

Heru Nurwasito, Ir., M.Kom.
NIP. 19650402 199002 2 001

PENGANTAR

Alhamdu lillaah, segala puji dan syukur Penyusun panjatkan kepada Alloh Yang Maha Suci dan Maha Tinggi atas segala kemudahan, rahmat, dan hidayahNya yang tiada pernah berhenti, sehingga Penyusun dapat menyelesaikan skripsi yang berjudul "*Face Recognition* Berdasar *Eigenface* Menggunakan *Metode Principal Component Analysis (PCA)*" ini dengan baik.

Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya.

Penyusun ingin mengucapkan terima kasih kepada beberapa pihak yang telah membantu dan mendukung penyelesaian skripsi ini, yaitu:

1. Ibu dan Ayah, serta Adik yang selalu mendo'akan dan mendukung Penyusun dalam menyelesaikan skripsi ini.
2. Bapak Heru Nurwasito, Ir., M.Kom. selaku Ketua Jurusan Teknik Elektro.
3. Bapak Rudy Yuwono, ST., M.Sc. selaku Sekretaris Jurusan Teknik Elektro.
4. Bapak Arief Andy Soebroto, ST., M.Kom. selaku KKDK Teknik Informatika dan Komputer.
5. Bapak Sutrisno, Ir., MT. selaku dosen pembimbing yang telah banyak memberikan bimbingan dan arahan terhadap penyusunan skripsi ini.
6. Bapak Suprapto, ST., MT. selaku dosen pembimbing yang telah banyak memberikan bimbingan dan arahan terhadap penyusunan skripsi ini.
7. Staf Pengajar, Administrasi, dan Ruang Baca Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
8. Wike Astrid, Rizky Noviasri, Indriati dan Eriq Muhammad atas dukungan dan bantuan dalam penelitian.
9. Rekan-rekan kerja di SMA Negeri 3 Malang: Aswin Suharsono, Fariz Andri, Rian Yasri Y., Wahyu Setiya R., Wibisono Sukmo W., dan Yusuf Bahtiar, atas dorongan serta bantuan moral dan materi hingga terselesaikannya skripsi ini.

10. Rekan-rekan seperjuangan: Iswahyudi, Akhmad Adiasta, Adam Zulfikar, Nur Kholis, Isnawan, Zakaria, Farid, Reyhan, Galang, Lutfi, Yanu, Alif, Huda, Alfian, Agung, Ahmad Nasiruddin, atas dorongan dan bantuan moral dan materi hingga terselesaikannya skripsi ini.
11. Dwi Oktavianto W.N., atas dedikasinya dalam membantu rekan-rekan mahasiswa kritis.
12. Rekan-rekan mahasiswa Teknik Elektro Universitas Brawijaya, khususnya angkatan 2002, atas dorongan dan bantuan demi kelancaran penyusunan skripsi ini.
13. Semua pihak yang tidak dapat Penyusun sebutkan satu-persatu di sini, yang telah banyak memberikan bantuan hingga dapat terselesaikannya skripsi ini.

Semoga Alloh membalas dengan limpahan hidayah, berkah, dan kebaikan-kebaikan yang jauh lebih baik daripada yang telah diberikan kepada Penyusun. Hanya balasan dariNya-lah sebaik-baik balasan.

Penyusun menyadari bahwa skripsi ini masih memiliki kekurangan dan jauh dari kesempurnaan. Untuk itu, saran dan kritik yang membangun sangat Penyusun harapkan. Semoga skripsi ini membawa manfaat bagi Penyusun maupun pihak lain yang menggunakannya.

Malang, Agustus 2009

Penyusun

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

GALIH CANDRA SETIAWAN. 2009: Face Recognition Berdasar Eigenface Menggunakan Metode Principal Component Analysis (PCA). Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing: Sutrisno, Ir., MT. dan Suprapto, ST., MT

Diantara banyak metode yang dapat digunakan untuk melakukan pengenalan wajah (face recognition) adalah Principal Component Analysis (PCA) diikuti dengan penggunaan metode Euclidean Distance. PCA digunakan untuk mereduksi matrik citra eigenfaces yang masih berdimensi tinggi menjadi matrik citra berdimensi rendah. Euclidean distance digunakan untuk mengukur tingkat kesamaan nilai antara citra input (test set) dan citra yang tersimpan dalam database (training set).

Pengujian dilakukan menggunakan database citra dari Yale University dengan format GIF, grayscale. Dalam database tersebut dua set data. Yale data set A berisi image dengan tingkat brightness yang berbeda-beda. Yale data set B berisi image dengan ekspresi yang berbeda-beda.

Skenario pengujian diarahkan untuk menguji kemampuan sistem untuk mencocokkan wajah, mengenali wajak dalam ekspresi yang berbeda dan untuk menguji kemampuan sistem mengenali wajah dalam tingkat brightness yang berbeda.

Hasil pengujian menunjukkan bahwa kemampuan sistem mengenali wajah dengan benar dalam mencocokan wajah sebesar 100%, kemampuan sistem mengenali ekspresi berbeda sebesar 82,95 % dan kemampuan sistem mengenali wajah dalam tingkat brightness yang berbeda sebesar 40%

Kata kunci: Face recognition, eigenfaces, PCA, euclidean distance
.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi *face recognition* sangat ditentukan oleh pengembangan algoritma pengolahan citra. Pada tahun 1960 dikembangkan sistem yang menggunakan geometri titik-titik tertentu pada wajah manusia (contohnya mata, hidung, mulut) dan keterhubungan mereka (sudut, panjang, lebar, dan lainnya) untuk melakukan pengenalan wajah. Teknik ini biasa disebut dengan *feature-based face recognition*. Tetapi metode ini dihadapkan pada kendala tingkat akurasi yang belum memuaskan karena tidak melibatkan seluruh unsur-unsur wajah [KRU-04:1].

Kemudian, pada tahun 1980 mulai dikembangkan teknik pengenalan wajah yang baru. Yaitu dengan turut serta memperhitungan penampilan serta tekstur wajah. Seringkali dengan langsung menggunakan input citra wajah dua dimensi pada sistem yang dikembangkan. Teknik ini biasa disebut dengan *template-based face recognition.*

Perdebatan antara teknik *feature-based* dan *template-based* untuk pengenalan wajah akhirnya diselesaikan dengan penelitian yang dilakukan oleh Brunelli dan Poggio yang berhasil membuktikan bahwa teknik *template-based* terbukti lebih superior. Salah satu teknik pengenalan wajah yang dikembangkan pada masa kini pada umumnya merupakan pengembangan dari metode *subspace* yang dicetuskan oleh Turk dan Pentland dengan sebutan *eigenfaces* [ZHA-03: 20].

Salah satu metode pengolahan *eigenface* adalah menggunakan *Principal Component Analysis* (PCA). PCA adalah metode yang pertama dan yang paling berhasil dalam melakukan ekstraksi informasi wajah. PCA memproyeksikan citra wajah berdimensi tinggi menjadi *space* berdimensi rendah. Penurunan dimensi sangat berguna untuk meningkatkan efisiensi proses pengenalan citra [NAV-07:1].

Dalam teknik *template-based face recognition,* untuk mengukur tingkat kemiripan citra, dapat digunakan teori *euclidean distance* atau disebut juga metode *normalized correlation* [LIW-05:1].

Berdasarkan uraian di atas, maka dalam tugas akhir ini akan dikembangkan perangkat lunak yang akan melakukan proses *face recognition* dengan menerapkan metode PCA.

1.2. PERUMUSAN MASALAH

Rumusan masalah yang disusun berdasar permasalahan yang tertulis pada latar belakang adalah sebagai berikut:

1. Melakukan proses *feature extraction* menggunakan metode *Principal Component Analysis* (PCA) terhadap citra *training set* dan *test set.*
2. Menggunakan metode *euclidean distance* untuk melakukan proses *face recognition* dari hasil *feature extraction*.
3. Melakukan pengukuran akurasi sistem dalam melakukan *face recognition.*

1.3. BATASAN MASALAH

Dari permasalahan di atas, berikut ini diberikan batasan masalah untuk menghindari melebarnya masalah yang akan diselesaikan:

1. Input sistem berupa citra wajah manusia dari penampakan depan. Citra wajah akan dinormalisasi oleh sistem untuk kemudian citra wajah hasil normalisasi ini akan digunakan dalam pengenalan wajah manusia.
2. Citra wajah latih memiliki ukuran panjang dan lebar dalam jumlah pixel yang sama dengan citra uji.
3. Citra wajah yang digunakan untuk *training set* dan *test set* adalah citra dengan format GIF *grayscale*.
4. Sistem operasi yang digunakan adalah Microsoft XP Profesional.
5. Aplikasi perangkat lunak dibuat dengan menggunakan bahasa pemrograman Java.

1.4. TUJUAN

Tujuan dari penulisan tugas akhir ini adalah melakukan pengenalan wajah terhadap citra wajah manusia.

1.5. MANFAAT

Manfaat penelitian ini antara lain:

Bagi Penulis

a. Menerapkan ilmu yang telah diperoleh dalam Teknik Elektro konsentrasi sistem informatika dan komputer Universitas Brawijaya.
b. Mengembangkan sistem *face recognition* yang tepat sehingga mendapatkan hasil yang maksimal.

Bagi user

a. Dapat melakukan proses *face recognition* dengan metode PCA secara otomatis dan akurat.

1.6 SISTEMATIKA PENULISAN

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Membahas kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi ini. Kajian pustaka diperlukan untuk melakukan kajian terhadap karya ilmiah yang berkaitan dengan skripsi ini, meliputi penerapan *face recognition* dengan metode PCA. Dasar teori yang diperlukan berdasar kajian pustaka untuk penyusunan skripsi ini adalah konsep *eigenfaces, eigenvalue, eigenvector*, PCA, dan *euclidean distance*.

BAB III Metode Penelitian

Membahas metode yang digunakan dalam penulisan yang terdiri dari studi literatur, perancangan perangkat lunak, implementasi perangkat lunak, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan

Analisis kebutuhan perangkat lunak dan perancangan perangkat lunak.

BAB V Implementasi

Membahas mengenai implementasi dari perancangan perangkat lunak ke dalam algoritma pemrograman.

BAB VI Pengujian dan Analisis

Pengujian perangkat lunak, pengujian akurasi PCA, dan analisis hasil pengujian.

BAB VII Penutup

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran–saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Pada bab ini dibahas mengenai kajian pustaka dan dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai pengembangan aplikasi *face recognition* dengan metode *Principal Component Analysis* (PCA). Dasar teori yang diperlukan untuk penyusunan skripsi ini adalah mengenai *face recognition*, *eigenface*, PCA, dan *euclidean distance*.

2.1 Kajian tentang *Face Recognition*

Teknologi *face recognition* adalah suatu proses perhitungan yang dilakukan oleh komputer dengan tujuan agar komputer dapat mengenali wajah seseorang [AND-08:1]. Teknologi ini dapat digunakan untuk mempercepat proses pencarian informasi mengenai seseorang berdasarkan foto atau gambar wajah seseorang, dengan syarat data yang dicari harus sudah dimasukkan terlebih dahulu ke dalam sistem [INT-09:3].

Face recognition telah menjadi alternatif solusi dalam berbagai bidang yang membutuhkan identifikasi seseorang. Wajah merupakan bagian dari identifikasi biometrik karena merupakan bagian langsung dari tubuh manusia yang tidak mudah untuk dicuri atau diduplikasi seperti halnya metode konvensional yang menggunakan password ataupun kartu [DES-07:1].

Sejak tahun 1960 telah dikembangkan berbagai metode ekstraksi citra wajah. Selanjutnya, hasil ekstraksi akan menjadi objek dilakukannya *recognition* dengan algoritma tertentu. Beberapa objek *recognition* adalah *feature* (objek) istimewa, *contour* (tepi), dan *eigenface*. *Recognition* berdasar objek dikenal dengan istilah *featured based recognition* [HAD-04:3], *recognition* berdasar tepi dikenal dengan istilah *contour matching recognition* [GAN-06:1], sedangkan yang menggunakan *eigenface* dikenal dengan istilah *eigenface-based recognition* [LAW-04:1].

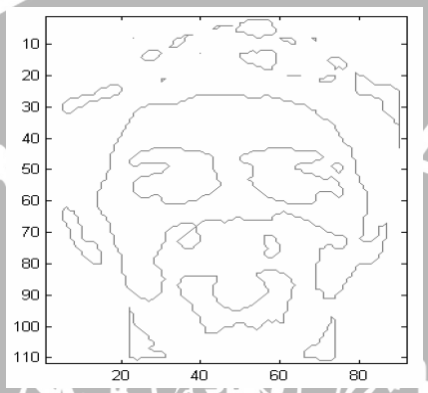
2.1.1 *Featured Based Recognition*

Pendekatan berbasiskan *feature* secara eksplisit menggunakan pengetahuan tentang wajah dan menerapkan metode-metode pendeteksian klasik di mana ciri-ciri dasar diperoleh melalui analisis berbasiskan pengetahuan. Karakteristik wajah yang terlihat jelas, misalnya warna kulit wajah dan geometri wajah menjadi elemen pembeda dalam pendeteksian.

Biasanya metode-metode yang termasuk dalam pendekatan ini dilengkapi dengan pengukuran jarak, sudut, dan luas yang diperoleh dari tampilan wajah [HAD-04:4].

2.1.2 *Contour Matcing Recognition*

Contour adalah batas pertemuan pixel citra yang memiliki tingkat perbedaan intensitas yang tinggi. Dengan mengenali batas pertemuan pixel tersebut, akan diperoleh pola *contour* wajah.



Gambar 2-1Contoh hasil generate contour pada citra wajah

Sumber [GAN-06:4]

Metode *contour matching*, hasil deteksi *contour* akan dilakukan *overlap* dengan *contour* wajah yang ada di dalam database. Kemudian akan dilakukan perhitungan tingkat perbedaan *contour* dengan metode tertentu. Semakin kecil tingkat perbedaan, maka *contour* akan dikenali sebagai wajah yang sama [GAN-06:10].

2.1.3 *Eigenfaces*

Teknik *eigenface* yang pertama kali dikemukakan oleh Turk dan Pentland pada tahun 1991 sangat handal untuk menyelesaikan masalah pengenalan wajah. Teknik eigenface menggunakan lebih banyak informasi dengan mengklasifikasikan wajah berdasar pola wajah yang umum. Pola wajah juga termasuk bagian utama *(featured)* wajah, tetapi bukan hanya bagian itu saja. Dengan menggunakan lebih banyak informasi, analisis *eigenface* secara alamiah lebih efektif daripada teknik *featured based face recognition* [KRU-04:1].

Pada dasarnya, *eigenfaces* adalah vektor dasar untuk wajah. Konsep ini dapat dihubungkan langsung pada konsep yang paling dasar dari teknik elektro, yaitu analisis *fourier*. Analisis *fourier* menunjukkan jika penjumlahan sinusida pada frekuensi yang

berbeda dapat menghasilkan sinyal yang sempurna. Dengan cara yang sama, penjumlahan dari *eigenfaces* dapat dengan sempurna merekonstruksi wajah manusia [KRU-04:1].

2.2 Kaitan *Eigenface* dengan *Principal Component Analysis* (PCA)

Eigenface berdasar *Principal Component Analysis* (PCA) atau yang dikenal sebagai Kahunen-Loven, digunakan sebagai pereduksi dimensi. Dengan metode ini, citra berdimensi tinggi akan diubah menjadi berdimensi rendah. Proses reduksi akan memaksimalkan jarak antara semua citra wajah dalam database. Jarak maksimum ini akan berguna untuk mengklasifikasi citra dalam database.

Misalkan dimiliki citra wajah $I(x,y)$ dengan ukuran pixel n x n dengan intensitas warna sebesar 8 bit *grayscale*. Maka citra tersebut dapat dianalogikan sebagai vektor berdimensi n^2, sehingga citra berdimensi 256 x 256 akan menjadi sebuah vektor yang berdimensi 65.536, atau sama dengan sebuah titik dalam sebuah ruang berdimensi 65.536 yang selanjutnya disebut sebagai *image space*. Kumpulan citra wajah yang berukuran sama umumnya memiliki pola yang sama, sehingga tidak akan terdistribusi secara acak dalam ruang dimensi ini. Dan dapat dideskripsikan menjadi ruang dimensi yang lebih kecil menggunakan teknik PCA. Yaitu menemukan vektor-vektor yang paling sesuai untuk menggambarkan ketersebaran citra wajah pada *image space* [SAP-08:20].

Masing-masing vektor sepanjang n^2 ini akan mendefinisikan citra wajah sebesar *N*, dan merupakan kombinasi linier dari citra wajah asli. Karena vektor-vektor ini merupakan *eigenvector* dari *covariance matrik* dari citra wajah, dan mereka akan nampak seperti wajah, maka mereka disebut sebagai *eigenfaces*.

Untuk mendapatkan *eigenfaces*, diperlukan kumpulan wajah sejumlah *M* yang akan menjadi database wajah-wajah yang dikenal oleh sistem. Seluruh citra wajah haruslah memiliki dimensi yang sama dalam ukuran pixel. Lalu dilakukan konversi tiap citra wajah menjadi vektor kolom Γ_n dengan panjang *N* di mana *N* adalah n x n pixel dari citra wajah. Sehingga akan dimiliki kumpulan citra wajah $(\Gamma_1, \Gamma_2, \ldots \Gamma_n)$

Gambar 2-1Contoh kumpulan citra wajah 8-bit *grayscale* dari 5 individu yang berbeda, citra paling kanan adalah citra rata-rata
sumber [SAP-08: 21]

Tiap individu dapat memiliki lebih dari satu citra wajah, dan tiap individu harus memiliki jumlah citra wajah yang sama. Selanjutnya, koleksi citra wajah yang dimiliki ini disebut dengan sebutan *face space* dengan ukuran dimensi *N*. Selanjutnya diperlukan wajah rata-rata (*average face, ψ*) dari *face space* yang dimiliki dengan menggunakan persamaan:

$$\psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n \tag{2.1}$$

lalu dihitung perbedaan (*difference,Φ*) dari tiap citra wajah Γ dengan ψ

$$\Phi_i = \Gamma_i - \psi \tag{2.2}$$

selanjutnya masing-masing Φ digunakan untuk mendapatkan *covariance matrik* (*C*) dari Φ yang telah dimiliki

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = \frac{1}{M}\sum_{n=1}^{M}\begin{pmatrix} var(p_1) & \cdots & cov(p_1,p_N) \\ \vdots & \ddots & \vdots \\ cov(p_N,p_1) & \cdots & var(p_N) \end{pmatrix}_n = AA^T \tag{2.3}$$

Di mana A = [Φ_1 Φ_2 … Φ_M] dan p_1 adalah pixel ke-I dari wajah ke – n

Lalu dicari *eigenvalue* dan *eigenvector* dari *C*. Karena *C* berdimensi N x N, hal ini menyebabkan dimensi *C* menjadi sangat besar (sebesar jumlah pixel wajah) dan akan memerlukan komputasi yang sangat berat untuk mencari *eigenvalue* dan *eigenvector* sebanyak N dari *C*. PCA menyebutkan bahwa karena hanya dimiliki citra sebanyak *M*, maka hanya akan dimiliki *M* buah *eigenvector* yang memiliki pengaruh dan perlu disimpan sebagai *feature vector*. Oleh karena itu, dibentuk matrik *L* dengan ukuran sebesar M x M dengan perumusan:

$$L_{mn} = \Phi_m^T \Phi_n \tag{2.4}$$

$$L = A^T A \tag{2.5}$$

sehingga dapat diperoleh kumpulan *eigenvector* v_i dan *eigenvalue* L_i dari matrik L yang memenuhi perumusan sebagai berikut ini:

$$Lv_i = L_i v_i \tag{2.6}$$

$$A^T A v_i = L_i v_i \tag{2.7}$$

Dan dengan mengalikan kedua sisi persamaan dengan matrik A

$$AA^T A v_i = L_i A v_i \tag{2.8}$$

yang membuktikan bahwa Av_i adalah *eigenvector* dari $C = AA^T$. Dengan ini, dapat dibentuk *eigenvector* v_i sebanyak *M* dari matrik *L* untuk membentuk sejumlah *M eigenvector* μ_i dari *C* yang selanjutnya disebut sebagai *eigenfaces*. Karena $CAv_i = L_iAv_i$, maka Av_i selanjutnya ditulis sebagai μ_i untuk melambangkan *eigenvector* dari *C*. *Eigenvalue* sejumlah *M* yang dimiliki oleh *L* adalah *M* buah *eigenvalue* terbesar (*feature vector*) dari *C*.

$$\mu_i = Av_i \qquad (2.9)$$

dari *M* buah citra wajah yang masing-masing vektor kolomnya berukuran *N*, didapatkan *M* buah *eigenfaces* seperti gambar 2.2 di bawah ini

Gambar 2-2 *Eigenface* dalam urutan berdasarkan *eigenvalue*-nya dari yang paling besar ke yang paling kecil (dari kiri ke kanan)
sumber [SAP-08:22]

dari *eigenfaces* yang didapat, dapat dilakukan kompresi data menjadi sebuah matrik bobot *W* yang berukuran *P x M* dimana P adalah jumlah *feature vector* yang disimpan, di mana pada umumnya dimensi *W* jauh lebih kecil daripada dimensi ($\Gamma_1, \Gamma_2, \ldots \Gamma_M$).

$$W = \mu^T A \qquad (2.10)$$

Untuk mendapatkan kembali data asli A' (bukan benar-benar data asli *A*, melainkan A'. Karena *A* telah berubah menjadi A' (dikarenakan telah terjadi kompresi PCA) dengan menggunakan hanya *u* dan *W* dan ψ dengan menggunakan persamaan

$$A' = \mu \times W \qquad (2.11)$$

Karena *A'* merupakan matrik yang terbentuk dari vektor kolom Φ_i, lalu masing masing Φ_i pada A' ditambahkan dengan ψ karena sebelumnya nilai Φ_i didapat dari persamaan 2.2 sehingga pada akhirnya matrik A' akan berisi vektor kolom Γ_i, yaitu vektor kolom Γ_i yang telah terkompresi.

$$A_i' = A_i' + \psi \qquad (2.12)$$

2.3 Kajian Tentang Metode Kompresi Matrik Citra menggunakan *Principal Component Analysis* (PCA)

[DEL-06:1] menyatakan *Principal Component Analysis* (PCA) merupakan teknik statistik yang umum digunakan untuk menemukan pola tertentu dalam data berdimensi tinggi (seperti misalnya data yang berupa citra) dan mengekspresikan data dalam tata cara

tertentu untuk mengekspos persamaan dan perbedaan dalam data tersebut. PCA memproyeksikan citra wajah berdimensi tinggi menjadi *space* berdimensi rendah. Penurunan dimensi sangat berguna untuk meningkatkan efisiensi proses pengenalan citra.

Keunggulan PCA yaitu jika pola dalam suatu data telah ditemukan, data tersebut kemudian dapat dikompresi (misalkan mengurangi dimensi data) tanpa kehilangan banyak informasi dari data yang sebenarnya [NAV-07:3]. Sesuai dengan yang disebutkan dalam [SAP-08], langkah-langkah yang dilakukan untuk melakukan PCA pada suatu kumpulan data, yaitu:

1. Mengumpulkan Data

Misalkan dimiliki data 2 dimensi berupa nilai *x* dan *y* yang selanjutnya disebut sebagai *data awal* seperti di bawah ini.

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	0.9

6.1.1.1.1. Tabel 2.1
Sumber [SAP-08:28]

2. Data Normal

Lalu dicari *mean* dari masing-masing dimensi data, dan mengurangi nilai masing-masing elemen data dengan *mean* dari dimensi-nya sendiri. Dari contoh data di atas, dapat dicari nilai $\bar{x}$ dan $\bar{y}$ yaitu masing-masing 1.81 dan 1.91. Selanjutnya data baru ini disebut *data normal*.

x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

6.1.1.1.2. Tabel 2.2
Sumber [SAP-08:18]

3. *Covariance Matrix*

Hitung *covariance matrix* (*C*) dari data normal. Karena data normal yang digunakan di sini berupa data 2 dimensi, maka *C* akan berbentuk matrik 2 dimensi. Dari kumpulan data pada data normal di atas, diperoleh *C* sebagai berikut:

$$\begin{pmatrix} 0{,}616555556 & 0{,}615444444 \\ 0{,}615444444 & 0{,}716555556 \end{pmatrix}$$

Dari nilai elemen non-diagonal pada *C* yang berupa nilai positif, diperoleh informasi bahwa nilai variabel *x* dan *y* pada data awal membesar secara bersamaan.

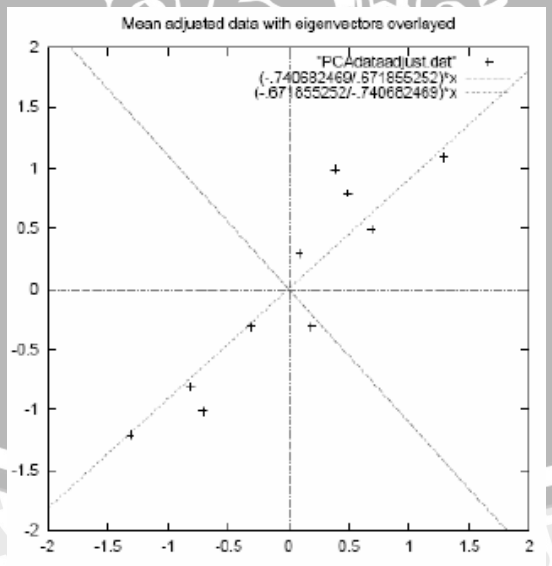
4. *Eigenvalue dan Eigenvector*

Karena *C* adalah berupa *square matrix*, maka dapat dicari *eigenvalue* dan *eigenvector* dari *C*. Karena *C* adalah matrik 2×2, maka dimiliki 2 *eigenvalue* dan 2 *eigenvector*. *Eigenvalue* dan *eigenvector* dari *C* adalah

$$L = \begin{pmatrix} 0{,}0490833989 \\ 1{,}2840277100 \end{pmatrix}$$

$$v = \begin{pmatrix} -0{,}735178656 & -0{,}677873399 \\ 0{,}677873399 & -0{,}735178656 \end{pmatrix}$$

Eigenvector yang diperoleh di sini masing-masing memiliki panjang satu. Bila data normal di atas digambarkan bersama dengan *eigenvector* yang diperoleh, maka akan diperoleh plot data sebagai berikut:



Gambar 2.3 Plot data yang telah di normalisasi dengan *eigenvector* dari *covariance matrix* yang diperoleh dari data normal
Sumber [SAP-08:19]

Perhatikan susunan data pada data normal yang memiliki pola seperti yang digambarkan pada plot data di atas. 2 *eigenvector* yang diperoleh dari *C* juga digambarkan pada plot data berupa garis putus-putus. Dari *eigenvector* yang diperoleh, membentuk 2 buah garis diagonal pada plot.

Salah satu *eigenvector* tergambarkan tepat di tengah-tengah plot data, *eigenvector* ini menggambarkan bagaimana kedua data set memiliki hubungan sepanjang garis tersebut. Dari pencarian *covariance matrix* dan *eigenvector* ini, dapat dilakukan transformasi data menjadi hanya sebuah garis vektor.

5. *Feature Vector*

Dari *eigenvalue* yang didapat, diperoleh informasi bahwa *eigenvector* yang melintas di tengah-tengah data memiliki *eigenvalue* terbesar. *Eigenvector* ini disebut dengan nama *principal component* dari data. Di langkah ini, dilakukan pengurutan *eigenvector* berdasarkan besar *eigenvalue* yang dimiliki masing-masing *eigenvector* secara menurun. Sehingga *eigenvector* dengan *eigenvalue* terbesar berada di urutan teratas.

Singkatnya, jika dimiliki data dengan dimensi *n*, maka dapat dibentuk *covariance matrix* berdimensi *n*. Lalu dari *covariance matrix* yang diperoleh, dapat diperoleh *n* buah *eigenvector* dan *eigenvalue*. Lalu dipilih *p* buah *eigenvector* yang memiliki *eigenvalue* terbesar yang selanjutnya disebut dengan *feature vector*.

Dari *eigenvector* yang diperoleh pada tahap 4, kedua *eigenvector* dapat digunakan sebagai *feature vector* atau hanya menggunakan *eigenvector* yang memiliki *eigenvalue* terbesar sebagai *feature vector*. Dari kedua *eigenvalue* di atas, 1,2840277100 adalah *eigenvalue* terbesar. Sehingga *eigenvector* yang dipilih sebagai *feature vector* adalah

$$\begin{pmatrix} -0{,}677873399 \\ -0{,}735178656 \end{pmatrix}$$

6. Kumpulan Data Baru

Dari *feature vector* yang diperoleh, dapat ditentukan kumpulan data baru dengan cara mengalikan *vector transpose* dari *feature vector* dengan *matrix transpose* dari data normal. Di bawah ini merupakan kumpulan data hasil perkalian dengan hanya satu *eigenvector* sebagai *feature vector* sehingga menyebabkan data normal terkompresi menjadi data satu dimensi saja.

x
-0.827970186
1.77758033
-0.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

6.1.1.1.3. Tabel 2.3
Sumber [SAP-08:20]

6.1.1.1.4.2.4 ***Euclidean Distance***

Metode *euclidean distance* menyebutkan bahwa jarak antara dua titik adalah garis terpendek yang menghubungkan dua titik itu. Semakin kecil jarak antara dua titik, semakin dekat kedua titik. Dalam bidang *euclidean*, R_n, jarak antara dua titik, x dan y, didefinisikan dengan:

$$D = |x - y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

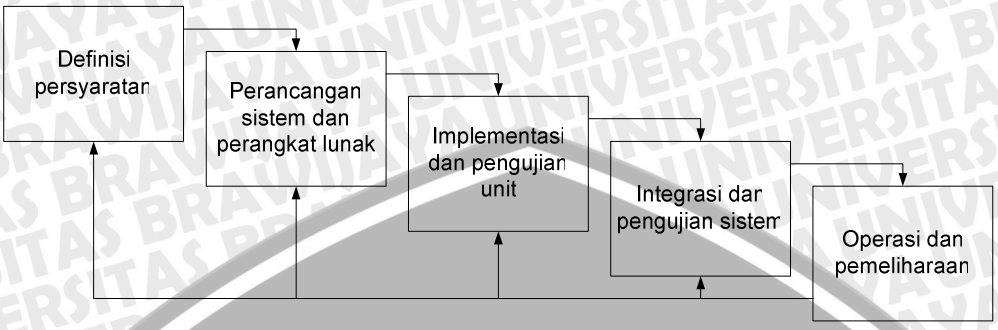
Dalam kasus pencarian citra wajah, wajah dapat direpresentasikan sebagai vektor dalam dimensi M. Untuk menemukan kesamaan nilai dari kedua citra wajah, derajat kesamaan didefinisikan sebagai jarak antara dua titik dalam vektor. Jarak disimbolkan sebagai D (p,q) Sebuah citra wajah *u* semakin mirip dengan citra wajah *v* dibandingan dengan citra wajah *w*, jika D(u, v) < D(u,w) [LIW-05].

2.5 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak adalah disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal spesifikasi sistem sampai pemeliharaan sistem setelah digunakan [SOM-03:7]. Rekayasa perangkat lunak merupakan teknologi yang meliputi proses, sekumpulan metoda dan sederetan alat bantu untuk pengembangan perangkat lunak [PRE-01:47]. Rekayasa perangkat lunak diperlukan sebagai suatu metode panduan dalam proses pengembangan suatu perangkat lunak agar menghasilkan sebuah perangkat lunak yang benar dan berkualitas.

Model proses perangkat lunak adalah representasi abstrak dari proses perangkat lunak. Setiap model proses merepresentasikan suatu proses dari sudut pandang tertentu sehingga hanya memberikan informasi parsial mengenai proses tersebut. Salah satu model

proses perangkat lunak adalah model pengembangan *waterfall.* Model ini dapat digambarkan seperti pada Gambar 2.9.



Gambar 2.4 : Model pengembangan *Waterfall*
Sumber : [SOM-03:43]

Model pengembangan *waterfall* adalah model yang menyarankan pendekatan pengembangan secara sekuen dan sistemik untuk mengembangan perangkat lunak. Model *waterfall* dimulai dari proses definisi persyaratan dilanjutkan dengan perancangan sistem dan perangkat lunak, implementasi dan pengujian unit, integrasi dan pengujian sistem, dan terakhir adalah operasi dan pemeliharaan. Setiap proses menghasilkan masukan untuk proses selanjutnya, sehingga suatu proses tidak dapat dimulai bila proses sebelumnya masih belum selesai [SOM-03:43].

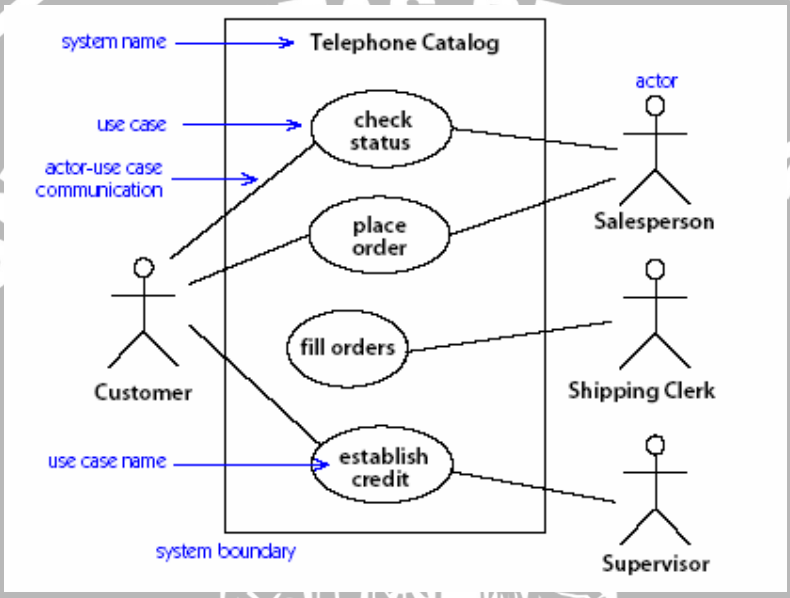
Proses definisi persyaratan atau analisis kebutuhan, perancangan, implementasi, dan pengujian memiliki dua macam, yaitu metode struktural dan metode berorientasi objek. Skripsi ini menggunakan metode analisis kebutuhan, perancangan, implementasi dan pengujian yang berorientasi objek dan menggunakan bahasa pemodelan UML (*Unified Modeling Language)*. UML adalah bahasa pemodelan standar yang terdiri dari kumpulan diagram yang terintegrasi. UML dikembangkan untuk membantu pengembang perangkat lunak dan pengembang sistem menyelesaikan tugas-tugas seperti: spesifikasi, visualisasi, desain arsitektural, konstruksi, simulasi, dan dokumentasi [CHO-03:20].

2.5.1 Analisis Kebutuhan (*Requirements Analysis*)

Analisis berorientasi objek (*Object Oriented Analysis/*OOA*)* berhubungan dengan pengembangan model berorientasi dari domain aplikasi. Objek yang teridentifikasi merefleksikan entitas dan operasi yang berhubungan dengan masalah yang akan dipecahkan [SOM-03:246]. Diagram pemodelan sistem aplikasi secara keseluruhan yang sesuai dengan analisis kebutuhan sistem digambarkan dalam diagram *use-case*. *Use case diagram* merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya.. *Use case*

merupakan fungsionalitas dari sistem yang dipersepsi oleh aktor. Aktor adalah sebuah entitas-entitas luar yang berkomunikasi dengan sistem [HAR-04:267].

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. *Use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain [DHA-03:4].



Gambar 2.5 Contoh *use-case diagram*
Sumber: [RUM-99:64]

2.5.2 Perancangan (*Design*)

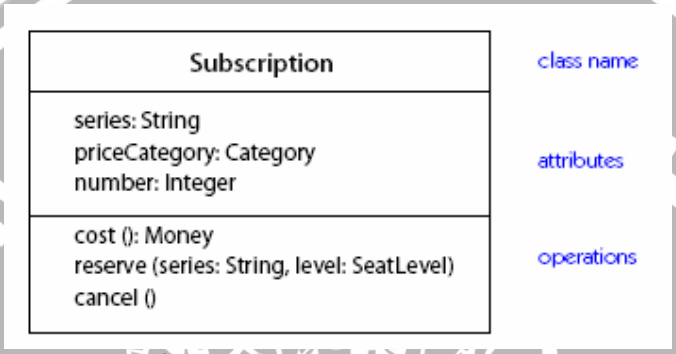
Perancangan berorientasi objek (*Object Oriented Design*/OOD) berhubungan dengan pengembangan model berorientasi objek dari sistem perangkat lunak untuk mengimplementasikan persyaratan atau analisis kebutuhan (OOA) yang teridentifikasi. Objek-objek pada OOD berhubungan dengan solusi masalah yang sedang dipecahkan [SOM-03:246]. Proses-proses umum yang digunakan untuk OOD memiliki sejumlah tahapan, yaitu memahami dan mendefinisikan konteks dan mode penggunaan sistem, merancang arsitektur sistem, mengidentifikasi objek utama sistem, mengembangkan model perancangan dan menspesifikasikan *interface* objek [SOM-03:252].

Pada tahap perancangan ini, digunakan pemodelan dengan menggunakan tiga macam diagram, yaitu *class diagram* dan *sequence diagram.*

2.5.2.1 Diagram Kelas (*Class Diagram)*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi) [DHA-03:5].

Class diagram menggambarkan struktur dan deskripsi *class, package,* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class diagram* memiliki tiga area pokok, yaitu nama (dan *stereotype*), atribut dan metoda.



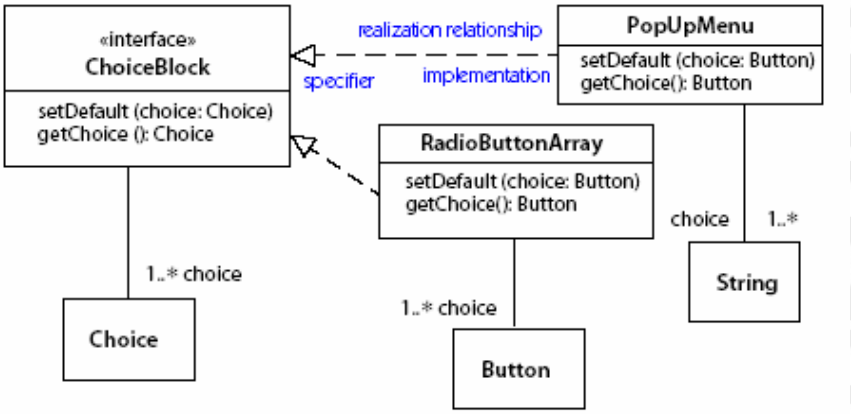
Gambar 2.6 Contoh *Class Diagram*
Sumber: [RUM-99:44]

Atribut dan metoda dapat memiliki salah satu sifat berikut [DHA-03:5]:

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja.

Hubungan antar *class* dapat dijelaskan sebagai berikut [DHA-03:6]:

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
3. Pewarisan, yaitu hubungan hierarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram.*



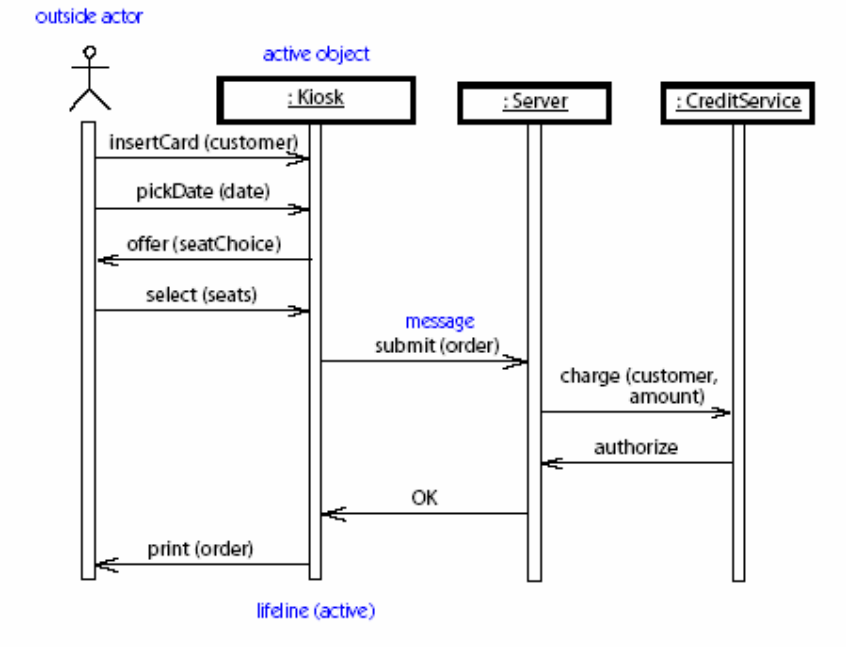
Gambar 2.7 Contoh *Class Diagram*
Sumber: [RUM-99:55]

2.5.2.2 Diagram Sequence (*Sequence Diagram)*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, *query* termasuk aktor, memiliki *lifeline*vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message* [DHA-03:8].



Gambar 2.8 Contoh *Sequence Diagram*
Sumber: [RUM-99:87]

2.5.3 Implementasi

Implementasi perangkat lunak merupakan realisasi desain perangkat lunak dengan menggunakan bahasa pemrograman berorientasi objek. Bahasa pemrograman berorientasi objek seperti Java mendukung implementasi langsung dari objek dan menyediakan fasilitas untuk mendefinisikan kelas objek [SOM-03:246].

Java adalah bahasa yang dapat dijalankan dimana pun dan di sembarang platform apapun, di beragam lingkungan: *internet, intranets, consumer electronic products*, dan *computer applications.* Keunggulan yang dimiliki bahasa pemrograman Java yaitu [HAR-03:13-17]:

- Bahasa yang sederhana

  Java dirancang agar mudah dipelajari dan digunakan secara efektif. Java tidak menyediakan fitur-fitur rumit bahasa pemrograman level tinggi. Banyak pekerjaan pemrogram yang mulanya harus dilakukan manual, sekarang digantikan dikerjakan Java secara otomatis seperti dealokasi memori.

- Bahasa berorientasi objek

  Java merupakan salah satu bahasa pemrograman yang berorientasi objek.

- Bahasa yang dikompilasi

Sebelum program dalam bahasa Java dijalankan, program dikompilasi terlebih dahulu menggunakan *Java Compiler*. Kompilasi akan menghasilkan file "*bytecode*" yang serupa fungsinya dengan file kode mesin. Program "*bytecode*" yang dihasilkan dapat dieksekusi di sembarang *Java Interpreter*.

- Bahasa yang independen terhadap *platform*

  Platform independence adalah kemampuan program bekerja di sistem operasi atau sistem komputer berbeda. Bahasa Java merupakan bahasa yang secara sempurna tidak tergantung *platform.*

- Bahasa *Multithreding*

  Java menyediakan fitur untuk menulis program *multithreaded*, program mempunyai lebih dari satu *thread* eksekusi pada saat yang sama sehingga memungkinkan program menangani beberapa tugas secara konkuren.

- Bahasa yang mampu diperluas

  Java mendukung *native method*, yaitu fungsi ditulis dalam bahasa lain, biasanya C/C++.

2.5.4 Pengujian *(Testing)*

Pengujian adalah suatu proses menjalankan sistem atau komponen berdasarkan kondisi tertentu (kasus uji), mengamati atau menulis hasilnya dan membuat evaluasi di beberapa bagian sistem atau komponen [COP-04]. Pengujian merupakan salah satu elemen penting dari penjaminan kualitas perangkat lunak dan merepresentasikan *review* akhir dari spesifikasi, perancangan dan implementasi. Pengujian dilakukan agar perangkat lunak sedapat mungkin terbebas dari kesalahan sebelum digunakan oleh *user* [PRE-01:437].

2.5.4.1 Teknik Pengujian

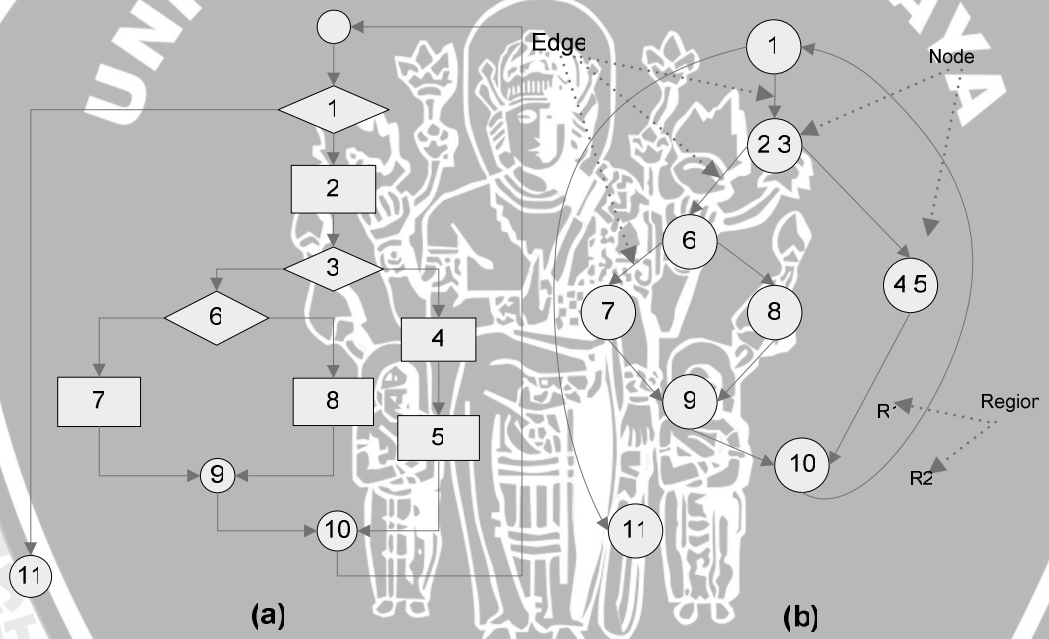
Pengujian perangkat lunak memerlukan perancangan kasus uji (*test case*) agar dapat menemukan kesalahan dalam waktu singkat dan usaha minimum. Metode perancangan kasus uji menyediakan mekanisme penting yang dapat membantu menjamin kompleksitas pengujian dan keandalan yang tinggi untuk mengatasi kesalahan [PRE-01:443]. Perangkat lunak dapat diuji melalui dua cara yaitu *white box testing* dan *black box testing.*

2.5.4.1.1 White Box Testing

White box testing adalah teknik pengujian yang menguji berdasarkan jalur internal, struktur dan implementasi dari perangkat lunak yang sedang diuji [COP-04]. *White box testing* adalah teknik pengujian yang menggunakan struktur kontrol dari prosedur yang

terdapat dalam perancangan untuk membuat kasus uji [PRE-01:444]. Ada dua jenis pengujian yang termasuk *white box testing* yaitu *basis path testing* dan *control structure testing.*

Pada skripsi ini menggunakan *basis path testing* yaitu *white box testing* yang dibuat berdasarkan tingkat kompleksitas dari algoritma hasil perancangan. Langkah-langkah *basis path testing* yaitu mendefinisikan *flow graph* berdasarkan *mapping* dari *flow chart,* menentukan ukuran kompleksitas (*cyclomatic complexity*) dan mendefinisikan kasus uji. Dari *flowchart* seperti yang ditunjukkan pada Gambar 2.14 (a) bisa dimodelkan ke dalam sebuah *flow graph* pada Gambar 2.15 (b). Jumlah pengujian (kasus uji) yang harus dieksekusi dapat ditunjukkan dengan nilai *cyclomatic complexity*. *Cyclomatic complexity* adalah angka yang menyatakan jumlah jalur independen atau jalur dasar dari sebuah program.



Gambar 2.9 Gambar transformasi dari *flow chart* (a) ke *flow graph* (b)
Sumber : [PRE-01:447]

Jalur independen (*independent path*) adalah setiap jalur dalam program yang memiliki setidaknya satu set pernyataan atau kondisi yang baru sama sekali (belum pernah digunakan oleh jalur sebelumnya) [PRE-01:446]. Untuk menentukan *cyclomatic complexity* bisa dilakukan dengan beberapa cara, di antaranya [PRE-01:448]:

1. Jumlah *region* pada *flow graph* sesuai dengan *cyclomatic complexity*.
2. *Cyclomatic complexity* V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node.*

3. V(G) = P + 1, dimana P adalah jumlah *predicate node* yaitu *node* yang merupakan kondisi (ada 2 atau lebih *edge* akan keluar *node* ini).

2.5.4.1.2 Black Box Testing

Black box testing adalah teknik pengujian yang menguji hanya berdasarkan kebutuhan dan spesifikasi [COP-04]. *Black box testing* juga disebut sebagai *behavioral testing* dan berfokus pada kebutuhan fungsi dari perangkat lunak [PRE-01:459]. Proses umum yang terjadi pada *black box testing* yaitu [COP-04]:

- Kebutuhan atau spesifikasi dianalisa terlebih dahulu.
- Penentuan input valid terpilih berdasarkan spesifikasi untuk menentukan perangkat lunak berjalan dengan benar. Input yang tidak valid juga harus dipilih untuk memverifikasi bahwa perangkat lunak dapat mendeteksinya dan menanganinya dengan baik.
- Penentuan output yang diharapkan sesuai dengan input yang telah dipilih.
- Pengujian dibuat dengan input yang telah dipilih.
- Pengujian dijalankan.
- Output yang sebenarnya dibandingkan dengan output yang diharapkan.
- Penentuan dibuat menyangkut perangkat lunak berfungsi sesuai dengan spesifikasi yang telah ditentukan.

2.5.4.2 Strategi Pengujian

Strategi untuk pengujian perangkat lunak merupakan aktifitas mengintegrasikan metode perancangan kasus uji ke dalam sebuah rangkaian langkah-langkah pengujian yang terencana sehingga menghasilkan perangkat lunak yang baik. Strategi menyediakan urutan langkah yang harus dilakukan sebagai bagian dari pengujian. Setiap langkah direncanakan kemudian dikerjakan dan ditentukan berapa banyak usaha, waktu, dan sumber daya yang dibutuhkan. Strategi untuk melakukan pengujian perangkat lunak, dimulai dari dengan "pengujian kecil" bergerak menuju ke "pengujian besar". Pengujian berorientasi objek akan memulai pengujian dari *unit testing*, bergerak menuju *integration testing* dan berakhir pada *validation testing* [PRE-01:477].

2.5.4.2.1 Unit Testing

Unit testing berfokus pada usaha verifikasi pada unit terkecil dari perancangan perangkat lunak yaitu pada komponen atau modul. *Unit testing* berorientasi *white box* dan setiap langkah dapat dikerjakan secara paralel untuk berbagai komponen. *Unit testing* pada

suatu modul dilakukan dengan menguji *interface*, struktur data lokal, kondisi batas, jalur independen dan jalur penanganan kesalahan [PRE-01:485].

2.5.4.2.2 Integration Testing

Integration testing merupakan pengujian yang dilakukan untuk menguji beberapa komponen yang saling berinteraksi. *Integration testing* berorientasi *black box* dan mempunyai dua pola pengujian yaitu *top-down integration* dan *bottom-up integration*. *Top down integration* mempunyai dua jenis pendekatan integrasi yaitu *depth-first integration* (pengujian dimulai dari jalur utama) dan *breadth-first integration* (pengujian dilakukan secara urut per *level*) [PRE-01:488]. Langkah-langkah proses integrasi pada *top down integration*, yaitu [PRE-01:489]:

- Modul kontrol utama digunakan sebagai *test driver* dan *stub* digunakan untuk menggantikan semua komponen dibawahnya.
- Pemilihan pendekatan integrasi yang diinginkan (*depth* atau *breadth first*).
- Pengujian dikerjakan untuk setiap komponen yang diintegrasikan.
- *Stub* digantikan dengan komponen yang sebenarnya setelah menyelesaikan serangkaian pengujian.
- Proses akan terus dilakukan sampai membentuk sebuah perangkat lunak yang utuh.

Bottom-up integration akan memulai integrasi dari komponen pada level terendah di struktur program perangkat lunak. Langkah-langkah proses integrasi pada *bottom –up integration*, yaitu [PRE-01:490]:

- Komponen pada level terndah digabung ke dalam sebuah sub fungsi (*cluster*).
- Driver akan dibuat untuk menguji setiap *cluster*.
- Driver akan diganti dengan modul sesungguhnya setelah sub fungsi teruji.
- Proses akan terus dilakukan sampai membentuk sebuah perangkat lunak yang utuh.

2.5.4.2.3 Validation Testing

Validation testing merupakan pengujian yang dilakukan pada perangkat lunak secara keseluruhan untuk memastikan bahwa perangkat lunak telah memenuhi seluruh kebutuhan (*requirement*) yang telah ditentukan. *Validation testing* lebih banyak menggunakan teknik *black box testing*. Pengujian dilakukan dirancang agar dapat menjamin semua kebutuhan fungsional, karakteristik perilaku dan kebutuhan performansi telah dicapai serta dokumentasi yang benar [PRE-01:495].

BAB III
METODE PENELITIAN

Pada bab ini dijelaskan mengenai langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan bagaimana aplikasi dapat dikembangkan lebih lanjut.

3.1 Studi Literatur

Tahap ini meliputi pengumpulan data dan sumber-sumber penelitian yang bertujuan untuk mendapatkan gambaran, detail dan dasar teori mengenai: *Eigenfaces, Principal Component Analysis*, serta *Euclidean Distance*. Juga mendalami materi lain yang berhubungan dengan tugas akhir

3.2 Perancangan

Perancangan didasarkan pada teori-teori yang ada sehingga dapat diaplikasikan pada sistem yang dibuat yaitu sistem *Face Recognition* yang menggunakan eigenface sebagai obyek recognition. Algoritma yang digunakan pada *face recognition* ini yaitu algoritma *PCA* dan dilanjutkan dengan *Euclidean distance*.

Untuk membangun sistem face recognition ini, perancangan yang dilakukan meliputi dua tahap. Proses analisis kebutuhan dilakukan pada tahap pertama dan tahap yang kedua adalah proses perancangan perangkat lunak.

Tahap analisis kebutuhan terdiri dari dua langkah yaitu membuat daftar kebutuhan *user* dan menggunakan pemodelan *use case diagram* untuk menggambarkan kebutuhan tersebut. Sedangkan proses perancangan perangkat lunak mempunyai dua tahap, yaitu perancangan umum dan perancangan detail. Perancangan umum menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem keseluruhan. Perancangan detail menggunakan *class diagram* dan *sequence diagram* sebagai pemodelan perangkat lunak.

Perancangan aplikasi dilakukan untuk mempermudah implementasi, analisis algoritma dan pengujian. Perancangan aplikasi berdasarkan *Object Oriented Analysis* dan *Object Oriented Design* yaitu menggunakan pemodelan UML (*Unified Modeling*

Language). Pada tahap ini dibuat suatu diagram pemodelan sistem aplikasi secara keseluruhan yang sesuai dengan analisis kebutuhan sistem yang digambarkan dalam diagram *use-case*. Perancangan dilakukan dengan membuat diagram *sequence* yang memperlihatkan interaksi pengguna dengan objek-objek dalam aplikasi dan diagram kelas untuk melihat hubungan antar kelas, objek dan *interface* yang ada di dalam aplikasi.

Perancangan berorientasi objek (*Object Oriented Design/OOD)* berhubungan dengan pengembangan model berorientasi objek dari sistem perangkat lunak untuk mengimplementasikan kebutuhan-kebutuhan yang telah teridentifikasi pada analisis kebutuhan (*Object Oriented Analysis/OOA*). [SOM-03:246]. *Use case diagram* yang telah dimodelkan pada tahap analisis kebutuhan memberikan sebuah pandangan terhadap kebutuhan fungsional dari *user* yang harus disediakan oleh sistem perangkat lunak. Proses perancangan perangkat lunak mempunyai dua tahap, yaitu perancangan umum dan perancangan detail. Perancangan umum menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem secara keseluruhan. Perancangan detail menggunakan *class diagram* dan *sequence diagram* sebagai pemodelan perangkat lunak. Perancangan detail melakukan perancangan terhadap pola hubungan antar komponen-komponen detail (dalam konteks berorientasi objek adalah klas dan objek) sehingga mampu membentuk sebuah fungsi kerja yang mampu memberikan pelayanan terhadap kebutuhan *user.*

3.3 Implementasi

Implementasi aplikasi dilakukan dengan mengacu kepada perancangan aplikasi. . Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman berorientasi objek yaitu menggunakan bahasa pemrograman Java.

3.4 Pengujian dan Analisis

Pengujian aplikasi yaitu untuk mengetahui kesesuaian analisa kebutuhan yang dibuat dengan implementasi aplikasi. Analisis sistem dilakukan dengan membandingkan sistem aplikasi yang dengan teori yang ada sehingga didapatkan suatu kesimpulan.

Pengujian perangkat lunak yang digunakan yaitu *whitebox testing* untuk menguji struktur dari perangkat lunak *dan blackbox testing* untuk menguji perilaku dari perangkat lunak ketika diberi masukan data tertentu.

Pengujian algoritma dilakukan untuk menguji tingkat akurasi proses *face recognition* menggunakan metode PCA. Pengujian akan dibagi kedalam empat buah skenario.

Skenario pertama, percobaan menggunakan citra test set satu ekspresi kemudian training set terdiri dari orang-orang yang berbeda. Masing masing orang satu ekspresi. Tujuan dari percobaan ini adalah untuk mengetahui kemampuan sistem dalam mencocokkan wajah orang yang sama antara test set dan training set.

Skenario kedua, percobaan menggunakan citra training set yang memiliki berbagai ekspresi dari beberapa orang, Citra test set yang digunakan memiliki ekspresi yang berbeda dengan training set. Tujuan percobaan ini adalah untuk mengetahui kemampuan sistem dalam mengenali wajah dalam ekspresi yang berbeda-beda

Skenario ketiga, Training set dan test set memiliki ekspresi yang sama. Citra dalam training set, terdapat citra dari beberapa orang yang berbeda. Tiap citra pada training set memiliki level brightess yang berbeda-beda. Tujuan dari skenario ini untuk mengetahui kemampuan sistem dalam mengenali wajah dalam kondisi brightness yang berbeda-beda.

Proses pengujian dilakukan dengan menggunakan image database dari Yale University.

3.5 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

BAB IV
PERANCANGAN

Pada skripsi ini dirancang suatu pengembangan aplikasi *Face Recognition* untuk *user login* dengan menggunakan metode *Principal Component Analysis*. Perancangan yang dilakukan meliputi dua tahap. Proses analisis kebutuhan dilakukan pada tahap pertama dan tahap yang kedua adalah proses perancangan perangkat lunak. Tahap analisis kebutuhan terdiri dari dua langkah yaitu membuat daftar kebutuhan *user* dan menggunakan pemodelan *use case diagram* untuk menggambarkan kebutuhan tersebut. Proses perancangan perangkat lunak mempunyai dua tahap, yaitu perancangan umum dan perancangan detail. Perancangan umum menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem keseluruhan. Perancangan detail menggunakan *class diagram* dan *sequence diagram* sebagai pemodelan perangkat lunak.

4.1 Analisis Kebutuhan

Pengembangan sebuah perangkat lunak bertujuan untuk menghasilkan perangkat lunak yang dapat memenuhi kebutuhan *user*. Setiap pengembangan sebuah sistem perangkat lunak memerlukan adanya dokumentasi terhadap kebutuhan-kebutuhan *user* agar tujuan tersebut tercapai. Tahap analisis kebutuhan dilakukan dengan memodelkan kebutuhan ke dalam *use case diagram. Use case diagram* bertujuan untuk menggambarkan kebutuhan-kebutuhan fungsional yang harus disediakan oleh sistem *face recognition* agar dapat memecahkan permasalahan yang dihadapi oleh *user.*

4.1.1 Daftar Kebutuhan

Daftar kebutuhan merupakan daftar yang menguraikan kebutuhan-kebutuhan *user* yang harus disediakan oleh perangkat lunak baik kebutuhan fungsional maupun non fungsional. Proses yang dilakukan sebelum menentukan daftar kebutuhan adalah mengidentifikasi aktor yang menggunakan sistem *face recognition*. Tabel 4.1 memperlihatkan seorang aktor beserta penjelasannya yang merupakan hasil dari proses identifikasi aktor.

Tabel 4.1 Deskripsi aktor

Aktor	Keterangan
User	aktor yang akan menggunakan sistem *Face Recognition*

Sumber: Hasil Analisis Kebutuhan

Penyusunan daftar kebutuhan fungsional maupun kebutuhan non-fungsional dari sistem Face Recognition dilakukan setelah identifikasi aktor. Daftar kebutuhan fungsional sistem disertai dengan nama *use case* yang merepresentasikan fungsionalitas dari kebutuhan tersebut. Daftar kebutuhan fungsional dan kebutuhan non-fungsional tersebut ditunjukkan pada Tabel 4.2.

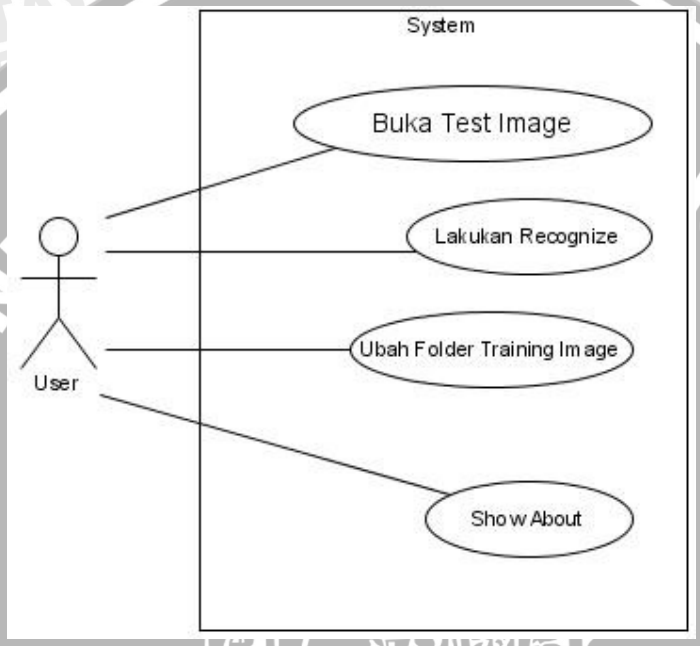
Tabel 4.2 Daftar Kebutuhan Fungsional dan Non Fungsional Sistem Face Recognition

No	Kebutuhan	Use case
1.	Sistam harus mampu melakukan proses face recognition, memilih wajah yang sesuai antara input(test image) dengan yang ada dalam fólder database image (training image)	Lakukan Recognize
2.	Sistem harus dapat merubah folder training image	Buka Training Image
3.	Sistem harus dapat memilih image yang akan di uji	Buka test Image
4.	Sistem harus dapat menunjukkan keterangan tentang sistem	Show About
5.	Algoritma yang digunakan adalah Principal Component Analysis (PCA) dan Euclidean distance	-
6.	Citra yang digunakan hádala citra wajah tampak depan yang berasal dari dataset Yale University	-
7	Nilai akurasi algoritma diperoleh dari tingkat keberhasilan sistem mengenali wajah dalam test image dengan benar	-
8	Semua komponen dari sistem *Face Recognition* dikembangkan dengan bahasa pemrograman Java	-

Sumber: Hasil Analisis Kebutuhan

4.1.2 *Use Case Diagram*

Use case diagram merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use case*, aktor dan hubungannya.. *Use case* merupakan fungsionalitas dari sistem yang diinisiasi oleh aktor. Aktor adalah sebuah entitas-entitas luar yang berkomunikasi dengan sistem [HAR-04:267]. Pemodelan dalam *use case diagram* yang menggambarkan fungsionalitas yang disediakan oleh sistem Face Recognition terhadap *user* diperlihatkan pada gambar 4.1.



Gambar 4.1 *Use Case Diagram*
Sumber: Hasil Analisis Kebutuhan

Gambar 4.1 memperlihatkan empat buah *use case* yaitu buka *test image, lakukan recognize,* ubah folder training image, dan show about *Use case-use case* tersebut yang menggambarkan kebutuhan sistem Face Recognition. Setiap *use case* dapat dijelaskan lebih lengkap dalam skenario *use case*.

Skenario *use case* menjelaskan secara detail interaksi dari masing-masing *use case* yang terdapat pada diagram *use case*. Skenario *use case* berisi uraian nama *use case*, aktor yang berhubungan dengan *use case* tersebut, tujuan dari *use case*, deskripsi global tentang *use case*, pra-kondisi yang harus dipenuhi dan pasca-kondisi yang diharapkan setelah berjalannya fungsional *use case*. Skenario *use case* juga memberikan penjelasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor (aliran utama), serta kejadian alternatif yang akan terjadi jika suatu kondisi tidak bisa terpenuhi (aliran alternatif).

Kebutuhan fungsional untuk membuka *test image* yang diinginkan *user* direpresentasikan oleh *use case buka test image*. Skenario untuk *use case buka test image* Dokumen dijelaskan pada tabel 4.3.

Tabel 4.3 Skenario untuk *use case Buka Test Image*

Nama *Use case*	Buka Test Image
Aktor	*User*
Tujuan	Membuka salah satu image yang akan diujikan ke dalam sistem dan manampilkanya
Deskripsi	*User* dapat membuka image yang akan dideteksi melalui menu yang disediakan. System akan menampilkan citra yang telah dipilih sebagai bahan validasi.
Pra-kondisi	Sistem dijalankan dan image wajah telah tersedia dalam direktori lokal.
Pasca-kondisi	Image yang diinginkan telah dipilih dan ditampilkan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memilih image yang akan didiagnosis dengan menekan menu "*Open Image*" pada halaman utama.	2. Sistem menampilkan form untuk membuka image yang diinginkan.
	3. Sistem menampilkan image yang telah dipilih pada layar.

Sumber: Hasil Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat menampilkan melakukan proses recognition direpresentasikan oleh *use case* Lakukan Recognize. Skenario untuk *use case* Lakukan Recognize dijelaskan pada Tabel 4.4.

Tabel 4.4 Skenario untuk *use case Lakukan Recognize*

Nama *Use case*	Lakukan Recognize
Aktor	*User*
Tujuan	Melakukan proses face recognition dan menampilkan image

	hasil face recognition
Deskripsi	*User* dapat membuka image yang akan direcognize melalui menu yang disediakan. System akan menampilkan citra yang telah dipilih. Kemudian user dapat menekan tombol recognize untuk menjalankan proses.
Pra-kondisi	Sistem dijalankan dan image wajah telah tersedia dalam direktori lokal.
Pasca-kondisi	Image yang berhasil direcognize ditampilkan.

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan menu recognize yang ada di halaman utama	2. Sistem menampilkan hasil recognition pada layar
Alternatif aliran 1: Eksepsi jika user belum memilih test image	
	1. Sistem menampilkan pesan "Sorry, can't recognize"
Alternatif aliran 2: Eksepsi jika user memilih test image yang tidak berekstensi GIF atau tidak memiliki dimensi yang sama dengan training image	
	1. Sistem menampilkan pesan "Sorry can't recognize"

Sumber: Hasil Analisis Kebutuhan

Kebutuhan fungsional agar sistem dapat mengganti folder training image direpresentasikan oleh *use case* Ubah Folder Training Image. Skenario untuk *use case* Ubah Training Image dijelaskan pada Tabel 4.5.

Tabel 4.5 Skenario untuk *use case* Ubah Folder Training Image

Nama *Use case*	Ubah Folder Training Image
Aktor	*User*
Tujuan	Untuk merubah lokasi folder training image

Deskripsi	User dapat merubah lokasi training image dengan menekan tombol option pada menu utama. Sistem akan menampilakan menu untuk mem-*browse* direktori harddisk
Pra-kondisi	Sistem dijalankan dan image wajah telah tersedia dalam direktori lokal.
Pasca-kondisi	Sistem mengganti folder lokasi training image

Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* memilih folder training image dengan menekan tombol option pada menu utama	2. Sistem menampilkan menu untuk mem-browse direktori.

Sumber: Hasil Analisis Kebutuhan

Kebutuhan fungsional untuk memberikan petunjuk cara penggunaan sistem kepada *user* direpresentasikan oleh *use case* Show About. Skenario untuk *use case* Bantuan dijelaskan pada tabel 4.6.

Tabel 4.6 **Skenario untuk *use case* Show about**

Nama *Use case*	Show About
Aktor	*User*
Tujuan	Memberikan petunjuk cara penggunaan sistem kepada *User*.
Deskripsi	Untuk memberikan petunjuk cara penggunaan sistem dan hal-hal yang harus dipersiapkan untuk mengoperasikan sistem kepada *User* melalui halaman bantuan.
Pra-kondisi	Sistem sudah dijalankan .
Pasca-kondisi	Tampil halaman yang berisi petunjuk dan hal-hal yang harus

	dipersiapkan untuk mengoperasikan sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari Sistem**
1. *User* menekan tombol "*About*" pada halaman utama untuk mendapatkan petunjuk cara pengoperasian sistem.	2. Sistem menampilkan halaman berisi informasi cara pengoperasian sistem dan hal-hal yang harus dipersiapkan untuk memulai pengoperasiannya.

Sumber: Hasil Analisis Kebutuhan

4.2 Perancangan Perangkat Lunak

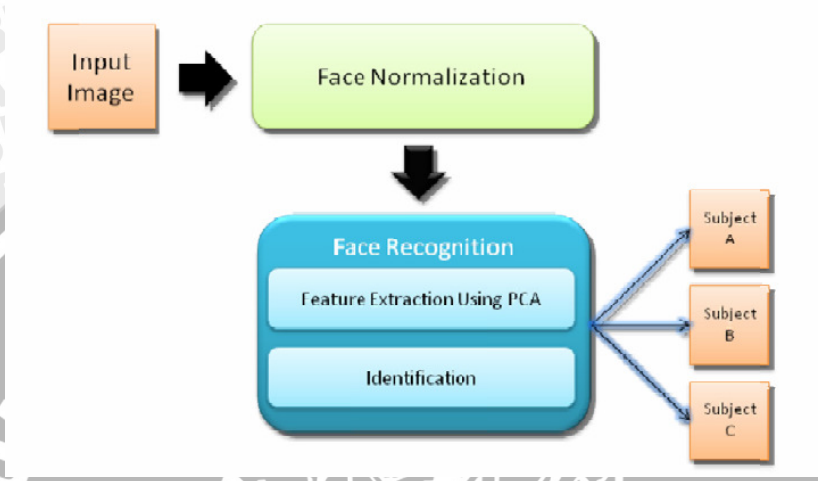
Perancangan berorientasi objek (*Object Oriented Design/OOD)* berhubungan dengan pengembangan model berorientasi objek dari sistem perangkat lunak untuk mengimplementasikan kebutuhan-kebutuhan yang telah teridentifikasi pada analisis kebutuhan (*Object Oriented Analysis/OOA*). [SOM-03:246]. *Use case diagram* yang telah dimodelkan pada tahap analisis kebutuhan memberikan sebuah pandangan terhadap kebutuhan fungsional dari *user* yang harus disediakan oleh sistem perangkat lunak. Proses perancangan perangkat lunak mempunyai dua tahap, yaitu perancangan umum dan perancangan detail. Perancangan umum menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem secara keseluruhan. Perancangan detail menggunakan *class diagram* dan *sequence diagram* sebagai pemodelan perangkat lunak. Perancangan detail melakukan perancangan terhadap pola hubungan antar komponen-komponen detail (dalam konteks berorientasi objek adalah klas dan objek) sehingga mampu membentuk sebuah fungsi kerja yang mampu memberikan pelayanan terhadap kebutuhan *user.*

4.2.1 Perancangan Umum

Perancangan Umum terdiri dari perancangan umum sistem *Face Recognition* dan 3nd *Party Class/Package.* Perancangan Perancangan umum Sistem Information Retrieval menggambarkan relasi antar paket (*package)* dan klas (*class)* sebagai pemodelan sistem secara keseluruhan. 3nd *Party Class/Package* menjelaskan mengenai klas dan paket tambahan yang digunakan dalam sistem *Face Recognition.*

4.2.1.1 Perancangan Umum Sistem *Face Recognition*

Perancangan Perancangan umum Sistem *Face Recognition* menggambarkan secara global tentang elemen-elemen pembentuk sistem *Face Recognition*. Gambaran umum sistem ditunjukkan pada diagram blok pada gambar 4.2

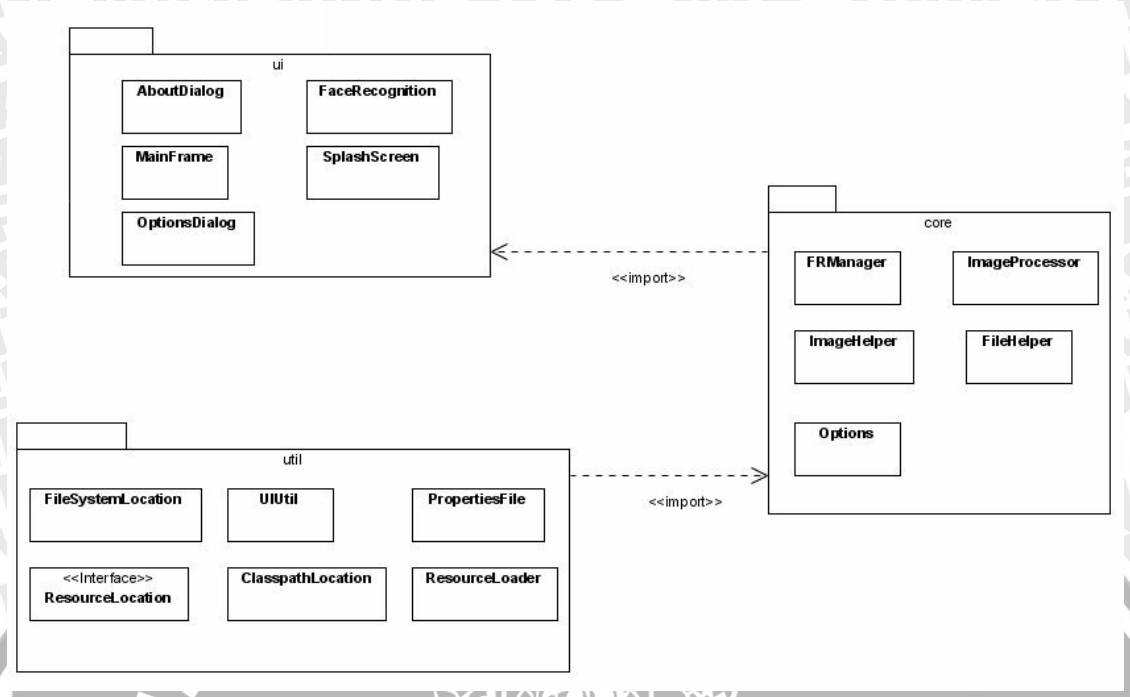


Gambar 4.2 Diagram sistem face recognition

Sumber: Perancangan

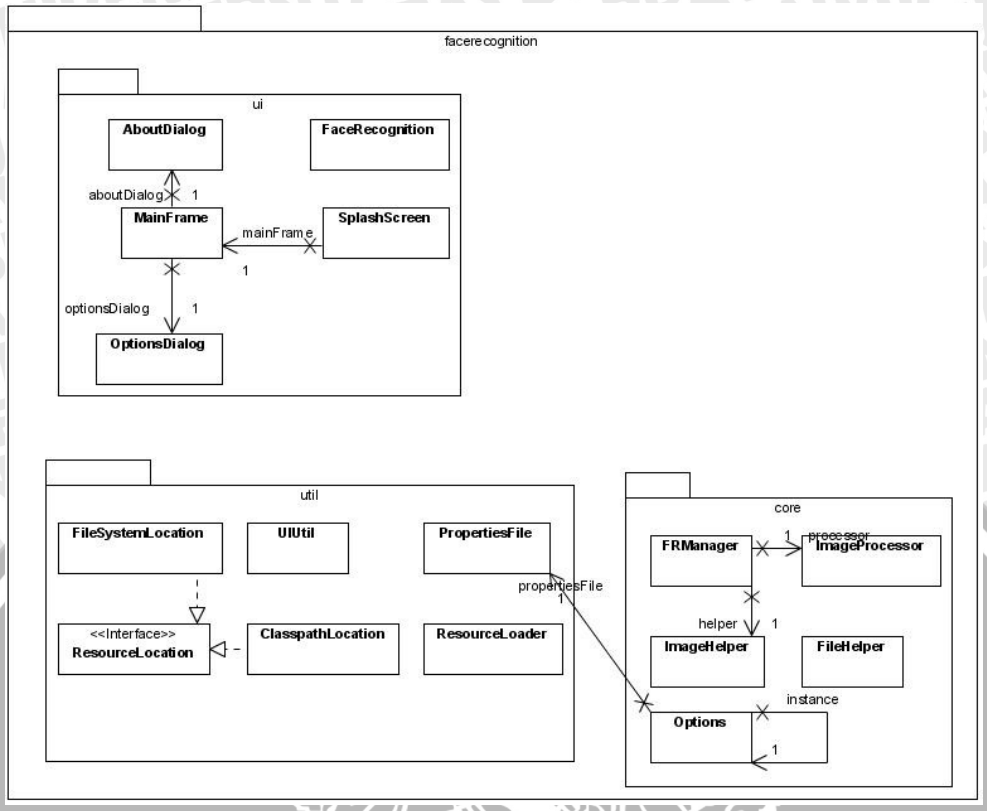
Diagram pada gambar 4.2 menunjukkan proses-proses yang terjadi dalam sistem. Input image akan terlebih dulu dinormalisasi. Kemudian akan di ekstraksi menggunakan PCA. Pada proses PCA, matrik image akan menjadi lebih sederhana. Matrik yang telah disederhanakan tersebut akan masuk pada proses identifikasi. Proses identifikasi adalah proses penghitungan jarak terdekat menggunakan metode euclidean distance. Image yang memiliki jarak terdekat dianggap sebagai image yang berasal dari orang yang sama.

Sistem *Face Recognition* mempunyai 16 buah klas yang telah dikelompokkan menjadi 3 buah paket yaitu paket ui, core dan util. Relasi antar paket yang digunakan untuk Sistem Face Recognition dapat dilihat pada gambar 4.3



Gambar 4.3 *Package Diagram*
Sumber: Perancangan

Relasi antar klas yang digunakan untuk Sistem Face Recognition ini dapat dilihat pada gambar 4.4.



Gambar 4.4 Relasi antar klas
Sumber: Perancangan

Pada package ui, kelas yang ada di dalamnya berfungsi untuk mengatur kerja GUI. Klas AboutDialog berfungsi menmunculkan menu dialog. Klas FaceRecognition berfungsi untuk mengatur image dalam GUI, Klas MainFrame berfungsi untuk menampilkan image input dan outpuk pada GUI. Klas OptionDialog berfungsi untuk menampolkan menu option. Klas ImageProcessor dan klas ImageHelper merupakan kelas utama dilakukanya pengolahan image. Klas FileHelper berfungsi untuk mengatur pemanggilan file image. Klas ImageProcessor menjadi tempat dilakukannya proses pengolahan matrix image. Klas FRManager menjadi tempat dilakukannya proses recognize.

4.2.1.2 3nd *PartyClass / Package*

3nd *Party Class/Package* menjelaskan mengenai klas dan paket tambahan yang digunakan dalam sistem *Face Recognition.* Sistem ini menggunakan paket `cern` yang berupa file `cern.jar` dan dapat di download di http://www.info.cern.ch/asd/lec++/lhep/manual/RefGuide/Units.html. Paket ini digunakan untuk melakukan proses pengolahan matrik image.

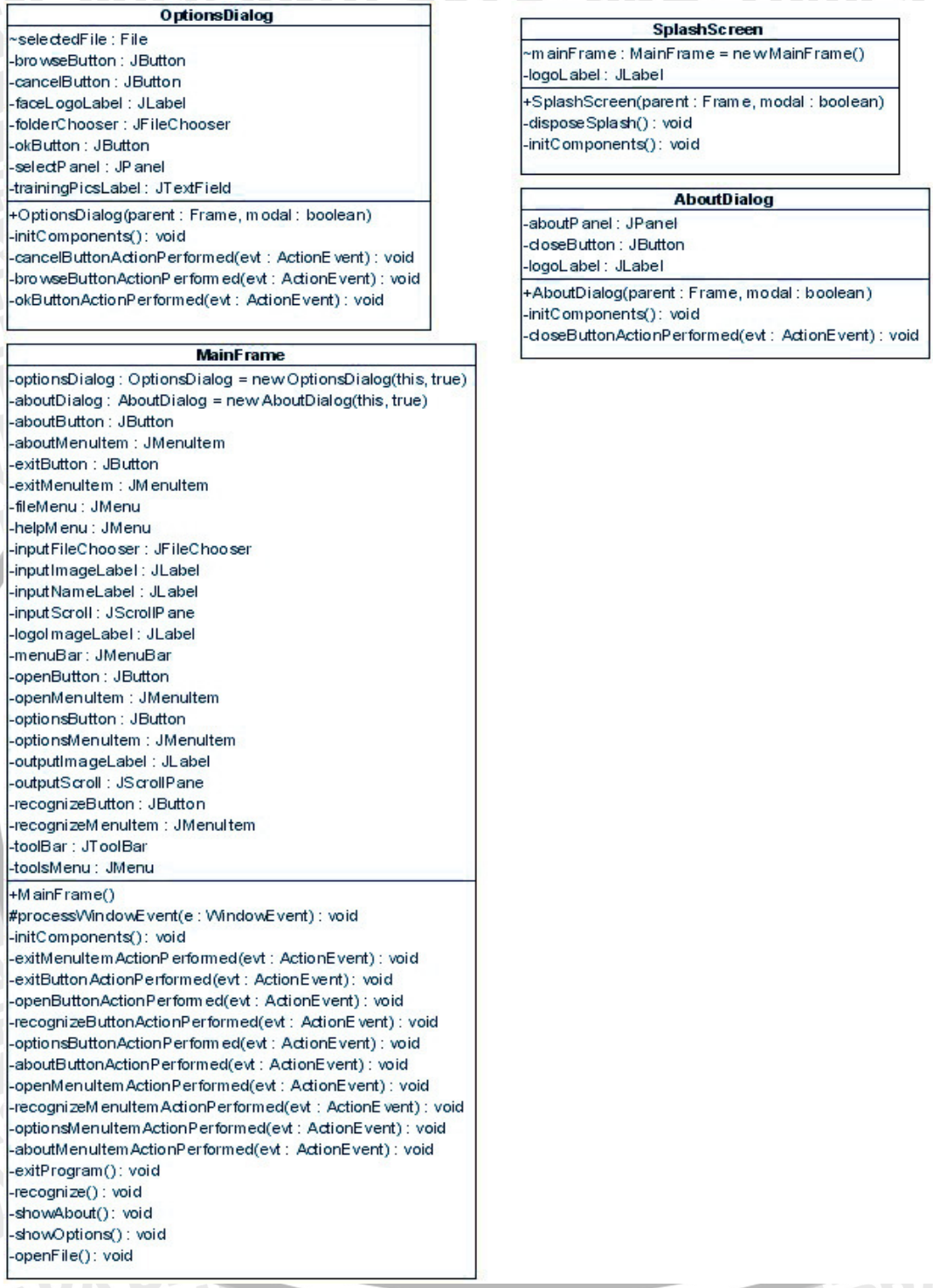
4.2.2 Perancangan Detail

Perancangan detail menggunakan diagram klas (*class diagram)* dan diagram sekuensial *(sequence diagram)* sebagai pemodelan perangkat lunak. Perancangan detail menjelaskan mengenai pola hubungan antar komponen-komponen detail (klas dan objek) sehingga mampu membentuk sebuah fungsi yang mampu memberikan pelayanan terhadap kebutuhan *user.*

4.2.2.1 Diagram Klas *(Class Diagram)*

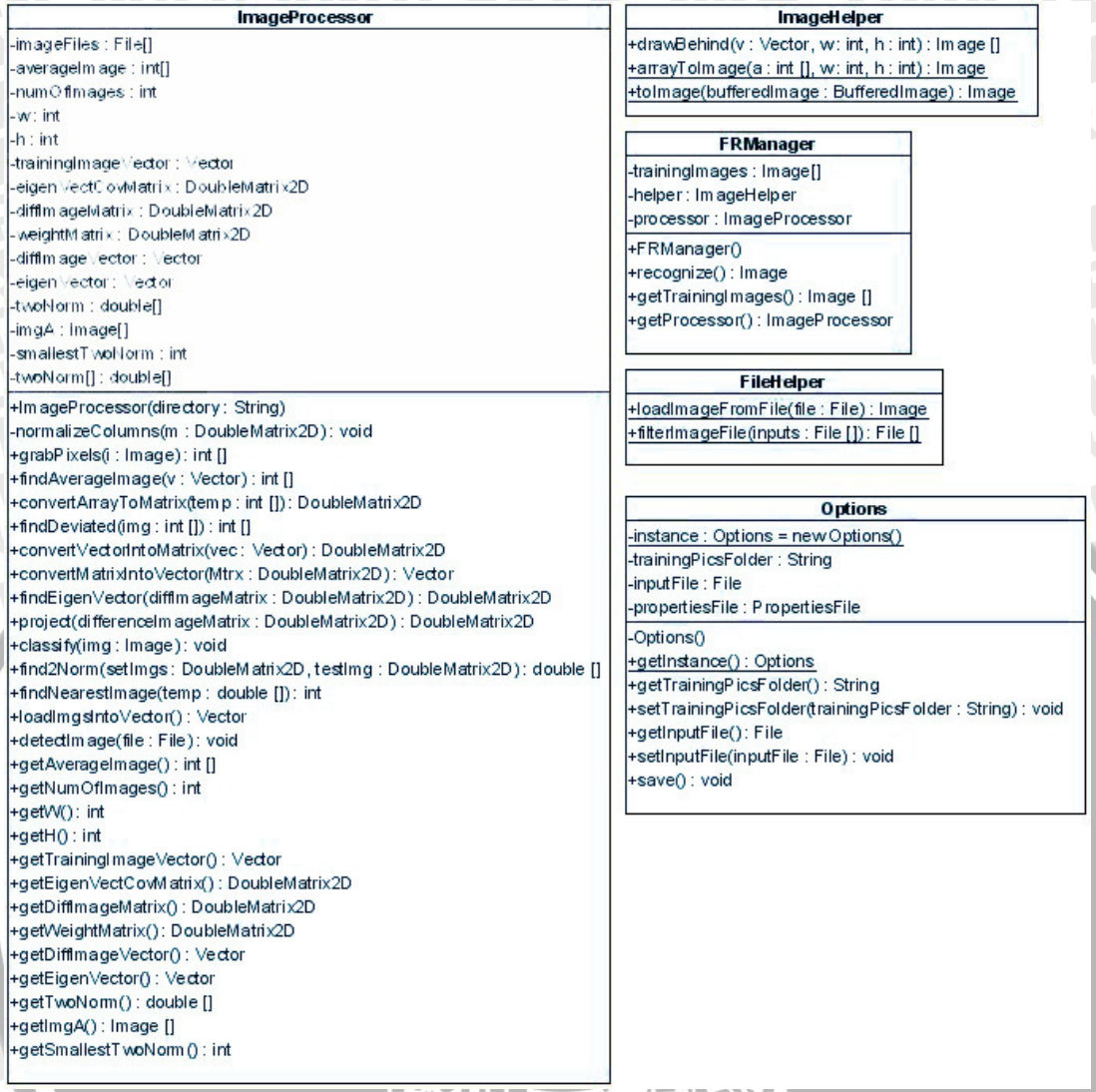
Class diagram menggambarkan struktur dan deskripsi klas, paket dan objek beserta hubungan satu sama lain seperti pewarisan, asosiasi, dan lain-lain. Klas adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Klas menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi) [DHA-03:5].

Sistem Face Recognition mempunyai klas-klas yang telah dikelompokkan ke dalam berbagai paket diantara lain paket ui, util dan core. Paket ui terdiri dari 4 buah klas yaitu klas OptionDialog, MainFrame, Splash Screen dan About Dialog. Atribut dan operasi dari masing-masing klas digambarkan dalam diagram klas pada Gambar 4.5.



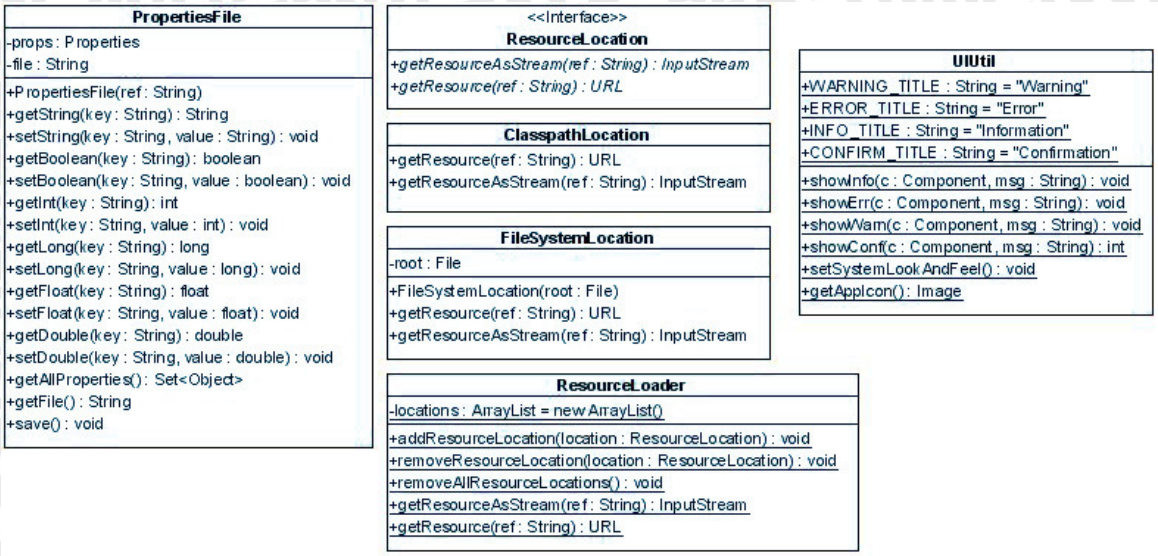
Gambar 4.5 Diagram klas anggota paket ui
Sumber: Perancangan

Paket core terdiri dari 5 buah klas yaitu klas ImageProcessor, ImageHelper, FRManager, FileHelper dan Options. Atribut dan operasi dari masing-masing klas digambarkan dalam diagram klas pada Gambar 4.6.



Gambar 4.6 Diagram klas anggota paket core
Sumber: Perancangan

Paket util terdiri dari 6 buah klas yaitu klas PropertiesFile, ResourceLocation, ClasspathLocation, FileSystemLocation, ResourceLoader dan UIUtil Atribut dan operasi digambarkan dalam diagram klas pada Gambar 4.7.



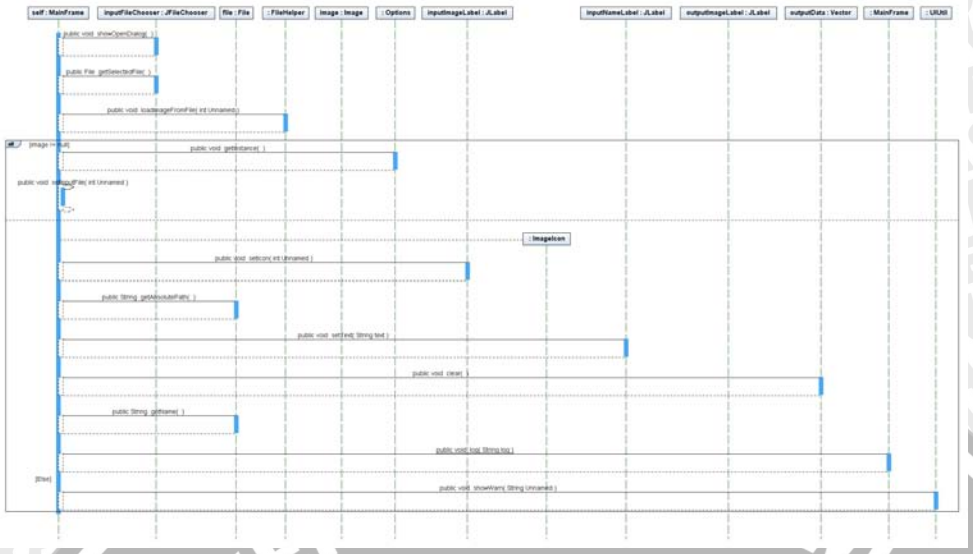
Gambar 4.7 Diagram klas anggota paket util
Sumber: Perancangan

4.2.2.2 Diagram Sekuensial *(Sequence Diagram)*

Pemodelan aliran jalannya proses interaksi antar objek atau klas yang disusun berdasarkan urutan waktu ditunjukkan oleh *sequence diagram. Sequence diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Diagram sekuensial (*sequence diagram)* disusun dengan mengambil acuan pada *use case* dan klas-klas yang membentuk fungsionalitas yang digambarkan pada *use case* tersebut.

- **Diagram sekuensial untuk *use case* Buka Test Image**

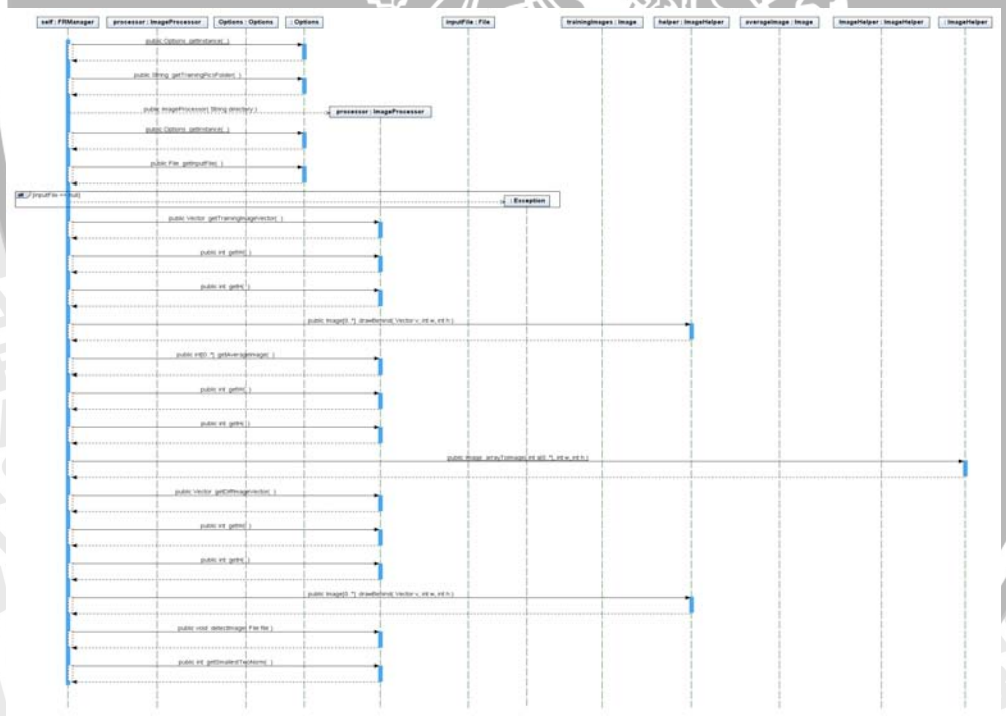
Gambar 4.8 merupakan diagram sekuensial yang menunjukkan proses memilih test image. Test image akan menjadi input proses *face recognition*.



Gambar 4.8 Diagram sekuensial Buka Test Image
Sumber: Perancangan

- **Diagram sekuensial untuk *use case* Lakukan Recognize**

Gambar 4.9 merupakan diagram sekuensial untuk proses saat melakukan proses *face recognition*.



Gambar 4.9 Diagram sekuensial LakukanRecognize
Sumber: Perancangan

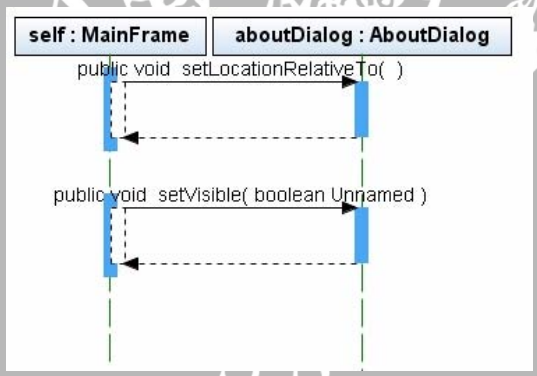
- **Diagram sekuensial untuk *use case* Ubah Folder Training Image**

Gambar 4.10 merupakan diagram sekuensial untuk proses merubah folder *training image*. Folder *training image* adalah database *image* dari *face recognition*

Gambar 4.10 Diagram sekuensial Ubah Folder Training Image
Sumber: Perancangan

- **Diagram sekuensial untuk *use case* Show About**

Gambar 4.11 merupakan diagram sekuensial proses mambuka menu About.



Gambar 4.11 Diagram sekuensial Show About
Sumber: Perancangan

BAB V
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak.dibuat. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi tiap klas pada file program, implementasi algoritma, implementasi antarmuka aplikasi dan beberapa kendala dalam implementasi.

5.1 Spesifikasi Sistem

Hasil analisis kebutuhan dan perancangan perangkat lunak yang telah diuraikan pada Bab 4 menjadi acuan untuk melakukan implementasi menjadi sebuah sistem *Face Recognition* yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak

5.1.1 Spesifikasi Perangkat Keras

Pengembangan sistem *Face Recognition* menggunakan sebuah komputer dengan spesifikasi minimum perangkat keras yang dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi perangkat keras komputer

Nama Komponen	**Spesifikasi**
Prosesor	IntelPentium(R) 4 dual Core CPU 1,6 Ghz
Memori (RAM)	2 GB
Hardisk	Maxtor 6L080L0, kapasitas 80 GB, 7200 rpm
MainBoard	Jetway I31GM4

Sumber: Implementasi

5.1.2 Spesifikasi Perangkat Lunak

Pengembangan sistem *Face Recognition* menggunakan perangkat lunak dengan spesifikasi yang dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi perangkat lunak komputer

Sistem operasi	Microsoft Windows Professional version 2002
Bahasa pemrograman	Java
Tools pemrograman	JDK 1.5.0_02
Lingkungan pemrograman	Java(TM) 2 Runtime Environment, Standart Edition (build 1.5.0_02-b09)
IDE *(Integrated Development Environment)*	MyEclipse Enterprise Workbench 5.1.0 GA

Sumber: Implementasi

5.2 Batasan-Batasan Implementasi

Beberapa batasan dalam mengimplementasikan sistem adalah sebagai berkut:

1. Data citra wajah yang digunakan adalah wajah yang terdapat dalam dataset Yale University yang didapatkan dari alamat http://cvc.yale.edu/projects/yalefaces/yalefaces.html
2. Tipe data citra yang digunakan adalah GIF grayscale.
3. Skripsi ini lebih memfokuskan pada proses mengubah citra wajah menjadi data matrik berdimensi rendah menggunakan metode Principal Component Analysis (PCA) dan mencari kesamaan antar image menggunakan metode Euclidean Distance.

5.3 Implementasi Klas pada File Program

Setiap klas yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program *.java. Tabel 5.3 menjelaskan mengenai pasangan antara klas dengan *file* program yang digunakan untuk mengimplentasikannya.

Tabel 5.3 Implementasi klas pada kode program *.java

No.	Paket	Nama Klas	Nama *File* Program
1.	ui	MainFrame	MainFrame.java
2.		AboutDialog	AboutDialog.java
3.		OptionDialog	OptionDialog.java
4.		FaceRecognition	FaceRecognition.java
5.		SplashScreen	SplashScreen.java
6.	Core	FRManager	FRManager.java
7.		ImageProcessor	ImageProcessor.java
8.		ImageHelper	ImageHelper.java
9.		FileHelper	FileHelper.java
10.		Options	Options.java
11.	util	FileSystemLocation	FileSystemLocation.java
12.		UIUtil	UIUtil.java
13.		ResourceLocation	ResourceLocation.java
14.		ClasspathLocation	ClasspathLocation.java
15.		PropertiesFile	PropertiesFile.java
16.		ResourceLiader	ResourceLoader.java

Sumber: Implementasi

5.4 Implementasi Algoritma

Langkah-langkah umum dari proses face recognition ini adalah mengekstraksi informasi (Principal Component)dari sebuah image wajah dan mengatur informasi itu dalam struktur data yang sesuai kemudian membandingkannya dengan sekumpulan image yang telah diatur pula

1. Mencari eigen vector dan eigen values dari sebuah matrik kovarian image.
2. memilih eigen vector yang memiliki nilai tertinggi (principal) untuk digunakan dalam proses komparasi.
3. Melakukan komparasi image berdasar nilai nilai eigen tertinggi.
4. Menentukan image yang memiliki selisih nilai terkecil, kemudian dianggap sebagai image yang sama.

Langkah-langkah detail

1. Ambil sekumpulan image wajah. Semua imege harus memiliki ukuran dimensi sama. Image disimpan dalam sebuah array [Γ] berdimensi n*n, yang bisa kita anggap sebagai sebuah vektor image

   $\{\Gamma_i \mid i = 1,...,M\}$

   Γ = array image

   M = jumlah image

   i = image ke

   proses ini menggunakan method *loadImgsIntoVector()*

2. Mencari image rata-rata dari sekumpulan image diatas

   $$\psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \quad \text{..... (1)}$$

   Ψ = image rata-rata

   Proses ini menggunakan method *findAverageImage(Vector v)*

3. Mancari Deviasi image

deviated [avg - img1 , avg - img2, ... , avg - img.n] images

$$\Phi_i = \Gamma_i - \psi; i = 1,..., M \quad \text{.....(2)}$$

I = image ke

Φ_i = Deviasi image

Proses ini menggunakan method *findDeviated(int[] img)*

4. Menghitung matrik kovarian

$$C = AA^T \text{ .....(3)}$$

$$C = \begin{bmatrix} c(1,1) & c(1,2) & \cdots & c(1,d) \\ c(2,1) & c(2,2) & \cdots & c(2,d) \\ \vdots & \vdots & & \vdots \\ c(d,1) & c(d,2) & \cdots & c(d,d) \end{bmatrix}$$

dimana

$$A = [\Phi_1,..., \Phi_M]$$

C = Kovarian matrik

c = Elemen matrik

A = Matrik deviasi image

A^T = Transpose matrik A

Φ_1 = Deviasi image

Untuk mencari kovarian, dilakukan proses 4.a sampai 4.c dibawah ini

4.a Untuk melakukan perhitungan diatas, kita terlebih dahulu mencari matrik L

$$L = A^T A \text{ .....(4)}$$

Kemudian menemukan eigen vector [v] yang berhubungan

$V_l(l=1,...,M)$

l = elemen matrik L

Proses ini dilakukan menggunakan method *findEigenVector(DoubleMatrix2D diffImageMatrix)*

4.b Kemudian mencari eigenvector dari matrik kovarian C

$$U = [u_1,..., u_M] = [\Phi_1,..., \Phi_M][v_1,..., v_M] = A.V \text{ .....(5)}$$

Dimana $u_l (l = 1,..., M)$ adalah eigen vector dari C

proses ini dilakukan dengan method *normalizeColumns(DoubleMatrix2D m)*

Dengan menggunakan eigen vector diatas, kita dapat menkonstruksi eigenfaces. Tetapi dalam metode PCA, hanya akan menggunakan eigen vector dengan nilai eigen yang tinggi. Eigen vector dengan nilai yang lebih rendah dari nilai threshold dapat diabaikan. Sehingga kita hanya menyimpan image yang berdasar nilai eigen tertinggi. Kumpulan image ini disebut face space. Untuk melakukan itu menggunakan java, kita menggunakan package *colt algebra*.

Berikut adalah langkah dalam implementasi

Dari persamaan 4, kita mencari

$L = A^T A$ .....(4)

Mengubahnya menjadi sebuah *DoubleDenseMatrix2D* menggunakan kelas *matric.colt*

4.c mencari eigen vector yang berkaitan

Untuk mencari eigen vector, Langkah ini menggunakan kelas *cern.colt.matrix.linalg.EigenvalueDecomposition.* Melalui proses ini akan menjadi sebuah M * M matrix.

M= jumlah training image

Dengan mengkalikan nya dengan 'A' [matrik deviasi image] kita akan mendapatkan nilai matrix eigenvector [U] dari kovarian dari 'A'. Ini akan menjadi M * X [dimana X adalah jumlah total piksel dalam sebuah image]

Proses ini dilakukan dalam method *project(DoubleMatrix2D differenceImageMatrix)*

5. Mengklasifikasi image

Dilakukan dalam kelas classify(Image img)**.** Proses diatas telah mampu menghasilkan citra eigenfaces. Eigenface ini akan digunakan sebagai dasar pengenalan wajah. Disini terdapat sebuah test image yang memiliki dimensi sama dengan training image. Yang pertamakali dilakukan terhadap test image adalah mengubah image menjadi matrik [Γ] sehingga semua pixel test image disimpan dalam matrik berukuran berukuran n(rows)*1(column). Proses ini dilakukan menggunakan method *detectImage(File file)* untuk

menentukan apakah benar yang dipanggil adalah file image atau bukan, dan method *loadImageFromFile(File file)* untuk merubah image menjadi matrik.

6. Melakukan pembobotan terhadap setiap training image. Operasi ini dapat dilakukan dengan rumus

   Matrik bobot = Transpose (Matrik kovarian eigenvektor) * Matrik deviasi image

   $W = C^T * A$ …….…… (6)

   Matrik ini akan berukuran N xN dimana N adalah total jumlah face images. Tiap masukan dalam kolom akan merepresentasikan bobot yang bersesuaian dari image tertentu dangan memperhitungkan eigenvector tertentu. Proses ini dilakukan menggunakan method *project(DoubleMatrix2D differenceImageMatrix)*

7. Proyeksikan Γi dalam image space dengan operasi sederhana, tetapi disini kita memproyeksikan image tunggal dan karenanya kita akan mendapatkan matrik berukuran N[rows] dan 1[column]. Kita sebut matrik ini sebagai matrik Test projection

   $\Omega_k = U^T(\Gamma_k - \Psi); k = 1,..., N_c$ …… (7)

   Ω_k = Matrik proyeksi

   N = Jumlah training image

   K = 1,2,….,N

   Operasi ini dilakukan menggunakan method

   project(DoubleMatrix2D differenceImageMatrix)

8. Mencari jarak antara tiap elemen dari matrik testProjection dan eleman yang bersesuaian

   Dari matrik bobot(Weight). Kita bisa memperoleh matrik baru dengan N [rows] dan N [columns]Operasi ini dilakukan menggunakan method *project(DoubleMatrix2D differenceImageMatrix)*

9. Cari 2-Norm untuk matrik yang diturunkan diatas. Ini akan menjadi sebuah matrix dari 1 [rows] * N[column]. Cari nilai minimum untuk semua nilai column. Kolom dengan nilai terkecil mewakili image yang terdekat, dan dianggap sebagai image yg sama.

   Operasi ini menggunakan method :

find2Norm(DoubleMatrix2D setImgs, DoubleMatrix2D testImg),
norm2(DoubleMatrix2D A) dan
findNearestImage(double[] temp)

Sistem *Face Recognition* ini mempunyai beberapa proses utama yaitu mengolah image, mencari jarak terpendek antar matrik image dan mengganti folder training image.

5.4.1 Implementasi Algoritma Mengolah Image

Gambar 5.1 merupakan Implementasi proses mengenali image

```
METHOD recognize THROWS Exception IS Image

DECLARATION:
TYPE processor IS ImageProcessor
inputFile IS File
trainingImages IS array of Image

DESCRIPTION:
1   processor <- CREATE OBJECT ImageProcessor
2   CALL Options.getInstance().getTrainingPicsFolder())
3   inputFile <- CALL Options.getInstance().getInputFile()
4   IF inputFile == null THEN
5       throw CREATE OBJECT Exception()
6   END IF
7   trainingImages <- CALL helper.drawBehind
8   CALL processor.getTrainingImageVector(),
9   CALL processor.getW(), CALL processor.getH())
10  CALL processor.detectImage(inputFile)
11  RETURN CALL trainingImages[CALL
12  processor.getSmallestTwoNorm()]
13  END recognize
```

Gambar 5.1 Pseudocode proses pengolahan image pada metode recognize()
Sumber: Implementasi

Penjelasan pseudocode method recognize() pada Gambar 5.1 yaitu:

1. Baris 1 Membuat objek ImageProcessor.
2. Baris 2-3 Memanggil method untuk melakukan input file (membuka folder training image)
3. Baris 4-5 melakukan eksepsi jika file input bukan file image.
4. Baris 7 menampilkan training image yang dipilih
5. Baris 8-10 melakukan proses pengenalan dengan memanggil beberapa method yang dibutuhkan
6. Baris 11-12 Menentukan image hasil recognition

5.4.2 Implementasi Algoritma Mencari jarak terdekat dari 2 matrik

```
METHOD classify PARAMTER img THROWS Exception IS void
DECLARATION:
Img IS Image
twoNorm IS double
DESCRIPTION:
1   CALL DoubleMatrix2D testProjectionMatrix
2   twoNorm <- CREATE OBJECT numOfImages
3   testProjectionMatrix <- CALL
4   project(convertArrayToMatrix(findDeviated(grabPixels(img))))
5   twoNorm <- CALL find2Norm (weightMatrix, testProjectionMatrix)
6   smallestTwoNorm <- CALL findNearestImage(twoNorm)
7   OUT "SMALLEST TWO NORM= " smallestTwoNorm
8   END classify
```

Gambar 5.2 Implementasi Algoritma Mencari jarak terpendek
Sumber: Implementasi

Penjelasan algoritma mencari jarak terpendek 2 matrik untuk `classify()` pada Gambar 5.2 yaitu

1. Baris 1 menjelaskan instansiasi `testProjectionMatrix` dari `DoubleMatrix2D`.
2. Baris 2 membuat obyek numOfImages untuk mengidentifikasi jumlah training image
3. Baris 3 untuk memanggil beberapa method untuk merubah array menjadi matrik
4. Baris 4 untuk memanggil method yang akan mencari jarak semua matrik
5. Baris 5 untuk memanggil method yang mencari jarak terpendek antar matrik.
6. Baris 6 mengeluarkan jarak terpendek yang ditemukan

5.4.3 Implementasi Mengubah Training Folder

```
METHOD browseButtonActionPerformed PARAMETER evt IS void

DECLARATION:
TYPE

DESCRIPTION:
1   CALL folderChooser.showOpenDialog(CALL this)
2   selectedFile <- CALL: folderChooser.getSelectedFile()
3     IF selectedFile != null THEN
4     CALL trainingPicsLabel.setText(CALL
5   selectedFile.detAbsolutePath())
6     END IF
7   END browseButtonActionPerformed
```

Gambar 5.3 Implementasi merubah folder training set
Sumber: Implementasi

Penjelasan pseudocode method browseButtonActionPerformed dari gambar 5.3

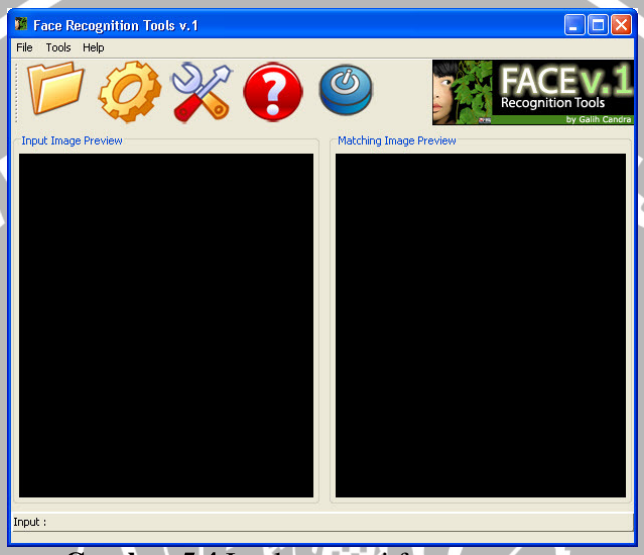
1. Baris 1 menjelaskan pemanggilan method folderChooser dan showOpenDialog untuk membuka eksplorer
2. Baris 2 mengidentifikasi file yang telah dipilih
3. Baris 3 Percabangan, melakukan perintah baris berikutnya jika kondisi benar, yaitu jika telah memilih file
4. Baris 4 mengidentifikasi path lokasi file
5. Baris 5 akhir percabangan

5.5 Implementasi Antarmuka Aplikasi Sistem Pendeteksi *Face Recognition*

Aplikasi Sistem *Face Recognition* mempunyai 2 buah aplikasi antarmuka yaitu Form Utama, dan Subform About. Semua aktivitas terjadi pada form Utama mulai dari proses memilih *test image*, melakukan *recognition* dan untuk mengubah folder *training image*. Untuk memberikan bantuan petunjuk cara pengoperasian aplikasi maka disediakan Subform About

5.5.1 Implementasi Antarmuka Form Utama

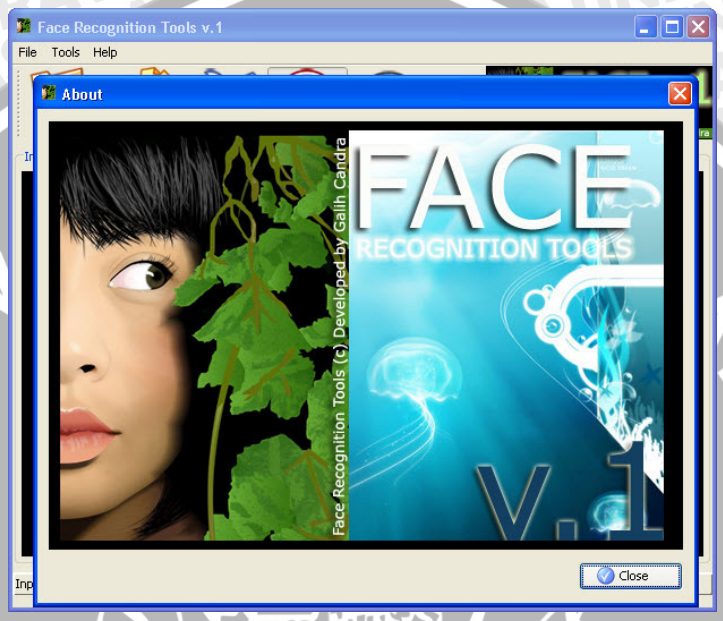
Aplikasi Sistem Face Recognition mempunyai 5 buah aplikasi utama yaitu Open image untuk input test image, Recognize untuk melakukan proses recognition, Options untuk menentukan folder training picture, About untuk menampilkan keterangan tentang aplikasi dan Exit untuk keluar dari aplikasi.



Gambar 5.4 Implementasi form utama
Sumber: Implementasi

5.5.2 Implementasi Antarmuka SubForm Help dan SubForm About

Halaman bantuan dan informasi seputar aplikasi dapat diakses pada menu About Informasi yang disediakan pada subForm Help adalah petunjuk cara mengoperasikan aplikasi yang disertai dengan gambar untuk memudahkan pengguna. SubForm About menyediakan informasi berupa pembuat aplikasi (*Author*), versi aplikasi (*Version*) dan software pemrograman yang digunakan untuk mengembangkan aplikasi (*Powered by*).



Gambar 5.5 Implementasi Form About
Sumber: Implementasi

BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis terhadap sistem *Fece Recogniion* yang telah dibangun. Proses pengujian dilakukan melalui tiga tahapan (strategi) yaitu pengujian unit, pengujian integrasi dan pengujian validasi. Pada pengujian unit dan pengujian integrasi, akan digunakan teknik pengujian *White Box (White Box Testing).* Pada pengujian validasi akan digunakan teknik pengujian *Black Box (Black Box Testing).* Proses analisis dilakukan dengan membandingkan sistem aplikasi yang dengan teori yang ada sehingga didapatkan suatu kesimpulan .

Pengujian

Proses pengujian dilakukan melalui tiga tahapan (strategi) yaitu pengujian unit, pengujian integrasi dan pengujian validasi.

Pengujian Unit

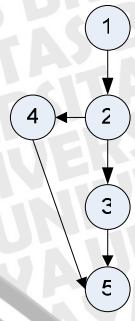
Sistem yang dibangun dengan paradigma pemrograman berorientasi objek menerapkan pengujian unit untuk suatu metode (operasi) dari suatu klas. Pada pengujian unit sistem *Face Recognition* ini, digunakan teknik pengujian *White Box (White Box Testing)* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph,* menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity),* menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan. Penulisan laporan skripsi ini hanya dicantumkan hasil pengujian unit untuk algoritma dari beberapa metode (operasi) saja (tidak untuk keseluruhan metode).

a. Pengujian Unit untuk Operasi `recognize()`

Gambar 6.1 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `recognize()`.

METHOD recognize THROWS Exception IS Image **DECLARATION:** TYPE processor IS ImageProcessor inputFile IS File trainingImages IS array of Image	**Node (N) = 5** **Edge (E) = 5**

```
DESCRIPTION:
(1) processor <- CREATE OBJECT ImageProcessor
    CALL Options.getInstance().getTrainingPicsFolder())
(2) inputFile <- CALL Options.getInstance().getInputFile()
    IF inputFile == null THEN
        throw CREATE OBJECT Exception()  (3)
    END IF
    trainingImages <- CALL helper.drawBehind
(4) CALL processor.getTrainingImageVector(),
    CALL processor.getW(), CALL processor.getH())
    CALL processor.detectImage(inputFile)
    RETURN CALL trainingImages[CALL processor.getSmallestTwoNorm()]
(5) END recognize
```



Gambar 6.1 Pemodelan algoritma `recognize()` ke dalam *flow graph*
Sumber: Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `recognize()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2
= 5 – 5 + 2
= 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 2 ditentukan dua buah basis set dari jalur independent yaitu:

Jalur 1 : 1-2-4-5

Jalur 2 : 1-2-3-5

b. Pengujian Unit untuk `classify()`

Operasi `processImage()` merupakan implementasi metode yang digunakan untuk mengubah citra biner menjadi matriks. Gambar 6.2 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `processImage()`.

```
METHOD classify PARAMTER img THROWS Exception IS void
DECLARATION:
TYPE Img IS Image
twoNorm IS double
DESCRIPTION:
(1) CALL DoubleMatrix2D testProjectionMatrix
    twoNorm <- CREATE OBJECT numOfImages
    testProjectionMatrix <- CALL
    project(convertArrayToMatrix(findDeviated(grabPixels(img))))
    twoNorm <- CALL find2Norm (weightMatrix, testProjectionMatrix)
    smallestTwoNorm <- CALL findNearestImage(twoNorm)
    OUT "SMALLEST TWO NORM= " smallestTwoNorm
    END classify
```

Node (N) = 1

Edge (E) = 0

(1)

Gambar 6.2 Pemodelan algoritma `classify()` ke dalam *flow graph*
Sumber: Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `classify()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

$$V(G) = E - N + 2$$
$$= 0 - 1 + 2$$
$$= 1$$

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 1 ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1

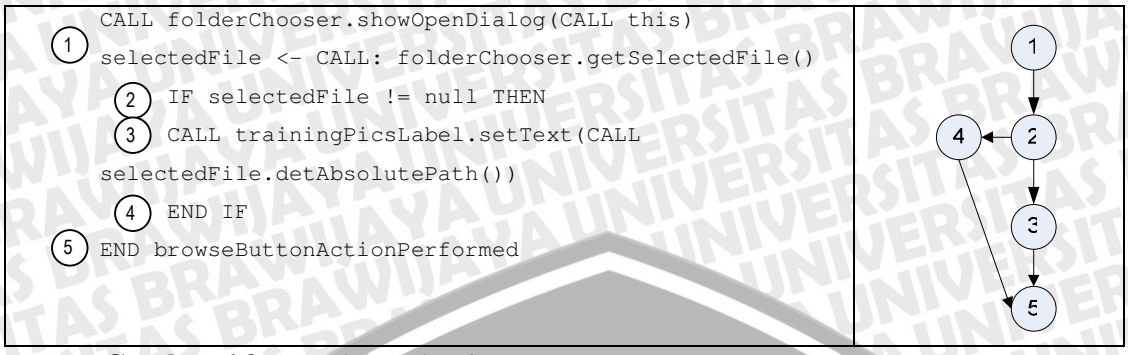
c. Pengujian Unit untuk browseButtonActionPerformed ()

Gambar 6.1 memperlihatkan proses pemodelan dalam *flow graph* pada operasi `browseButtonActionPerformed()`.

```
METHOD browseButtonActionPerformed PARAMETER evt IS void
DECLARATION:
    TYPE
DESCRIPTION:
```

Node (N) = 5

Edge (E) = 5



Gambar 6.3 Pemodelan algoritma `browseButtonActionPerformed()` ke dalam *flow graph*
Sumber: Pengujian

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap operasi `recognize()` menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2.

V(G) = E – N + 2
= 5 – 5 + 2
= 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 2 ditentukan dua buah basis set dari jalur independent yaitu:

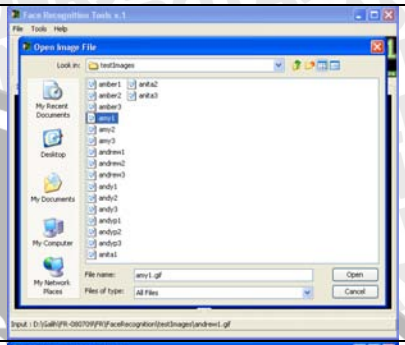
Jalur 1 : 1-2-4-5

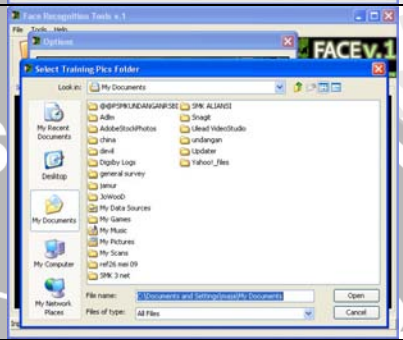
Jalur 2 : 1-2-3-5

6.1.1.2.6.1.2 Pengujian Validasi

Pengujian validasi digunakan untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Item-item yang telah dirumuskan dalam daftar kebutuhan dan merupakan hasil analisis kebutuhan akan menjadi acuan untuk melakukan pengujian validasi. Pengujian validasi menggunakan metode pengujian *Black Box,* karena tidak memerlukan untuk berkonsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan konformitas antara kinerja sistem dengan daftar kebutuhan. Pada skripsi ini dilakukan pengujian validasi terhadap sistem *Face Recognition*.

6.1.1.2.1.6.1.3.1 Kasus Uji Validasi

No	Hasil yang diharapkan	Hasil yang diharapkan	Hasil yang didapatkan	Status validitas
1	Kasus uji menentukan folder input test image	Sistem dapat membuka kotak dialog explorer, kemudian user menentukan folder yang berisi test image. User dapat memilih image yang hendak dijadikan test image		valid
2	Kasus uji menentukan file test image	Sistem dapat menampilkan file image yang dipilih user sebagai test image		valid
3	Kasus uji melakukan recognize terhadap test image	Sistem dapat melakukan pengenalan wajah, dan menampilkan hasil pengenalan wajah		valid
4	Kasus uji melakukan recognize tampa melakukan input image	Sistem menampilkan pesan bahwa tidak dapat melakukan proses recognize		valid

5	Kasus uji merubah folder database image (Training image)	Sistem dapat menampilkan lokasi folder yang sedang digunakan dan memberikan pilihan untuk melakukan perubahan folder		valid
6	Kasus uji memilih folder training image	Sistem dapat memberikan tampilan explorer sehingga memungkinkan user memilih folder training image yang baru		valid
6	Kasus uji menggunakan file yang tidak sesuai dengan format GIF dan dimensi yang sesuai dengan training image	Sistem memunculkan pesan tidak dapat melakukan recognize		valid

Tabel 6.1 Pengujian validasi
Sumber: Pengujian

6.2 Pengujian akurasi algoritma

Pengujian akurasi algoritma dilakukan untuk mengetahui tingkat ketepatan algoritma dalam melakukan face recognition. Pengujian ini dilakukan menggunakan database image dari Yale University yang berformat GIF. Terdiri atas 2 data set yaitu Yale Image Set A dan Yale Image Set B. Yale set A berisi image dengan brightness berbeda, sedangkan Yale set B berisi image dari ekspresi yang berbeda.

6.2.1 Pengujian skenario pertama

Percobaan menggunakan citra test set satu ekspresi kemudian training set terdiri dari orang-orang yang berbeda. Masing masing orang satu ekspresi. Tujuan dari percobaan ini adalah untuk mengetahui kemampuan sistem dalam mencocokkan wajah orang yang sama antara test set dan training set.

No.	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
1.	amber1	Amber	Berhasil
2.	amy1	Amy	Berhasil
3.	andrew1	Andrew	Berhasil
4.	andy1	Andy	Berhasil
5.	andyp1	Andyp	Berhasil
6.	anita1	Anita	Berhasil
7.	arnab1	Arnab	Berhasil
8.	chris1	Chris	Berhasil
9.	dane1	Dane	Berhasil
10.	eric1	Eric	Berhasil
11.	erin1	Erin	Berhasil
12.	gabe1	Gabe	Berhasil
13.	Hill1	Hill	Berhasil
14.	indra1	Indra	Berhasil
15.	jack1	jack1	Berhasil
16.	jill1	jill1	Berhasil
17.	jimmy1	Jimmy	Berhasil
18.	joan1	Joan	Berhasil
19.	kevin1	Kevin	Berhasil
20.	kurt1	Kurt	Berhasil
21.	lance1	Lance	Berhasil
22.	nate1	Nate	Berhasil
23.	paul1	Paul	Berhasil
24.	phil1	Phil	Berhasil
25.	pooja1	Pooja	Berhasil
26.	reuven1	Reuven	Berhasil
27.	tats1	Tats	Berhasil
28.	thad1	Thad	Berhasil
29.	Tim1	Tim	Berhasil
30.	tulika1	Tulika	Berhasil
31.	zach1	Zach	Berhasil

Tabel 6.2 Pengujian skenario 1
Sumber: Pengujian

Total = 31
Berhasil = 31
Tidak berhasil = 0
Persentase keberhasilan = (31/31)*100% = **100%**

6.2.2 Pengujian Skenario kedua

Percobaan menggunakan citra training set yang memiliki berbagai ekspresi dari beberapa orang, Citra test set yang digunakan memiliki ekspresi yang berbeda dengan

training set. Tujuan percobaan ini adalah untuk mengetahui kemampuan sistem dalam mengenali wajah dalam ekspresi yang berbeda-beda

Cara pengujian: satu gambar pertama (dianggap sebagai gambar normal) dari setiap orang diambil sebagai image latih, sedangkan gambar dari orang yang sama tetapi dengan ekspresi yang berbeda digunakan sebagai image uji. Gambar diambil dari 46 orang yang berbeda.

No.	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
1.	amber1	amber	Berhasil
2.	amber2	chris	Tidak Berhasil
3.	amber3	chris	Tidak Berhasil
4.	amy1	amy	Berhasil
5.	amy2	amy	Berhasil
6.	amy3	amy	Berhasil
7.	andrew1	andrew	Berhasil
8.	andrew2	andrew	Berhasil
9.	andrew3	andrew	Berhasil
10.	andy1	andy	Berhasil
11.	andy2	andy	Berhasil
12.	andy3	andy	Berhasil
13.	andyp1	andyp	Berhasil
14.	andyp2	jack	Tidak Berhasil
15.	andyp3	andy	Tidak Berhasil
16.	anita1	anita	Berhasil
17.	anita2	anita	Berhasil
18.	anita3	anita	Berhasil
19.	arnab1	arnab	Berhasil
20.	arnab2	arnab	Berhasil
21.	arnab3	arnab	Berhasil
22.	chris1	chris	Berhasil
23.	chris2	chris	Berhasil
24.	dane1	dane	Berhasil
25.	eric1	eric	Berhasil
26.	eric2	eric	Berhasil
27.	eric3	eric	Berhasil
28.	erin1	erin	Berhasil
29.	erin2	erin	Berhasil
30.	erin3	erin	Berhasil
31.	gabe1	gabe	Berhasil
32.	gabe2	gabe	Berhasil
33.	gabe3	gabe	Berhasil
34.	hill1	hill	Berhasil
35.	hill2	chris	Tidak Berhasil
36.	hill3	chris	Tidak Berhasil

37.	hill4	chris	Tidak Berhasil
38.	indra1	indra	Berhasil
39.	jack1	jack	Berhasil
40.	jack2	jack	Berhasil
41.	jack3	jack	Berhasil
42.	jill1	jill	Berhasil
43.	jill2	jill	Berhasil
44.	jill3	jill	Berhasil
45.	jimmy1	jimmy	Berhasil
46.	jimmy2	jimmy	Berhasil
47.	jimmy3	jimmy	Berhasil
48.	joan1	joan	Berhasil
49.	joan2	joan	Berhasil
50.	joan3	joan	Berhasil
51.	kevin1	kevin	Berhasil
52.	kevin2	kevin	Berhasil
53.	kevin3	kevin	Berhasil
54.	kurt1	kurt	Berhasil
55.	kurt2	jimmy	Tidak Berhasil
56.	kurt3	kurt	Berhasil
57.	lance1	lance	Berhasil
58.	lance2	paul	Tidak Berhasil
59.	lance3	tim	Tidak Berhasil
60.	nate1	nate	Berhasil
61.	nate2	nate	Berhasil
62.	nate3	nate	Berhasil
63.	paul1	paul	Berhasil
64.	paul2	paul	Berhasil
65.	paul3	paul	Berhasil
66.	phil1	phil	Berhasil
67.	phil2	phil	Berhasil
68.	phil3	phil	Berhasil
69.	pooja1	pooja	Berhasil
70.	pooja2	pooja	Berhasil
71.	pooja3	pooja	Berhasil
72.	reuven1	reuven	Berhasil
73.	reuven2	reuven	Berhasil
74.	tats1	tats	Berhasil
75.	tats2	tats	Berhasil
76.	tats3	tats	Berhasil
77.	thad1	thad	Berhasil
78.	thad2	thad	Berhasil
79.	thad3	thad	Berhasil
80.	tim1	tim	Berhasil
81.	tulika1	tulika	Berhasil
82.	tulika2	tulika	Berhasil
83.	tulika3	tulika	Berhasil
84.	zach1	zach	Berhasil

85.	zach2	zach	Berhasil
86.	zach3	zach	Berhasil
87.	crop1a	Crop1a	Berhasil
88.	crop2a	Crop1a	Berhasil
89.	crop3a	Crop1a	Berhasil
90.	crop4a	Crop1a	Berhasil
91.	crop5a	Crop1a	Berhasil
92.	crop6a	Crop2d	Tidak Berhasil
93.	crop1b	Crop2b	Berhasil
94.	crop2b	Crop2b	Berhasil
95.	crop3b	Crop2b	Berhasil
96.	crop4b	Crop2b	Berhasil
97.	crop5b	Crop2b	Berhasil
98.	crop6b	Crop2b	Berhasil
99.	crop1c	Crop2c	Berhasil
100.	crop2c	Crop2c	Berhasil
101.	crop3c	Crop2f	Tidak Berhasil
102.	crop4c	Crop2c	Berhasil
103.	crop5c	Crop2c	Berhasil
104.	crop6c	Crop2m	Tidak Berhasil
105.	crop1d	Crop2d	Berhasil
106.	crop2d	Crop2d	Berhasil
107.	crop3d	Crop2d	Berhasil
108.	crop4d	Crop2d	Berhasil
109.	crop5d	Crop2d	Berhasil
110.	crop6d	Crop2d	Berhasil
111.	crop1e	Crop2e	Berhasil
112.	crop2e	Crop2e	Berhasil
113.	crop3e	Crop 2l	Tidak Berhasil
114.	crop4e	Crop2e	Berhasil
115.	crop5e	Crop2e	Berhasil
116.	crop6e	Crop2e	Berhasil
117.	crop1f	Crop2f	Berhasil
118.	crop2f	Crop2f	Berhasil
119.	crop3f	Crop2n	Tidak Berhasil
120.	crop4f	Crop2f	Berhasil
121.	crop5f	Crop2f	Berhasil
122.	crop6f	Crop2c	Tidak Berhasil
123.	crop1g	Crop2g	Berhasil
124.	crop2g	Crop2g	Berhasil
125.	crop3g	Crop2f	Tidak Berhasil
126.	crop4g	Crop2g	Berhasil
127.	crop5g	Crop2m	Tidak Berhasil
128.	crop6g	Crop2g	Berhasil
129.	crop1h	Crop2h	Berhasil
130.	crop2h	Crop2h	Berhasil
131.	crop3h	Crop2j	Tidak Berhasil
132.	crop4h	Crop2h	Berhasil

133.	crop5h	Crop2h	Berhasil
134.	crop6h	Crop2h	Berhasil
135.	crop1i	Crop2i	Berhasil
136.	crop2i	Crop2i	Berhasil
137.	crop3i	Crop2i	Berhasil
138.	crop4i	Crop2i	Berhasil
139.	crop5i	Crop2i	Berhasil
140.	crop6i	Crop2o	Tidak Berhasil
141.	crop1j	Crop2f	Tidak Berhasil
142.	crop2j	Crop2j	Berhasil
143.	crop3j	Crop2j	Berhasil
144.	crop4j	Crop2j	Berhasil
145.	crop5j	Crop2f	Tidak Berhasil
146.	crop6j	Crop2j	Berhasil
147.	crop1k	Crop2k	Berhasil
148.	crop2k	Crop2k	Berhasil
149.	crop3k	Crop2k	Berhasil
150.	crop4k	Crop2k	Berhasil
151.	crop5k	Crop2k	Berhasil
152.	crop6k	Crop2k	Berhasil
153.	crop1l	Crop2l	Berhasil
154.	crop2l	Crop2l2l	Berhasil
155.	crop3l	Crop2l	Berhasil
156.	crop4l	Crop2l	Berhasil
157.	crop5l	Crop2l	Berhasil
158.	crop6l	Crop2l	Berhasil
159.	crop1m	Crop2m	Berhasil
160.	crop2m	Crop2m	Berhasil
161.	crop3m	Crop2m	Berhasil
162.	crop4m	Crop2m	Berhasil
163.	crop5m	Crop2m	Berhasil
164.	crop6m	Crop2m	Berhasil
165.	crop1n	Crop2j	Tidak Berhasil
166.	crop2n	Crop2n	Berhasil
167.	crop3n	Crop2j	Tidak Berhasil
168.	crop4n	Crop2j	Tidak Berhasil
169.	crop5n	Crop2j	Tidak Berhasil
170.	crop6n	Crop2g	Tidak Berhasil
171.	crop1o	Crop2j	Tidak Berhasil
172.	crop2o	Crop2o	Berhasil
173.	crop3o	Crop2j	Tidak Berhasil
174.	crop4o	Crop2o	Berhasil
175.	crop5o	Crop2o	Berhasil
176.	crop6o	Crop2j	Tidak Berhasil

Tabel 6.3 Pengujian skenario 2
Sumber: Pengujian

Total sampel = 176

Berhasil = 146

Tidak berhasil = 30

Persentase keberhasilan = (146/176)*100% = **82,95%**

6.1.1.2.2.6.2.3 Skenario Pengujian Ketiga

Pengujian Yale Database A (2) → brightness

Metode : setiap gambar dari masing-masing subjek diambil sebagai image latih secara acak, hal yang sama dilakukan untuk image latih

No.	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
1.	Subjek 1E	Subjek 1A	Berhasil
2.	Subjek 2P	Subjek 4G	Tidak Berhasil
3	Subjek 3J	Subjek 10J	Tidak Berhasil
4	Subjek 4I	Subjek 7E	Tidak Berhasil
5	Subjek 5K	Subjek 6N	Tidak Berhasil
6	Subjek 6H	Subjek 6N	Berhasil
7	Subjek 7M	Subjek 11P	Tidak Berhasil
8	Subjek 8J	Subjek 12J	Tidak Berhasil
9	Subjek 9N	Subjek 9D	Berhasil
10	Subjek 10C	Subjek 1A	Tidak Berhasil
11	Subjek 11A	Subjek 8A	Tidak Berhasil
12	Subjek 12D	Subjek 8A	Tidak Berhasil
13	Subjek 13G	Subjek 19A	Tidak Berhasil
14	-	-	-
15	Subjek 15O	Subjek 15J	Berhasil
16	Subjek 16B	Subjek 7E	Tidak Berhasil
17	Subjek 17F	Subjek 17B	Berhasil
18	Subjek 18K	Subjek 20L	Tidak Berhasil
19	Subjek 19F	Subjek 19A	Berhasil
20	Subjek 20E	Subjek 20L	Berhasil
21	Subjek 21L	Subjek 21C	Berhasil

Tabel 6.4 Pengujian skenario 3A
Sumber: Pengujian

Jumlah sampel yang tidak berhasil = 12

Jumlah sampel yang berhasil = 8

Persentase keberhasilan = (8/20)*100% = **40%**

Pengujian dengan Yale Database A (1) → brightness

Metode yang digunakan yaitu: 4 gambar awal dari setiap subjek diambil sebagai image latih, sedangkan 12 gambar sisanya digunakan sebagai image uji. Metode ini telah diujikan pada subjek pertama, di mana jumlah subjek terdapat sebesar 20 subjek.

Test image : subjek 1E – 1P

Training pics : subjek 1A – 1D

No.	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
1	Subjek 1E	Subjek 1D	Berhasil
2	Subjek 1F	Subjek 1C	Berhasil
3	Subjek 1G	Subjek 1C	Berhasil
4	Subjek 1H	Subjek 1B	Berhasil
5	Subjek 1I	Subjek 1B	Berhasil
6	Subjek 1J	Subjek 1B	Berhasil
7	Subjek 1K	Subjek 1A	Berhasil
8	Subjek 1L	Subjek 1B	Berhasil
9	Subjek 1M	Subjek 1C	Berhasil
10	Subjek 1N	Subjek 1A	Berhasil
11	Subjek 1O	Subjek 1B	Berhasil
12	Subjek 1P	Subjek 1B	Berhasil

Tabel 6.5 Pengujian skenario 3B
Sumber: Pengujian

Jumlah sampel yang tidak berhasil = 0
Jumlah sampel yang berhasil = 12
Persentase keberhasilan = (12/12)*100% = **100%**

6.3 Analisa Hasil Pengujian

Dari hasil pengujian didapetkan data-data sebagai berikut:

a. Hasil pengujian skenario pertama (brightness sama, ekspresi sama) menunjukkan angka 100%.
b. Hasil pengujian skenario kedua (brightness sama, ekspresi berbeda) menunjukkan angka 82,95%
c. Hasil pengujian skenario ketiga ada 2 macam yaitu, yang pertama (tingkat brightness sama, pada ekspresi berbeda) menunjukkan angka 100%, yang kedua (tingkat brightness berbeda, ekspresi berbeda) menunjukkan angka 40%

Dari angka-angka tingkat keberhasilan diatas menunjukkan bahwa sistem ini dapat bekerja cukup baik untuk mengenali ekspresi yang berbeda-beda yaitu sebesar 88,37%. Tetapi untuk

tingkat brightness yang berbeda-beda, maka angka keberhasilan menurun menjadi sebesar 40 %.

6.4 Analisa Penyebab Terjadi Kesalahan

Di bawah ini adalah data keluaran dari proses mencari jarak terpendek matrik hasil PCA.

a. kasus pengujian yang berhasil pada test image anita.gif

find2Norm: 1.1898919288449764E9 (amber)
find2Norm: 9.916194869043579E8 (amy)
find2Norm: 1.0838504484318807E9 (andrew)
find2Norm: 1.3873256289514017E9 (andy)
find2Norm: 1.107869044900039E9 (andyp)
find2Norm: 1.022313094289651E8 (anita)
find2Norm: 1.5461027193401077E9 (arnab)
find2Norm: 1.7173784253827999E9 (tulika)
find2Norm: 1.692734623520544E9 (zach)

jarak terpendek antara test image dan training image terdapat pada data ke 6 milik anita.gif

b. kasus pengujian yang gagal pada test image amber.gif

Perhitungan euclidean distance untuk image amber2.gif terhadap training set
find2Norm: 1.4264645406687448E9(amber)
find2Norm: 1.5539888995465586E9 (amy)
find2Norm: 1.5295579195480297E9 (andrew)
find2Norm: 1.2091664491141958E9 (andy)
find2Norm: 1.340480910162337E9 (andyp)
find2Norm: 1.6899250945162091E9 (anita)
find2Norm: 1.5342501999154618E9 (arnab)
find2Norm: 1.2371290161009135E9 (tulika)
find2Norm: 1.3209796082792208E9 (zach)

jarak terpendek diperoleh pada data ke 4 milik andy, seharusnya jarak terpendek yang benar adalah milik data pertama yaitu amber. Dari kasus diatas dapat disimpulkan, bahwa penyebab kesalahan proses *recognize* terjadi pada kesalahan mencari jarak terpendek dari kedua image menggunakan metode Euclidean distance

BAB VII

KESIMPULAN DAN SARAN

7.1 Kesimpulan

Bedasarkan hasil pengujian dan analisis yang dilakukan terhadap kinerja sistem, diambil kesimpulan sebagai berikut:

1. Image dengan format GIF dapat diubah menjadi matrik kemudian direduksi dimensinya menggunakan metode PCA
2. Matrik yang telah direduksi dapat dicari nilai kesamaannya menggunakan metode euclidean distance
3. Tingkat keberhasilan metode PCA dan euclidean distance untuk mengenali ekspresi cukup tinggi yaitu 82,95% dan kemampuan mengenali wajah dalam tingkat brightness berbeda lebih rendah yaitu 40 % pada dataset image dari Yale University

7.2 Saran

Saran yang dapat diberikan untuk pengembangan perangkat lunak aplikasi face recognition menggunakan PCA dan euclidean distance adalah:

1. Aplikasi face recognition dengan metode lain yang memiliki tingkat akurasi lebih tinggi
2. Aplikasi dilengkapi dengan metode face detection, yaitu metode untuk menentukan letak wajah kemudian baru dilanjutkan dengan face recognition, sehingga perangkat lunak ini dapat lebih aplikatif

DAFTAR PUSTAKA

[AND-08] Andrianto, Charnov Dodot, 2008, *Design and Implementation of Face Recognition System Based on the Modified Adaptive Neuro Fuzzy Inference System by Using Webcam*, Stikom

[COP-04] Copeland, Lee, 2004, "A Practitioner's Guide to Software Test Design", Artech House.

[DES-07] Desiani, Anita, 2007, *Kajian Pengenalan Wajah Dengan Menggunakan Metode Face-Arg dan Jaringan Syaraf Tiruan Back Propagation*, Jurusan Matematika Fakultas MIPA Universitas Sriwijaya

[HAD-04] Setiawan, Hadi, 2004, *Pengembangan Model Generatif Pengenalan Wajah pada Latar Belakang, Pose dan Iluminasi yang Bervariasi,* Desertasi Pasca Sarjana Institut Teknologi Bandung, Bandung

[HAR-03] Hariyanto, Bambang, 2003, "Esensi-Esensi Bahasa Pemrograman Java", Informatika, Bandung.

[HAR-04] Hariyanto, Bambang, 2004, "Rekayasa sistem Berorientasi Objek", Informatika, Bandung.

[GAN-06] Gandhe, S.T. Talele, K.T, Keskar, A.G, 2006, *Face Recognition Using Contour Matching,* IAENG International Journal Of Conputer Science, 35:2 IJCS_35_2_06

[INT-09] Introna, D Lucas and Nissenbaum, Helen, 2009, *Facial Recognition Technology, A Surver of Policy and Implementation Issues*, Computer Science and Information Law Depertment, New York University

[KRU-04] Krueger J., Robinson M, Kochelek D., Escara M., 2004, *Obtaining the Eigenface Basis*, The Connexions Project.

[LAW-04] Lawson, Ed, 2004, *Summary: "Eigenface for Recognition"*

[LIW-05] Liwei, Wang. Yan, Zhang. Jufu, Feng, 2005, *On Euclidean Distance of Images*, IEEE Transaction of Pattern Analysis and Machine Intelligence Vol 27 No 8

[NAV-07] Navarette, Pablo dan Solar, Javier R, 2007, *Eigenspace-Based Face Recognition*, Santiago Chile

[PRE-01] Pressman, Roger S, 2001, "Software Engineering: A Practitioner's approach", McGraw-Hill.

[SAP-08] Saputra, Gede Hendra, 2008, *Human Face Recognition Using Bayesian Intrapersonal/Extrapersonal Classifier,*Institut Teknologi Telkom, Bandung

[SOM-03] Sommerville, Ian, 2003, "Software Engineering", Erlangga

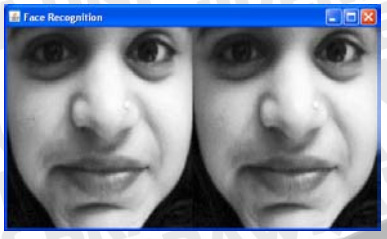
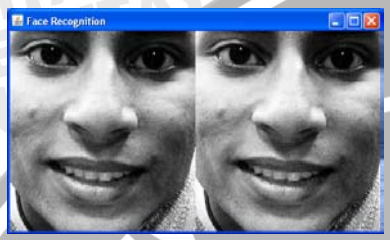
[ZHA-03] Zhao,W. Chellapa, R. dan Phillips, PJ, 2003, *Face Recognition: A Literature Survey.* ACM Computing Surveys, vol 35 no. 4, pp 399-458

LAMPIRAN 1

Data Pengujian Skenario 1

Lampiran 1: Pengujian Skenario 1

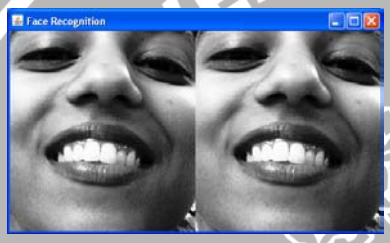
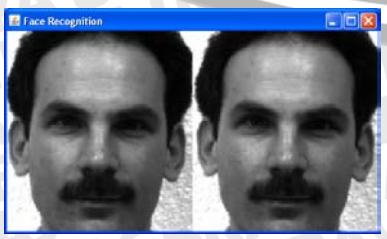
No.	Hasil Running Program	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
32.		amber1	amber	Berhasil
33.		amy1	amy	Berhasil
34.		Andrew1	andrew	Berhasil
35.		andy1	andy	Berhasil
36.		andyp1	andyp	Berhasil

37.		anita1	anita	Berhasil
38.		arnab1	arnab	Berhasil
39.		chris1	chris	Berhasil
40.		dane1	dane	Berhasil
41.		eric1	eric	Berhasil
42.		erin1	erin	Berhasil

				
43.		gabe1	gabe	Berhasil
44.		hill1	hill	Berhasil
45.		indra1	indra	Berhasil
46.		jack1	jack	Berhasil
47.		jill1	jill	Berhasil

48.		jimmy1	jimmy	Berhasil
49.		joan1	joan	Berhasil
50.		kevin1	kevin	Berhasil
51.		kurt1	kurt	Berhasil
52.		lance1	lance	Berhasil
53.		nate1	nate	Berhasil

54.		paul1	paul	Berhasil
55.		phil1	phil	Berhasil
56.		pooja1	pooja	Berhasil
57.		reuven1	reuven	Berhasil
58.		tats1	tats	Berhasil
59.		thad1	thad	Berhasil

				
60.		tim1	tim	Berhasil
61.		tulika1	tulika	Berhasil
62.		zach1	zach	Berhasil
63.		crop1a	Crop1a	Berhasil
64.		crop2b	Crop2b	Berhasil

65.		crop2c	Crop2c	Berhasil
66.		crop2d	Crop2d	Berhasil
67.		crope	Crop2e	Berhasil
68.		crop2f	Crop2f	Berhasil
69.		crop2g	Crop2g	Berhasil
70.		crop2h	Crop2h	Berhasil

71.		crop2i	Crop2i	Berhasil
72.		crop2j	Crop2j	Berhasil
73.		crop2k	Crop2k	Berhasil
74.		crop2l	Crop2l	Berhasil
75.		crop2m	Crop2m	Berhasil
76.		crop2n	Crop2n	Berhasil

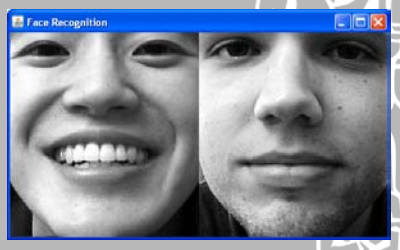
				
77.		crop2o	Crop2o	Berhasil

Total sampel = 46
Berhasil = 46
Tidak berhasil = 0
Persentase keberhasilan = (46/46)*100% = **100%**

LAMPIRAN 2

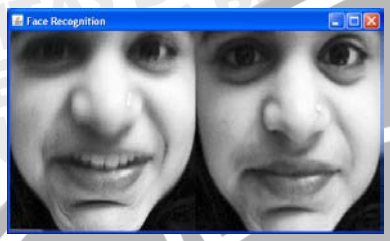
Data Pengujian Skenario 2

Lampiran 2: Pengujian Skenario 2

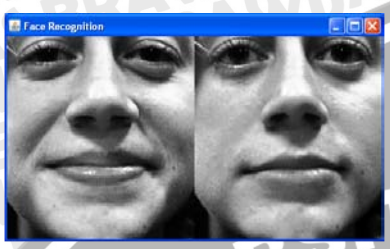
No.	Hasil Running Program	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
78.		amber1	amber	Berhasil
79.		amber2	chris	Tidak Berhasil
80.		amber3	chris	Tidak Berhasil
81.		amy1	amy	Berhasil
82.		amy2	amy	Berhasil

83.		amy3	amy	Berhasil
84.		andrew1	andrew	Berhasil
85.		andrew2	andrew	Berhasil
86.		andrew3	andrew	Berhasil
87.		andy1	andy	Berhasil
88.		andy2	andy	Berhasil

				
89.		andy3	andy	Berhasil
90.		andyp1	andyp	Berhasil
91.		andyp2	jack	Tidak Berhasil
92.		andyp3	andy	Tidak Berhasil
93.		anita1	anita	Berhasil

94.		anita2	anita	Berhasil
95.		anita3	anita	Berhasil
96.		arnab1	arnab	Berhasil
97.		arnab2	arnab	Berhasil
98.		arnab3	arnab	Berhasil
99.		chris1	chris	Berhasil

100.		chris2	chris	Berhasil
101.		dane1	dane	Berhasil
102.		eric1	eric	Berhasil
103.		eric2	eric	Berhasil
104.		eric3	eric	Berhasil
105.		erin1	erin	Berhasil

				
106.		erin2	erin	Berhasil
107.		erin3	erin	Berhasil
108.		gabe1	gabe	Berhasil
109.		gabe2	gabe	Berhasil
110.		gabe3	gabe	Berhasil

111.		hill1	hill	Berhasil
112.		hill2	chris	Tidak Berhasil
113.		hill3	chris	Tidak Berhasil
114.		hill4	chris	Tidak Berhasil
115.		indra1	indra	Berhasil
116.		jack1	jack	Berhasil

117.		jack2	jack	Berhasil
118.		jack3	jack	Berhasil
119.		jill1	jill	Berhasil
120.		jill2	jill	Berhasil
121.		jill3	jill	Berhasil
122.		jimmy1	jimmy	Berhasil

				
123.		jimmy2	jimmy	Berhasil
124.		jimmy3	jimmy	Berhasil
125.		joan1	joan	Berhasil
126.		joan2	joan	Berhasil
127.		joan3	joan	Berhasil

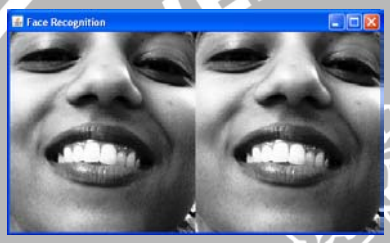
128.		kevin1	kevin	Berhasil
129.		kevin2	kevin	Berhasil
130.		kevin3	kevin	Berhasil
131.		kurt1	kurt	Berhasil
132.		kurt2	jimmy	Tidak Berhasil
133.		kurt3	kurt	Berhasil

134.		lance1	lance	Berhasil
135.		lance2	paul	Tidak Berhasil
136.		lance3	tim	Tidak Berhasil
137.		nate1	nate	Berhasil
138.		nate2	nate	Berhasil
139.		nate3	nate	Berhasil

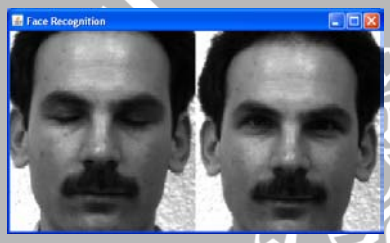
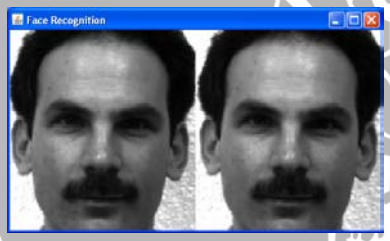
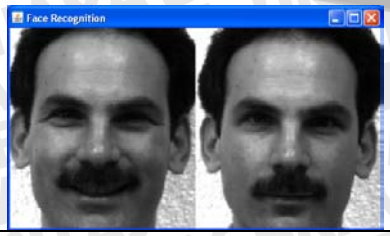
				
140.		paul1	paul	Berhasil
141.		paul2	paul	Berhasil
142.		paul3	paul	Berhasil
143.		phil1	phil	Berhasil
144.		phil2	phil	Berhasil

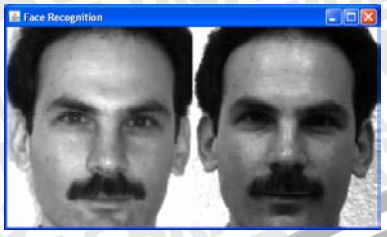
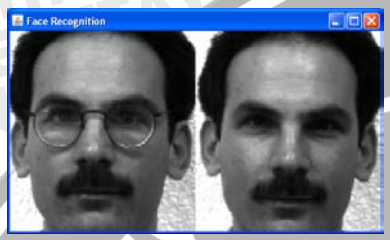
145.		phil3	phil	Berhasil
146.		pooja1	pooja	Berhasil
147.		pooja2	pooja	Berhasil
148.		pooja3	pooja	Berhasil
149.		reuven1	reuven	Berhasil
150.		reuven2	reuven	Berhasil

151.		tats1	tats	Berhasil
152.		tats2	tats	Berhasil
153.		tats3	tats	Berhasil
154.		thad1	thad	Berhasil
155.		thad2	thad	Berhasil
156.		thad3	thad	Berhasil

				
157.		tim1	tim	Berhasil
158.		tulika1	tulika	Berhasil
159.		tulika2	tulika	Berhasil
160.		tulika3	tulika	Berhasil
161.		zach1	zach	Berhasil
162.		zach2	zach	Berhasil

				
163.		zach3	zach	Berhasil
164.		crop1a	Crop1a	Berhasil
165.		crop2a	Crop1a	Berhasil
166.		crop3a	Crop1a	Berhasil
167.		crop4a	Crop1a	Berhasil

168.		crop5a	Crop1a	Berhasil
169.		crop6a	Crop2d	Tidak Berhasil
170.		crop1b	Crop2b	Berhasil
171.		crop2b	Crop2b	Berhasil
172.		crop3b	Crop2b	Berhasil
173.		crop4b	Crop2b	Berhasil

174.		crop5b	Crop2b	Berhasil
175.		crop6b	Crop2b	Berhasil
176.		crop1c	Crop2c	Berhasil
177.		crop2c	Crop2c	Berhasil
178.		crop3c	Crop2f	Tidak Berhasil
179.		crop4c	Crop2c	Berhasil

				
180.		crop5c	Crop2c	Berhasil
181.		crop6c	Crop2m	Tidak Berhasil
182.		crop1d	Crop2d	Berhasil
183.		crop2d	Crop2d	Berhasil
184.		crop3d	Crop2d	Berhasil
185.		crop4d	Crop2d	Berhasil

				
186.		crop5d	Crop2d	Berhasil
187.		crop6d	Crop2d	Berhasil
188.		crop1e	Crop2e	Berhasil
189.		crop2e	Crop2e	Berhasil
190.		crop3e	Crop 2l	Tidak Berhasil

191.		crop4e	Crop2e	Berhasil
192.		crop5e	Crop2e	Berhasil
193.		crop6e	Crop2e	Berhasil
194.		crop1f	Crop2f	Berhasil
195.		crop2f	Crop2f	Berhasil
196.		crop3f	Crop2n	Tidak Berhasil

197.		crop4f	Crop2f	Berhasil
198.		crop5f	Crop2f	Berhasil
199.		crop6f	Crop2c	Tidak Berhasil
200.		crop1g	Crop2g	Berhasil
201.		crop2g	Crop2g	Berhasil
202.		crop3g	Crop2f	Tidak Berhasil

				
203.		crop4g	Crop2g	Berhasil
204.		crop5g	Crop2m	Tidak Berhasil
205.		crop6g	Crop2g	Berhasil
206.		crop1h	Crop2h	Berhasil
207.		crop2h	Crop2h	Berhasil

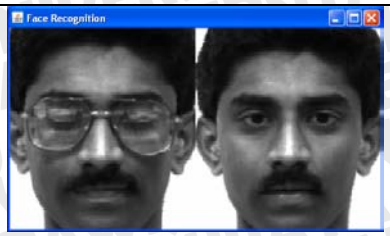
208.		crop3h	Crop2j	Tidak Berhasil
209.		crop4h	Crop2h	Berhasil
210.		crop5h	Crop2h	Berhasil
211.		crop6h	Crop2h	Berhasil
212.		crop1i	Crop2i	Berhasil
213.		crop2i	Crop2i	Berhasil

214.		crop3i	Crop2i	Berhasil
215.		crop4i	Crop2i	Berhasil
216.		crop5i	Crop2i	Berhasil
217.		crop6i	Crop2o	Tidak Berhasil
218.		crop1j	Crop2f	Tidak Berhasil
219.		crop2j	Crop2j	Berhasil

				
220.		crop3j	Crop2j	Berhasil
221.		crop4j	Crop2j	Berhasil
222.		crop5j	Crop2f	Tidak Berhasil
223.		crop6j	Crop2j	Berhasil
224.		crop1k	Crop2k	Berhasil

225.		crop2k	Crop2k	Berhasil
226.		crop3k	Crop2k	Berhasil
227.		crop4k	Crop2k	Berhasil
228.		crop5k	Crop2k	Berhasil
229.		crop6k	Crop2k	Berhasil
230.		crop1l	Crop2l	Berhasil

231.		crop2l	Crop2l2l	Berhasil
232.		crop3l	Crop2l	Berhasil
233.		crop4l	Crop2l	Berhasil
234.		crop5l	Crop2l	Berhasil
235.		crop6l	Crop2l	Berhasil
236.		crop1m	Crop2m	Berhasil

				
237.		crop2m	Crop2m	Berhasil
238.		crop3m	Crop2m	Berhasil
239.		crop4m	Crop2m	Berhasil
240.		crop5m	Crop2m	Berhasil
241.		crop6m	Crop2m	Berhasil

242.		crop1n	Crop2j	Tidak Berhasil
243.		crop2n	Crop2n	Berhasil
244.		crop3n	Crop2j	Tidak Berhasil
245.		crop4n	Crop2j	Tidak Berhasil
246.		crop5n	Crop2j	Tidak Berhasil
247.		crop6n	Crop2g	Tidak Berhasil

248.		crop1o	Crop2j	Tidak Berhasil
249.		crop2o	Crop2o	Berhasil
250.		crop3o	Crop2j	Tidak Berhasil
251.		crop4o	Crop2o	Berhasil
252.		crop5o	Crop2o	Berhasil
253.		crop6o	Crop2j	Tidak Berhasil

Total sampel = 176
Berhasil = 146
Tidak berhasil = 30
Persentase keberhasilan = (146/176)*100% = **82,95%**

LAMPIRAN 3

Data Pengujian Skenario 3

Lampiran 3: Pengujian Skenario 3

No.	Hasil Running Program	TestImage (Input)	TrainingPics (Output)	Berhasil/Tidak berhasil
254.		Subjek 1E	Subjek 1A	Berhasil
255.		Subjek 2P	Subjek 4G	Tidak Berhasil
256.		Subjek 3J	Subjek 10J	Tidak Berhasil
257.		Subjek 4I	Subjek 7E	Tidak Berhasil

				
258.		Subjek 5K	Subjek 6N	Tidak Berhasil
259.		Subjek 6H	Subjek 6N	Berhasil
260.		Subjek 7M	Subjek 11P	Tidak Berhasil

261.		Subjek 8J	Subjek 12J	Tidak Berhasil
262.		Subjek 9N	Subjek 9D	Berhasil
263.		Subjek 10C	Subjek 1A	Tidak Berhasil
264.		Subjek 11A	Subjek 8A	Tidak Berhasil

265.		Subjek 12D	Subjek 8A	Tidak Berhasil
266.		Subjek 13G	Subjek 19A	Tidak Berhasil
267.		Subjek 15O	Subjek 15J	Berhasil
268.		Subjek 16B	Subjek 7E	Tidak Berhasil

				
269.		Subjek 17F	Subjek 17B	Berhasil
270.		Subjek 18K	Subjek 20L	Tidak Berhasil
271.		Subjek 19F	Subjek 19A	Berhasil
272.		Subjek 20E	Subjek 20L	Berhasil

				
273.		Subjek 21L	Subjek 21C	Berhasil

Total sampel = 20
Berhasil = 12
Tidak berhasil = 8
Persentase keberhasilan = (8/20)*100% = **40%**