UJI KEAKURASIAN METODE EIGENFACE DALAM FACE RECOGNITION

SKRIPSI

Diajukan sebagai salah satu syarat untuk kelulusan
Jenjang Strata Satu (S1)
pada Program Studi Teknik Informatika

Oleh:

BINTANG ANUGRAH
NIM. 361501040

SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER

INDONESIA MANDIRI

BANDUNG

2019

LEMBAR PENGESAHAN

UJI KEAKURASIAN METODE EIGENFACE DALAM FACE RECOGNITION

Oleh:

BINTANG ANUGRAH
NIM. 361501040

Draf Skripsi Ini Telah Disetujui Untuk
Diajukan Dalam Sidang Tugas Akhir
Sarjana Teknik Informatika

Pada

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA & KOMPUTER
INDONESIA MANDIRI

Bandung,
Disahkan oleh

Ketua Program Studi,	Dosen Pembimbing,
Chalifa Chazar, S.T.,M.T.	**Dr. Ichsan Ibrahim, S.Si., M.Si.**
NIDN. 0421098704	NIDN. 0411047503

LEMBAR PERSETUJUAN REVISI SKRIPSI

UJI KEAKURASIAN METODE EIGENFACE DALAM FACE RECOGNITION

Oleh:

BINTANG ANUGRAH
361501040

Telah Melakukan sidang tugas akhir dan telah melakukan revisi sesuai dengan Perubahan dan perbaikan yang diminta pada saat sidang tugas akhir.

Bandung, September 2019
Menyetujui

No	Nama	Keterangan	Tanda Tangan
1	Dr.ichsan Ibrahim, S.Si.,M.Si.	Pembimbing	
2	Dr.Chairuddin, lr., M.M., MT	Penguji 1	
3	Haryoso Wicaksono, S.Si., M.M.,M,Kom.	Penguji 2	

Bandung,
Mengetahui,
Ketua Program Studi,

Chalifa Chazar, S.T.,M.T.
NIDN. 042198704

LEMBAR PERNYATAAN

Dengan ini saya menyatakan bahwa:

1) Tugas akhir ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik. Baik di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri maupun perguruan tinggi lainnya.

2) Tugas akhir ini murni merupakan karya penelitian saya sendiri dan tidak menjiplak karya pihak lain. Dalam hal ada bantuan atau arahan dari pihak lain maka telah saya sebutkan identitas dan jenis bantuannya di dalam lembar ucapan terimakasih.

3) Seandainya ada karya pihak lain yang ternyata memiliki kemiripan dengan karya seni saya ini, maka hal ini adalah diluar pengetahuan saya dan terjadi tanpa kesengajaan dari pihak saya.

Pernyataan ini saya buat dengan sesungguhnya dan apabila dikemudian hari terbukti adanya kebohongan dalam pernyataan ini, maka saya bersedia menerima sanksi akademik sesuai norma yang berlaku di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Bandung, 23 September 2019
Penulis

Bintang Anugrah

361501040

ABSTRAK

UJI KEAKURASIAN METODE EIGENFACE DALAM FACE RECOGNITION

Bintang Anugrah

361501040

Pengenalan wajah adalah salah satu teknologi biometrik yang telah banyak diaplikasikan dalam sistem security selain pengenalan retina mata, pengenalan sidik jari dan iris mata. Dalam aplikasinya sendiri pengenalan wajah menggunakan sebuah kamera untuk menangkap wajah seseorang kemudian dibandingkan dengan wajah yang sebelumnya telah disimpan di dalam database tertentu. Ada beberapa macam metoda pengenalan wajah yaitu neural *network*, jaringan syaraf tiruan, neuro fuzzy adaptif dan *eigenface*. Secara khusus dalam paper ini metoda yang akan dijelaskan adalah metoda *eigenface*. Pada paper tugas akhir ini menawarkan metode *Eigenface*, dan menggunakan *webcam* untuk menangkap gambar secara *real-time*. Metode ini mempunyai komputasi yang sederhana dan cepat dibandingkan dengan penggunaan metode yang memerlukan banyak pembelajaran seperti jaringan syaraf tiruan. Secara garis besar proses dari aplikasi ini adalah kamera melakukan capture pada wajah. Kemudian didapatkan sebuah nilai R,G,B. Dengan menggunakan pemrosesan awal, dilakukan *resize*, RGB ke *Gray*,dan *histogram equalisasi* untuk perataan cahaya. Metode *eigenface* berfungsi untuk menghitung *eigenvalue* dan *eigenvector* yang akan digunakan sebagai fitur dalam melakukan pengenalan. Metode *Euclidean distance* digunakan untuk mencari jarak dengan data fitur yang telah didapat , dan jarak terkecil adalah hasilnya. Hasil dari Kombinasi- kombinasi dari metode *eigenface* dapat mendeteksi wajah dengan keakurasian sebesar 50-90%.

Kata Kunci : *EigenFace*, Gambar, Pengenalan Wajah

ABSTRACT

TEST ACCURACY IN EIGENFACE METHOD FACE RECOGNITION

Bintang Anugrah
361501040

Facial recognition system is one of the biometric technologies that have been widely applied in security systems in addition to other biometric tecnologic such as eye retina recognition,iris and fingerprint recognition. In its own application face recognition uses a camera to capture a person's face then compared to selected facial feature from given image with faces withen a databse. There are several types of facial recognition methods, namely neural networks, artificial neural networks, adaptive neuro fuzzy and eigenface. In this paper, used the eigenface methods to recognition facial features and the webcam to captures image in real time. This method has a simple and fast computation compared to the use of methods that require a lot of learning such as artificial neural networks. Then a value of R, G, B is obtained. Using the initial processing, resizing and convert RGB image to Grayscale image, and histogram equalization for light smoothing. The eigenface method functions to calculate the eigenvalue and eigenvector which will be used as a feature in making recognition. Euclidean distance method is used to find the distance with the data features that have been obtained, and the smallest distance is the result. Results from combinations of eigenface methods can detect faces with an accuracy of 50-90%.

Keyword : Eigenface, Images, Face Recognition

UCAPAN TERIMA KASIH

Dengan mengucapkan syukur Alhamdulillah, penelitian ini dapat diselesaikan untuk memenuhi syarat tugas akhir. Laporan penelitian dalam tugas akhir ini di ajukan untuk memenuhi dan melengkapi salah satu syarat akademik dalam kelulusan jenjang Strata Satu (S1) jurusan Teknik Informatika pada Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Penyusunan tugas akhir ini tidak lepas dari dukungan dan bimbingan dari berbagai pihak, maka pada kesempatan ini penulis ingin menyampaikan rasa terimakasih yang sebesar-besarnya kepada:

1. Bapak **Dr. Ichsan Ibrahim, S.Si., M.Si.** selaku Dosen pembimbing yang selalu meluangkan waktu, fikiran dan tenaga dalam memberikan bimbingan, masukan dan saran-sarannya.
2. Bapak Dr. Chairudin, M.T., M.M. selaku Ketua Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM).
3. Ibu Chalifa Chazar, S.T., M.T. selaku Ketua program studi Teknik Informatika Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM).
4. Seluruh Dosen, Staff dan Karyawan Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri (STMIK-IM) yang telah mendidik dan membantu dalam memberika informasi serat motivasi dalam proses studi maupun tugas akhir berlangsung

5. Teruntuk Kedua Orang Tua Tercinta Bapak Usep Husein Hermawan dan Ibu Neneng Heni Supriatna yang sangat penulis sayangi dan cintai. Terimakasih selalu memberikan nasehat, dukungan, didikan, kasih sayang, serta Do'a yang penuh dan tulus.

Akhir kata saya, berharap semoga dengan selesainya laporan penelitian Tugas Akhir ini dapat memberikan manfaat bagi semua pihak serta menambah wawasan bagi pemikiran kita semua. Terimakasih.

Bandung, 23 September 2019
Penulis

Bintang Anugrah

361501040

DAFTAR ISI

LEMBAR PENGESAHAN ..... **i**
LEMBAR PERSETUJUAN REVISI SKRPSI ..... **ii**
LEMBAR PERNYATAAN ..... **iii**
ABSTRAK ..... **ii**
ABSTRAC ..... **v**
KATA PENGANTAR ..... **ivi**
UCAPAN TERIMA KASIH ..... **viii**
DAFTAR ISI ..... **viii**
DAFTAR TABEL ..... **viiii**
DAFTAR GAMBAR ..... **xii**

BAB I PENDAHULUAN
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan Masalah 3
1.4 Batasan Masalah 3
1.5 Metode Penelitian 4
1.6 Sistematika Penulisan 6

BAB II LANDASAN TEORI
2.1 Sistem 8
2.1.1 Pengertian Sistem 8
2.1.2 Karakteristik Sistem 9
2.1.3 Klasifikasi Sistem 12
2.2 *Face Recognition* 13
2.2.1 Proses *Face Recognition* 14
2.2.2 Fungsi *Face Recognition* 16

2.3 *Computer Vision* 17
2.4 *Imagej* 19
2.5 Pengolahan Citra 20
2.5.1 Citra 20
2.5.2 Model Citra 21
2.5.3 RGB 22
2.5.4 Colour Filtering 23
2.5.5 Grayscale 24
2.5.6 Trehshold 24
2.5.7 Segmentasi Citra 25
2.6 *Visual Studio .NET* 26
2.7 Metode *Eigenface* 28
2.7.1 *Principal Component Analysis* (PCA) 30
2.8 OpenCV Library 30
2.8.1 Fitur Pada *OpenCV* 31
2.10 EmguCV 32
2.11 Konsep Dasar Basis Data 34
2.11.1 Pengertian *Database* Dan *Database Management System* (DBMS) 34
2.11.2Kegunaan Basis Data 35
2.11.3Komponen Basis Data 35
2.12 *Unified Modeling Language* (UML) 37
2.12.1*Use Case* Diagram 37
2.12.2 *Activity* Diagram 39
2.12.3 *Class* Diagram 40
2.12.4 *Sequence Diagram* 42
2.13 *Black Box testing* 44

BAB III ANALISIS DAN PERANCANGAN

3.1 *Communication* 45
3.1.1 Pengumpulan Data 45
3.1.2 Studi Literatur 46
3.1.3 Analisis Masalah 48
3.1.4 Analisis Metode *EigenFace* 48
3.1.4.1 Proses *Image Test* 49
3.1.4.2 Perhitungan *Eigenvalue* dan *Eigenvektor* 50
3.1.5 *Spesifikasi Hardware* 52
3.1.6 *Spesifikasi Software* 53
3.2 *Planning* (*Estimating, Scheduling, Tracking*) 53
3.3 Perancangan (*Design*) 55
3.3.1 *Use Case* Diagram 55
3.3.2 *Activity* Diagram 56
3.3.3 *Class Diagram* 59
3.3.4 *Sequence Diagram* 60
3.3.5 Perancangan *User Interface* 61

BAB IV IMPLEMENTASI DAN UJI COBA

4.1 *Construction* (*Code & Test*) 63
4.1.1 Implementasi 63
4.1.2 Pengujian *Training* wajah 66
4.1.3 Proses Penyimpanan Data 68
4.1.4 Proses *Eigenface* 69
4.1.5 Proses *ImageJ* 72

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan 80

5.2 Saran..... 80

DAFTAR PUSTAKA..... 80

LAMPIRAN_LAMPIRAN

DAFTAR TABEL

TABEL : 2.1. Simbol-simbol Use Case Diagram 38
TABEL : 2.2. Simbol-Simbol Activity Diagram 39
TABEL : 2.3. Simbol-Simbol Class Diagram 41
TABEL : 2.4. Simbol-Simbol Sequence Diagram 42
TABEL : 3.1. Studi Literatur 47
TABEL : 3.2. Penjadwalan Penelitian 54
TABEL : 4.1. Proses Pencarian Data 76
TABEL : 4.1. Pengujian Dengan Metode Black-Box 77

DAFTAR GAMBAR

GAMBAR : 2.1. Simbol Black box Testing 21
GAMBAR : 2.2. Simbol Black box Testing 44
GAMBAR : 3.1. Use Case Diagram Face Recognition 55
GAMBAR : 3.2. Activity Diagram Membuka Kamera 56
GAMBAR : 3.3. Activity Mendaftar Wajah 57
GAMBAR : 3.4. Activity Pendeteksian Wajah 58
GAMBAR : 3.5. Class Proses Face Recognition 59
GAMBAR : 3.6. Sequence Mendaftar 60
GAMBAR : 3.7. Sequence Mendeteksi 61
GAMBAR : 3.8. Halaman Home 61
GAMBAR : 3.9. Notifikasi Berhasil Menyimpan 62
GAMBAR : 3.10. Notifikasi Gagal Menyimpan 62
GAMBAR : 4.1. Halaman Membuka Kamera 64
GAMBAR : 4.2. Halaman Saat Mendaftar Wajah 65
GAMBAR : 4.3. Halaman Mengenali Wajah 65
GAMBAR : 4.4. Pengujian Pemotongan Wajah 66
GAMBAR : 4.5. Proses Citra Asli ke Grayscale 67
GAMBAR : 4.6. Proses Merubah Grayscale ke Tresshold 67
GAMBAR : 4.7. Proses Penyimpanan Data 68
GAMBAR : 4.8. Matriks Eigenvektor 69
GAMBAR : 4.9. Matriks Eigenvalue 70
GAMBAR : 4.10. Hasil Perhitungan Matriks 71
GAMBAR : 4.11. Proses Awal Dalam Dua Duah Wajah 73
GAMBAR : 4.12. Proses Find Edges 73
GAMBAR : 4.13. Hasil Proses Citra Grayscale ke Find Edges 74
GAMBAR : 4.14. Proses Compare Image 74
GAMBAR : 4.15. Proses Citra untuk Mengetahui Parameter 75

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Wajah manusia merupakan salah satu bagian dalam tubuh manusia yang sangat penting. Empat bagian dari panca indra manusia yaitu mata, telinga, lidah, dan hidung terdapat pada wajah manusia. Empat bagian tersebut memiliki peranan penting dalam keberlangsungan hidup manusia yaitu mata untuk melihat, telinga untuk mendengar, lidah untuk merasakan rasa, dan hidung sebagai indra penciuman. Empat indra tersebut ditambah dengan kulit merupakan alat manusia untuk dapat melakukan kegiatan dan menjalani hidup. Setiap manusia memiliki empat indra pada wajahnya dengan bentuk yang berbeda-beda antara satu manusia dengan manusia yang lain. Misalnya, bentuk mata, ukuran hidung, ukuran telinga, tekstur lidah, dan lain-lain.

Wajah manusia yang terdiri dari empat indra tersebut merupakan suatu obyek pertama yang dilihat oleh manusia lain. Wajah manusia juga merupakan obyek yang berperan sebagai pengetahuan akan identitas diri dari manusia tersebut. Ketika manusia lain bertemu dengan manusia yang lain, obyek pertama yang akan mereka lihat dari lawannya adalah wajah. Dari wajah lawan yang manusia lihat tersebut, ia dapat mengidentifikasi identitas pribadi dari manusia lawan tersebut seperti misalnya nama,

umur, tempat tanggal lahir, tempat tinggal, tempat berkuliah, jurusan, bahkan nomor telepon dan nama orang tua dari manusia tersebut

Setiap manusia memiliki wajah dengan bentuk dan struktur yang berbeda antar satu manusia dengan manusia yang lain. Dari teori tersebut muncullah suatu ilmu untuk mengenali wajah manusia (*face recognition*), yang dapat diaplikasikan dalam banyak bidang. Misalnya dalam bidang keamanan, dapat membantu mencari seorang penjahat menggunakan foto wajahnya, lalu pengenalan wajah juga dapat membantu perusahaan atau perkantoran dalam mengurus presensi atau kehadiran pegawai-pegawainya, dan masih banyak aplikasi pengenalan wajah pada bidang-bidang yang lain.

Face Recognition (Pengenalan Wajah) ini telah menjadi salah satu bidang yang banyak diteliti dan juga di kembangkan oleh para pakar pengenalan pola. Suatu citra digital dalam hal ini citra *grayscale* (citra abu-abu) dengan *pixel* sebagai ukuran citranya, memiliki kesesuaian dengan ukuran pada *matriks* yaitu baris dan kolom. Sehingga sebuah citra *grayscale* dapat digambarkan sebagai suatu matriks pada proses pengenalan wajah, citra digital yang berupa sampel wajah dari sampel wajah yang lain. Dimana ciri tersebut akan digunakan pada proses klasifikasi untuk mengklasifikasikan wajah yang ingin dikenali berdasarkan fitur-fitur yang dipilih. Proses ekstraksi fitur pada *eigenface* menggunakan metode menggunakan metode berbasis matrik yang disebut *Principal Component Analysis* (PCA). Sedangkan pada proses klasifikasi,

eigenfaces menggunakan ukuran kedekatan dua vektor yaitu *euclidean distance*.

(Abdillah : 2014)

Dari latar belakang di atas pada penelitian ini akan dibuat aplikasi menggunakan metode *eigenface* untuk mengenali wajah manusia, maka judul penelitian yang akan di ambil adalah "Uji Keakurasian Metode *Eigenface* Dalam *Face Recognition*"

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas adalah sebagai berikut:

Berapa tingkat keberhasilan atau akurasi yang dihasilkan dalam mengenali wajah dengan menggunakan metode *EigenFace*?

1.3 Tujuan Masalah

Tujuan dari penelitian yang ingin di dapatkan antara lain:

Mendapatkan tingkat keberhasilan atau akurasi pengenalan wajah dengan menggunakan metode *Eigenface.*

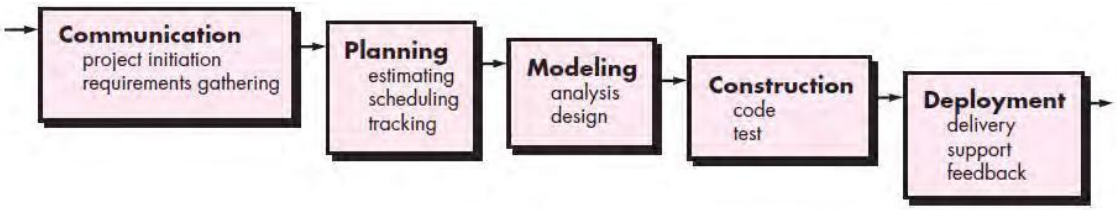
1.4 Batasan Masalah

Batasan masalah dalam perancangan aplikasi berbasis desktop:

1. Penulis hanya melakukan penelitian skripsi pada hasil training deteksi yang menggunakan *face recognition.*
2. Penelitian ini hanya melakukan sampai *construction*

1.5 Metode Penelitian

Metode penelitian dalam penelitian ini mengacu pada SDLC (*System Development Life Cycle*), Metode SDLC yang digunakan adalah Metode *Waterfall* GAMBAR 1.1 menjelaskan mengenai tahapan-tahapan dari Metode *Waterfall*.



GAMBAR 1.1. *Waterfall* pressman (Pressman, 2015)

Adapun penjelasan detail dari tahapan *waterfall* pada gambar 1.1 adalah sebagai berikut :

1. *Communication (Project Initiation & Requirements Gathering)*

   Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan *customer* demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendifinisikan fitur dan fungsi *software.* Pengumpulan data-data tambahan bisa diambil dari jurnal, artikel, dan internet.

2. *Planning (Estimasing, Scheduling & Tracking)*

   Tahap berikutnya adalah tahapan perncanaan yang mejelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk, kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan *tracking* proses pengerjaan sistem.

3. *Modeling (Analysis & Design)*

   Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur software, tampilan interface, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

4. *Construction (Code & Test)*

   Tahapan *construction* ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

5. *Deployment (Delivery, Support, Feedback)*

   Tahapan *deployment* merupakan tahapan implementasi *software* ke *customer*, pemeliharaan *software* secara berkala, perbaikan *software*, evaluasi *software*, dan pengembangan *software* berdasarkan umpan balik yang diberikan agar

sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya. (Pressman, 2015)

1.6 **Sistematika Penulisan**

Penulisan laporan Tugas Akhir ini, disusun per bab. Gambaran isi setiap bab antara lain :

BAB I : PENDAHULUAN

Bab ini memuat tentang latar belakang masalah, rumusan masalah, tujuan masalah, batasan masalah, metode penelitian serta sistematika penelitian.

BAB II : LANDASAN TEORI

Bab ini memuat tentang konsep dan landasan teori yang berhubungan dengan judul skripsi

BAB III : ANALISIS DAN PERANCANGAN

Bab ini memuat tentang perancangan sistem menggunakan *Unified Modeling Language* (UML) yang meliputi analisis pelaku, analisis sistem yang diusulkan (*Use Case Diagram, Activity Diagram* dan *Class Diagram*) menggunakan metode pengembangan model *waterfall* (Pressman, 2015).

BAB IV : IMPLEMENTASI & ANALISIS HASIL

Bab ini berisi memuat tentang penjelasan dalam pengoprasian program serta evaluasi implementasi program.

BAB V : KESIMPULAN DAN SARAN

Bab ini memuat kesimpulan yang didapat dari hasil penelitian serta saran dari hasil analisis penulis tentang permasalahan yang dibahas.

BAB II
LANDASAN TEORI

2.1 Sistem

2.1.1 Pengertian Sistem

Secara sederhana, suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling bergantung satu sama lain, dan terpadu. Menurut Gordon B. Davis dalam bukunya menyatakan, sistem bisa berupa abstrak atau fisis. Sistem yang abstrak adalah susunan yang teratur dari gagasan- gagasan atau konsepsi yang saling bergantung. Sedangkan sistem yang bersifat fisis adalah serangkaian unsur yang bekerjasama untuk mencapai suatu tujuan. Menurut Sutarman, "Sistem adalah kumpulan elemen yang saling berhubungan dan berinteraksi dalam satu kesatuan untuk menjalankan suatu proses pencapaian suatu tujuan utama (Sutarman, 2012).

Menurut Mustakini, "Sistem dapat didefinisikan dengan pendekatan prosedur dan pendekatan komponen, sistem dapat didefinisikan sebagai kumpulan dari prosedur-prosedur yang mempunyai tujuan tertentu". Terdapat dua kelompok pendekatan dalam mendefinisikan sistem, yaitu:

Pendekatan Sistem yang lebih menekankan pada prosedur, Mendefinisikan sistem sebagai suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan.

Berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk meneyelesaikan suatu sasaran tertentu. Pendekatan yang lebih menekankan pada elemen atau komponennya mendefinisikan sistem sebagai suatu kumpulan dari elemen-elemen yang saling berinteraksi untuk mencapai suatu tujuan tertentu. Berdasarkan beberapa pendapat yang dikemukakan di atas dapat ditarik kesimpulan bahwa sistem adalah suatu kumpulan bagian-bagian baik manusia atau pun bukan manusia yang saling berinteraksi untuk mencapai suatu tujuan.

2.1.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat yang tertentu yaitu mempunyai komponen-komponen (*components*), batas sistem (*boundary*), lingkungan luar sistem (*environment*), penghubung (*interface*), masukan (*input*), keluaran (*output*), pengolah (*process*) dan sasaran (*objectives*) atau tujuan (*goal*) (Jogiyanto, 1999).

1. Komponen Sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Suatu sistem tidak membatasi betapapun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem.

2. Batasan Sistem

   Batasan sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini

memungkinkan suatu sistem dipandang sebagai satu kesatuan. Batas suatu sistem menunjukan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Sistem

Lingkungan luar sistem dari suatu sistem adalah semua diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya. Keluaran (*output*) dari satu subsistem akan menjadi masukan (*input*) untuk subsistem lainnya dengan melalui penghubung. Dengan penghubung satu subsistem dapat berinteraksi dengan subsistem yang lainnya membentuk satu kesatuan.

5. Masukan Sistem

Masukan adalah energi yang dimasukan kedalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukan supaya sistem tersebut dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh didalam sistem komputer, program adalah *maintenance input* yang digunakan untuk mengoperasikan komputernya dan data adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem

Keluaran adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. Keluaran dapat merupakan masukan untuk subsistem yang lain. Misalnya untuk sistem komputer, panas yang dihasilkan adalah keluaran yang tidak berguna yang merupakan hasil sisa pembuangan, sedang informasi adalah keluaran yang dibutuhkan.

7. Pengolah Sistem

Sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah yang akan merubah masukan menjadi keluaran. Suatu sistem produksi akan mengolah masukan berupa bahan baku dan bahan-bahan yang lain menjadi keluaran berupa barang jadi.

8. Sasaran Sistem

Sistem pasti mempunyai tujuan atau sasaran. kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

2.1.3 Klasifikasi Sistem

sistem dapat diklasifikasikan dari beberapa sudut pandangan, diantaranya adalah sebagai berikut :

1. Sistem abstrak (*abstract sistem*) dan sistem phisik (*physical sistem*). Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara phisik. Sistem phisik merupakan sistem yang ada secara phisik, misalnya sistem komputer.
2. Sistem alamiah (*natural sistem*) dan sistem buatan (*human made sistem*). Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak di buat manusia. Misalnya sistem perputaran bumi. Sistem buatan manusia adalah sistem yang dirancang oleh manusia. Misalnya sistem informasi akuntansi karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.
3. Sistem tertentu (*deterministic sistem*) dan sistem tidak tentu (*probabilistic sistem*). Sistem tertentu beropersi dengan tingkah laku yang sudah dapat diprediksi. Interaksi diantara bagianbagiannya dapat dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Sistem komputer adalah contoh dari sistem tertentu yang tingkah lakunya dapat dipastikan berdasarkan program-program yang dijalankan. Sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilitas.

4. Sistem tertutup (*closed sistem*) dan sistem terbuka (*open sistem*) Sistem tertutup merupakan sistem yang tidak berhubungandengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur tangan dari pihak luarnya. Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Karena sistem sifat terbuka dan terpengaruh oleh lingkungan luarnya, maka suatu sistem harus mempunyai suatu sistem pengendalian yang baik (Jogiyanto, 2005).

2.2 *Face Recognition*

Face recognition adalah salah satu teknologi biometrik yang telah banyak diaplikasikan dalam sistem *security* selain pengenalan retina mata, pengenalan sidik jari dan iris mata. Dalam aplikasinya sendiri pengenalan wajah menggunakan sebuah kamera untuk menangkap wajah seseorang kemudian dibandingkan dengan wajah yang sebelumnya telah disimpan di dalam database tertentu. *Face recognition* adalah teknologi dari komputer yang memungkinkan kita untuk mengidentifikasi atau memverifikasi wajah seseorang melalui sebuah gambar digital. Caranya ialah dengan mencocokkan tekstur lekuk wajah kita dengan data wajah yang tersimpan di database (Minartiningtyas, Brigida Arie, 2013).

2.2.1 Proses *Face Recognition*

Dalam sistem keamanan biometrik dengan pengenalan struktur bentuk wajah ini membutuhkan peralatan kamera dalam pengidentifikasiannya. Adapun *device* pada *face recognition system* bekerja sebagai pengenal kode yang bekerja pada objek muka seseorang. *Device* ini mengambil kode berdasarkan bentuk geometri wajah. Jenis pengambilan data informasi pada *device* ini dibagi menjadi 2 (dua) tipe, yaitu tipe pengambilan secara 2D dan tipe pengambilan secara 3D. Tapi pada kenyataannya, penggunaan 3D lebih menguntungkan karena lebih spesifik untuk kode pengenal. Sehingga banyak perangkat keamanan yang menggunakan *face recognition system* dengan tipe 3D (Minartiningtyas, Brigida Arie, 2013).

Berikut adalah cara kerja pada *device face recognition system* yaitu:

1. Pendeteksian Wajah

   Pendeteksian wajah dilakukan dengan pengambilan foto wajah dari manusia dengan memindai foto 2D secara digital, atau bisa juga menggunakan video untuk mengambil foto wajah 3D.

2. Penjajaran

   Setelah wajah berhasil dideteksi, *software* akan dapat menentukan posisi, ukuran, dan sikap kepala. Pada *software* 3D foto wajah mampu dikenali hingga 90 derajat, sedangkan untuk software 2D posisi kepala harus menghadap kamera paling tidak 35 derajat

3. Pengukuran

   Selanjutnya *software* dapat mengukur lekukan yang ada pada wajah dengan menggunakan skala sub-milimeter (*microwave*) dan membuat *template*.

4. Representasi

   Kemudian jika *template* sudah jadi maka template tersebut dapat diterjemahkan kedalam sebuah kode yang unik, yang mempresentasikan setiap wajah.

5. Pencocokan

   Jika foto wajah yang telah direpresentasikan dan ketersediaan foto wajah dalam basis data sama-sama 3D, proses pencocokan dapat langsung dilakukan. Namun, saat ini masih ada tantangan untuk mencocokkan representasi 3D dengan basis data foto 2D. Teknologi baru kini tengah menjawab tantangan ini. Ketika foto wajah 3D diambil, *software* akan mengidentifikasikan beberapa titik (biasanya tiga titik) yaitu mata bagian luar dan dalam, serta ujung hidung. Berdasarkan hasil pengukuran ini *software* akan mengubah gambar 3D menjadi 2D, dan membandingkannya dengan gambar wajah 2D yang sudah ada di dalam basis data.

6. *Verifikasi* atau *Identifikasi*

   Verifikasi merupakan proses pencocokkan satu berbanding satu. Sedangkan *identifikasi* adalah perbandingan foto wajah yang diambil dengan seluruh gambar yang memiliki kemiripan dalam *database*.

7. Analisis Tekstur Wajah

   Kemajuan dalam *software face recognition* adalah penggunaan biometrik kulit atau keunikan tekstur kulit untuk meningkatkan akurasi hasil pencocokkan. Namun terdapat beberapa faktor yang menyebabkan proses analisis tekstur ini tidak dapat bekerja, misalnya pantulan cahaya dari kacamata atau foto wajah yang menggunakan kacamata matahari. Faktor penghambat analisis lainnya adalah rambut panjang yang menutupi bagian tengah wajah, pencahayaan yang kurang tepat (yang mengakibatkan foto wajah menjadi kelebihan atau kekurangan cahaya), serta resolusi yang rendah (foto diambil dari kejauhan).

2.2.2 Fungsi *Face Recognition*

Secara umum sistem pengenal ini banyak digunakan untuk mengenali wajah di kerumuman atau tempat banyak orang. ada beberapa fungsi terkait penggunaan sistem ini.

1. Beberapa agensi pemerintah juga menggunakannya untuk keamanan dan mengeliminasi kecurangan saat Pemilihan Umum. Pemerintah AS baru-baru ini memulai program yang disebut US-VISIT (*United States Visitor and Immigrant Status Indicator Technology*), yang ditujukan untuk turis asing yang masuk ke Amerika Serikat. Program ini membandingkan sidik jari dan foto turis ketika mengurus visanya dengan pusat data kriminal dan tersangka teroris.

2. Sistem ini juga banyak di gunakan untuk mengenali wajah pada bandara dan Bank atau tempat layanan pemerintah lainnya. Program ini akan melakukan pemindaian dalam waktu cepat untuk para pelanggan yang dengan sukarela memberikan informasi dan menyelesaikan perkiraan ancaman keamanan. Di bandara akan ada antrean khusus untuk *Registered Traveler* yang akan bergerak lebih cepat dengan memverifikasi turis dari fitur wajah mereka.
3. Penggunaan fitur pengenalan wajah juga tidak memerlukan kontak (*contactless*), sehingga pengguna tidak perlu bersentuhan dengan alat. Hal ini jelas lebih higienis untuk mencegah penularan penyakit. Karenanya fitur ini sangat cocok digunakan untuk rumah sakit, laboratorium, pabrik farmasi atau makanan, serta area-area publik.Tapi sama halnya dengan banyak teknologi yang sedang berkembang, potensi luar biasa fitur pengenalan wajah juga hadir dengan beberapa kekurangan. Yang paling disoroti adalah penggunaan sistem tanpa sepengetahuan Anda. Di tempat-tempat umum, sistem mengambil gambar pengunjung tanpa izin dan sepengetahuan mereka (Minartiningtyas, Brigida Arie, 2013).

2.3 *Computer Vision*

Computer Vision merupakan suatu proses otomatis yang mengitegrasikan sejumlah besar proses untuk persepsi awal, seperti akuisisi citra, pengolahn citra, klasifikasi, pengenalan (*recognition*) dan membuat keputusan. *Computer Vision* mencoba meniru

bagaimana cara kerja sistem visualmanusia (*Human Vision*) yang sebenarnya sangat kompleks. Objek atau citra yang dilihat ditangkap oleh manusia dengan indera penglihatan kemudian dilanjutkan ke otak untuk diinterpretasi sehingga manusia dapat mengerti objek apa yang terlihat dalam pandangan penglihatannya. Hasil interpretasi ini dapat dipakai untuk pengambilan keputusan (misal menghindar jika melihat ada mobil didepan).

Computer Vision didefinsikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali objekyang diamati atau diobservasi. Cabang ilmu ini bersama Intelejensia Semu (*Artificial Intelegence*) akan mampu menghasilkan sistem Intelijen Visual (*Visual Intellegence System*). Perbedaannya adalah *Computer Vision* lebih mempelajari bagaimana proses komputerdalam mengobservasi suatu objek. Namun komputer grafika lebih kearah pemanipulasian gambar (visual) secara digital. Bentuk sederhana dari grafika komputer adalah grafika komputer 2D dan kemudian berkembang menjadi grafika 3D. Pemprosesan citra (*image processing*), dan pengenalan pola (*pattern recognition*).

Computer vision adalah kombinasi antara pengolahan citra dan pengenalan pola. Pengolahan citra (*image processing*) merupakan bidang yang berhubungan dengan proses transformasi citra atau gambar (*image*). Proses ini bertujuan untuk mendapatkan kualitas citra yang lebih baik. Sedangkan pengenalan pola (*pattern recognition*), bidang ini berhubungan denga proses identifikasi objek pada citra atau interpretasi citra. Proses ini bertujuan untuk mengekstrak informasi atau pesan yang disampaikan oleh suatu citra.

Computer Vision diaplikasikan secara beragam, mulai dari *vision system* untuk mesin industri sebagai pemeriksa botol selama produksi, sampai kepada riset untuk kecerdasan buatan menggunakan komputer ataupun robot agar dapat mengenali lingkungan sekitar. Bidang *Computer Vision* meliputi teknologi analisis citra secara otomatis yang digunakan dibanyak bidang. *Machine vision* biasanya mengacu pada proses yang menggabungkan analisis citra secara otomatis dengan metode dan teknologi pengamatan otomatis dan bimbingan robot untuk aplikasi industri (Rinaldi, 2004).

2.4 *Imagej*

ImageJ adalah domain publik, berbasis Java program pengolah gambar yang dikembangkan di *National Institutes of Health*. *ImageJ* juga memiliki *plugin* Java dan *macro recordable*. akuisisi Kustom, analisis dan pengolahan *plugin* dapat dikembangkan dengan menggunakan *ImageJ built-in editor* dan *compiler* Java. *plugin user*-ditulis memungkinkan untuk memecahkan banyak masalah pengolahan gambar dan analisis, dari pencitraan sel secara dimensi tiga untuk pengolahan gambar radiologi, perbandingan sistem multi pencitraan. Data untuk sistem hematologi otomatis. arsitektur *plugin ImageJ* dan *built-in* lingkungan pengembangan telah membuatnya menjadi *platform* populer untuk mengajar pengolahan gambar (Ferreira dan Rasbon, 2012).

2.5 Pengolahan Citra

Pengolahan citra atau image processing adalah suatu proses yang dilakukan oleh suatu sistem pada masukan (*input*) berupa citra (*image*) yang mengasilkan (*output*) sehingga menjadi citra (*image*) yang kualitasnya lebih baik. Pengolahan citra bertujuan untuk memperbaiki kualitas citra agar mudah diinterpretasikan oleh manusia atau mesin (dalam hal ini komputer). Namun dengan berkembangnya dunia komputasi yang ditandai dengan semakin meningkatnya kapasitas dan kecepatan proses komputer, serta munculnya ilmu-ilmu komputer yang memungkinkan manusia dapat mengambil informasi dari suatu citra maka *image processing* tidak dapat dilepaskan dengan bidang *computer vision*.

2.5.1 Citra

Citra atau *image* adalah kombinasi antara titik, garis, bidang, dan warna untuk menciptakan suatu imitasi dari suatu objek, biasanya berupa objek fisik atau manusia. Citra dapat berwujud gambar (*image*) dua dimensi, seperti lukisan, foto, dan juga dapat berwujud tiga dimensi, seperti patung. Dalam tinjauan matematis, citra merupakan fungsi kontinyu dari intensitas cahaya pada bidang yang dapat disimpan dalam bentuk atau format digital. Citra tersebut oleh sebuah komputer didefinisikan sebagai sebuah bidang (dengan luas tertentu) yang terdiri dari banyak titik dengankoordinat tertentu. Banyaknya titik dalam bidang inilah yang dinamakan pixel. Semakin banyak pixel, citra yang dihasilkan semakin jelas

2.5.2 Model Citra

Citra terdiri dari citra kontinyu dan citra diskrit. Citra kontinyu dihasilkan dari sistem optik yangmenerima sinyal analog, misalnya mata manusia dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinyu. Beberapa sistem optik dilengkapi dengan fungsi digitalisasi sehingga ia mampu menghasilkan

citra diskrit, misalnya kamera digital dan *scanner*. Citra diskrit disebut juga citra digital.

Komputer digital yang umum dipakai saat ini hanya dapat mengolah citra digital. Citra digital merupakan suatu matriks dimana indeks baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar/ *pixel*/ piksel/ *pels/ picture element*) menyatakan tingkat

keabuan pada titik tersebut. Piksel sendiri memiliki dua parameter yaitu koordinat dan intensitas atau warna. Nilai yang terdapat pada titik (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu. Oleh sebab itu citra digital dapat (Rinaldi, 2004).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & \dots & \dots & f(1,M-1) \\ \dots & \dots & \dots & \dots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

GAMBAR : 2.1 *Matriks* Citra Digital

15 Komputer digital yang umum dipakai saat ini hanya dapat mengolah citra digital. Citra digital merupakan suatu matriks dimana indeks baris dan kolomnya menyatakan

suatu titik pada citra tersebut dan elemen matriksnya (yang disebut sebagai elemen gambar/ *pixel*/ piksel*/ pels/ picture element*) menyatakan tingkat keabuan pada titik tersebut. Piksel sendiri memiliki dua parameter yaitu koordinat dan intensitas atau

warna. Nilai yang terdapat pada titik (x,y) adalah f(x,y), yaitu besar intensitas atau warna dari piksel di titik itu. Oleh sebab itu citra digital dapat ditulis dalam bentuk gambar sebagai berikut. Gambar 2.2 Matriks Citra Digital Berdasarkan gambaran tersebut, secara matematis citra digital dapat dituliskan sebagai fungsi intensitas f(x,y), dimana harga x(baris) dan (kolom) merupakan koordinat posisi dan f(x,y) adalah nilai fungsi pada setiap titik (x,y) yang menyatakan besar intensitas citra atau tingkat keabuan atau warna dari piksel dititik tersebut. Pada proses digitalisasi diperoleh besar baris M dan kolom N hingga citra membentuk matrik M x N dan jumlah tingkat keabuan piksel G.

2.5.3 RGB

RGB sering digunakan didalam sebagian besar aplikasi komputer karena dengan ruang warna ini, tidak diperlukan transformasi untuk menampilkan informasi di layar monitor. Alasan diatas juga menyebabkan RGB banyak dimanfaatkan sebagai ruang warna dasar bagi sebagian besar aplikasi. Model warna RGB adalah model warna berdasarkan konsep penambahan kuat cahaya primer yaitu *Red,* Greendan *Blue*. Dalam suatu ruang yang sama sekali tidak ada cahaya, maka ruangan tersebut adalah gelap total. Tidak ada signal gelombang

16 cahaya yang diserap oleh mata kita atau RGB (0,0,0). Apabila ditambahkan cahaya merah pada ruangan tersebut, maka ruangan akan berubah warna menjadi merah misalnya RGB (255,0,0), semua benda dalam ruangan tersebut hanya dapat terlihat berwarna merah. Demikian juga apabila cahaya diganti dengan hijau atau biru.

Apabila diberikan 2 macam cahaya primer dalam ruangan tersebut seperti (merah dan hijau), atau (merah dan biru) atau (hijau dan biru), maka ruangan akan berubah warna masing-masing menjadi kuning, atau magenta atau cyan. Warna-warna yang dibentuk oleh kombinasi dua macam cahaya tersebut disebut warna sekunder. Warna Tersier adalah warna yang hanya dapat terlihat apabila ada tiga cahaya primer, jadi apabila dinon-aktifkan salah satu cahaya, maka benda tersebut berubah warna. Contoh warna tersier seperti abu-abu, putih. Pada perhitungan dalam program-program komputer model warna direpresentasi dengan nilai komponennya, seperti dalam RGB (r, g, b) masing-masing nilai antara 0 hingga 255 sesuai dengan urutan masing-masing yaitu pertama *Red*, kedua Green dan ketiga adalah nilai Blue dengan demikian masing-masing komponen ada 256 tingkat. Apabila dikombinasikan maka ada 256 x 256 x 256 atau 16.777.216 kombinasi warna RGB yang dapat dibentuk (Rinaldi 2004).

2.5.4 Colour Filtering

Colour filtering adalah metode yang berguna untuk menemukan sebuah warna yang terdapat pada sebuah citra atau gambar, kita dapat menentukan proses apa yang selanjutnya harus dilakukan. Pada dasarnya pencarian ini menggunakan kombinasi dari

komponen RGB yang terdapat pada citra. kombinasi RGB kemudian dijadikan filter untuk kemudian diproses oleh komputer (Rinaldi 2004).

2.5.5 Grayscale

Grayscale atau derajat keabuan merupakan proses awal dalam image processing yang merupakan suatu metode dalam sistem yang dibangun untuk dilakukan konversi citra dari berbagai format warna seperti RGB, BGR, RGBA, YCrCb, HSV. Tujuannya adalah untuk melakukan pencarian batas atau *edge* atau

19 tepi citra lalu menggambarkan tepi-tepi tersebut sehingga komputer akan dapat dengan mudah melihat bentuk atau kontur-kontur yang terdapat dengan citra. Setelah komputer dapat melihat bentuk tersebut maka komputer dapat dengan mudah melakukan berbagai proses komputasi terutama yang berhubungan dengan *recognizing, tracking*, dan *motion* pada suatu gambar bergerakbaik itu dari disk atau pun *real-time* dengan sumber input menggunakan kamera atau *webcam* (Rinaldi, 2004)

2.5.6 Trehshold

Threshold merupakan konversi citra hitam putih ke citra biner dilakukan dengan cara mengelompokkan nilai derajat keabuan setiap pixel kedalam 2 kelas, hitam dan putih. Pada citra hitam putih terdapat 256 level, artinya mempunyai skala “0” sampai “255” atau [0,255], dalam hal ini nilai intensitas 0 menyatakan hitam, dan nilai

intensitas 255 menyatakan putih, dan nilai antara 0 sampai 255 menyatakan warna keabuan yang terletak antara hitam dan putih (Rinaldi, 2004).

2.5.7 Segmentasi Citra

Segmentasi citra merupakan teknik untuk memisahkan atau mengisolasi suatu objek atau sebagian dari objek dari suatu image (memisahkan *fore ground* dengan *background*). Banyak situasi dalam *computer vision* yang membutuhkan segmentasi atau memisahkan foreground atau suatu objek terhadap *background*-nya guna melihat aktivitas pixel dari objek yang ingin kita lihat. Misalkan ketika suatu kamera keamanan dilengkapi pendeteksi pergerakan manusia atau objek lainnya yang menyerupai manusia dan beberapa binatang, maka kamera tersebut harus terlebih dahulu dapat melakukan segmentasi terhadap objek manusia ataupun binatang tertentu.Segmentasi biasa dilakukan dengan maksud agar pencarian pixel tidak terlalu memakan waktu dan *memory*, sehingga ketika terjadi komputasi pencarian tidak memerlukan pencarian secara menyeluruh pada suatu citra, cukup pencarian pada daerah yang biasa disebut *region of interest* sehingga dapat menghemat waktu dan *memory*. Pencarian yang dimaksud biasanya berupa proses tracking dan motion atau pergerakan suatu pixel. Komputasi pendeteksian motion ini akan terjadi hanya pada setiap point yang telah di segmentasi atau bisa disebut pada objek yang telah dispesifikkan. Ada banyak metode atau algoritma yang dapat digunakan dalam segmentasi bahkan dengan algoritma image *processing* biasa seperti morphology, flood fill, *threshold*, dan *pyramid* akan tetapi terdapat algoritma tersendiri yang dapat menghasilkan segmentasi yang cukup

baik terutama dalam segmentasi warna yang dipakai dalam penelitian ini.Dalam pembuatan kontur diperlukan rutin untuk melakukan proses pencarian kontur. Citra yang dipakai adalah citra hasil konversi menjadi grayscale dan dikenakan *threshold* sehingga menghasilkan citra bi-level (Rinaldi, 2004).

2.6 *Visual Studio .NET*

Menurut Marlon (2010) Pengertian dari *Microsoft Visual Basic* .Net yaitu sebuah alat yang digunakan untuk membangun aplikasi yang bergerak di atas sistem NET *Framework* dengan menggunakan bahasa basic. *Programmer* bisa membangun aplikasi *Windows Form*, aplikasi *web* berbasis ASP.NET, dan juga aplikasi *command-line* dengan alat ini. Alat ini dapat diperoleh secara terpilih dari beberapa produk lainnya (seperti *microsoft visual c++, visual c#, atau visual j#*), atau juga dapat diperoleh secara terpadu dalam *microsoft Visual Studio* .NET.

Ada beberapa tampilan *interface* pada *Microdoft Visual Basic* yang saling berhubungan yaitu :

1. Bagian Menu Program
2. Bagian *Toolbar*
3. Bagian *Toolbox*
4. Bagian Jendela *Properties*
5. Bagian *Form*
6. Bagian Jendela Proyek

7. Bagian Jendela *Layout*

Berikut ini adalah beberapa komponen yang secara langsung sering terlibat dalam pembuatan program menggunakan *Visual Basic* Komponen-komponen tersebut diantaranya adalah :

a. Project

ketika program pertama kali menggunakan *visual basic* maka digunakan akan komponen *project* yang akan diload, langkah selanjutnya adalah membuat modul atau menambah form-form atau mungkin membuat kode program.

b. Form

Form biasanya digunakan ketika kita akan meletakkan *object* apa saja yang akan dipakai dalam program, didalam t*oolbox* terdapat *object-object* yang diletakkan dan didesain dalam bagian *form. Form* merupakan suatu objek yang dipakai sebagai tempat bekerja program aplikasi, akan tersedia secara otomatis *form* yang baru bila membuat program aplikasi yang baru, yaitu dengan nama *form1*. Umumnya didalam suatu form terdapat garis titik-titik yang disebut dengan *Grid*.

c. ToolBox

Toolbox merupakan kotak alat yang berisi *icon-icon* untuk memasukkan objek tertentu ke dalam jendela *form. Toolbox* ini juga dapat dimodifikasi misalnya

dengan menambah komponen icon dengan cara melakukan klik kanan pada *toolbox* lalu memilih *Components* atau *Add Tab*.

d. Properties

Property digunakan untuk menentukan setting suatu objek. Didalam satu objek biasanya memiliki beberapa *property* yang dapat diatur langsung dari jendela *properties* atau lewat kode program. Dalam *property* kita bisa mendapatkan informasi mengenai warna, tinggi, lebar dan posisi sebuah objek. Pada tiap *property* memiliki nilai yang dapat mempengaruhi cara objek ditampilkan atau cara objek bekerja, dan juga bagi sebuah objek *property* biasanya mirip dengan variabel lokal dalam prosedur. *Property* langsung saling berkaitan dengan objek dan digunakan oleh proses-proses yang ada dalam objek.

e. Kode program

Kode program merupakan serangkaian kata yang memiliki perintah yang akan dilaksanakan jika suatu objek dijalankan. Kode program ini akan mengontrol dan menentukan jalannya suatu objek.

f. Event

Event adalah pristiwa atau kejadian yang diterima oleh suatu objek misalnya klik, drag, tunjuk dan lainnya.

2.7 Metode *Eigenface*

Kata eigenface sebenarnya berasal dari bahasa jerman yaitu "*eigenwert*" dimana "*eigen*" artinya karakteristik dan "*wert*" artinya nilai. *Eigenface* adalah salah satu algoritma pengenalan pola wajah yang berdasarkan pada *Principle Component*

Analysis (PCA) yang dikembangkan di MIT. Banyak penulis lebih menyukai istilah *eigen image*. Teknik ini telah digunakan pada pengenalan tulisan tangan, pembacaan bibir, pengenalan suara dan pencitraan medis. Menurut layman (Al Fatta, 2009) *Eigenface* adalah sekumpulan unsur wajah yang telah dibuat standar yang diambil dari analisis statistik dari banyak gambar wajah. *Algoritma eigenface* secara keseluruhan cukup sederhana. *Training image* direpresentasikan dalam sebuah *vector flat* (gabungan *vektor*) dan digabung bersama-sama menjadi sebuah matriks tunggal. *Eigenface* dari masing-masing citra kemudian diekstraksi dan disimpan dalam *file temporary* atau *database. Test image* yang masuk didefinisikan juga nilai *eigenfaces*-nya dan dibandingkan dengan *eigenfaces* dari *image database* atau *file temporary* (Prasetyo dan Rahmatun,2008).

Cara perhitungannya dilakukan dengan cara:

1. Pencarian matriks A berukuran N x N dapat dihitung dengan :

$$A.x=\lambda.x$$

λ dinamakan *eigenvalue* dari matriksA, sedangkan x merupakan *eigenvector* yang sama dengan scalar (λ).

2. Pencarian determinan dari matriksA dengan rumus :

$$|A-1\lambda|=0$$

3. Setelah nilai *eigenvalue* (λ) ditemukan langkah selanjutnya adalah mencari *eigenvektor* dengan menggunakan rumus :

$$A-\lambda I.x=0$$

2.7.1 *Principal Component Analysis* (PCA)

Principal Component Analysis (PCA) adalah sebuah transformasi linier yang biasa digunakan untuk mereduksi data. *Principal Component Analysis* (PCA) adalah sebuah teknik statistika yang berguna pada bidang pengenalan, klasifikasi dan mereduksi data citra. PCA juga merupakan teknik yang umum digunakan. Karena *Principal Component Analysis* (PCA) sangat ampuh untuk mereduksi data baik seperti teks, citra, dan sinyal.

2.8 OpenCV Library

OpenCV Library adalah singkatan dari *Open Computer Vision*, yaitu suatu *library* gratis yang dikembangkan oleh *Intel Corporation* yang di khususkan untuk melakukan *image prosessing*. Tujuaannya adalah agar komputer mempunyai kemampuan yang mirip dengan cara pengolahan visual pada manusia. *OpenCV* mempunyai API (*Aplication Programming Interface*) untuk *high level* maupun *low level*, terdapat fungsi-fungsi yang siap pakai, baik untuk loading, saving, akuisisi gambar maupun video.

Library ini terdiri dari fungsi-fungsi *Computer Visiondan* API (*Aplication Programming Interface*) untuk *image processing high level* maupun *low level* dan sebagai optimasi aplikasi *real-time*. *OpenCV* sangat disarankan untuk programmer yang akan berkutat pada bidang *Computer Vision*, karena *library* ini mampu menciptakan aplikasi yang handal, kuat dibidang digital vision, dan mempunyai

kemampuan yang mirip dengan cara pengolahan visual pada manusia, karena *library* ini bersifat cuma-cuma dan sifatnya yang *open source*, maka dari itu *OpenCV* tidak dipesan khusus untuk pengguna arsitektur Intel, tetapi dapat dibangun pada hampir semua arsitektur. Saat ini para *developer* dari *Intel Corporation* telah membuat berbagai macam versi, yaitu:

1. *OpenCV* untuk bahasa pemograman C/C++
2. *OpenCV* untuk bahasa pemograman C# (masih dalam tahap pengembangan)
3. *OpenCV* untuk bahasa pemograman Java.

Untuk bahasa pemograman C# dan Java, karena masih dalam tahap pengembangan, maka kita membutuhkan *library* lain sebagai pelengkap kekurangan yang ada. Namun untuk bahasa pemograman C/C++ tidak memerlukan *library* lainnya untuk pemrosesan pada *computer vision*.

2.8.1 Fitur Pada *OpenCV*

Berikut ini adalah fitur2 pada *library OpenCV* :

1. Manipulasi data gambar (alokasi memori, melepaskan memori, kopi gambar, setting serta konversi gambar).
2. Image/Video I/O (Bisa menggunakan kamera yang sudah didukung oleh *library* ini).

3. Manipulasi matrix dan vektor serta terdapat juga *routines linear algebra* (products, solvers, *eigenvalues*, SVD).
4. *Image processing* dasar (*filtering, edge detection*, pendeteksian tepi, sampling dan interpolasi, konversi warna, operasi morfologi, *histograms*, image *pyramids*).
5. Analisis struktural.
6. Kalibrasi kamera.
7. Pendeteksian gerak.
8. Pengenalan objek.
9. Basic GUI (*Display* gambar/video, *mouse/keyboard kontrol, scrollbar*).
10. *Image Labelling (line, conic, polygon, text drawing)*.

2.10 EmguCV

OpenCV (*Open Source Comput er Vision*) adalah sebuah *library* fungsi pemrograman *real time* untuk computer vision.Emgu CV adalah wrapper .Net untuk OpenCV. Dengan EmguCV, fungsi-fungsi dalam OpenCV bisa dipanggil melalui bahasa pemrograman yang *compatible* dengan .NET seperti C#, VB, dan VC++. Selain itu, Emgu CV juga cross platform sehingga dapat di-compile lewat Mono dan dijalankan di atas sistem operasi Linux atau Mac OS. Dari pengertian di atas telah diberikan deskripsi dari kedua *open source* tersebut. OpenCV merupakan *Library* yang cukup terkenal di dunia *Computer Vision. Computer Vision* adalah salah satu bidang di teknologi informasi yang fokus pada pemrosesan.

images atau gambar yang diperoleh dari dunia nyata untuk di ekstrak dan diinterpretasikan informasinya. Untuk mempermudah *depelover* dalam mengembangkan aplikasi yang menggunakan teknologi *computer vision*, digunakanlah *library* seperti VXL, Camellia, OpenCV, dan lainnya.

Maka dari itu EmguCV berperan untuk menjembatani C# dan OpenCV. EmguCV adalah wrapper .Net untuk OpenCV. Keuntungan menggunakan EmguCV yang paling utama adalah *library* ini sepenuhnya ditulis dengan bahasa pemrograman C# yang mana lebih aman karena pembuatanobjectatau *punreferencedi-manage* oleh garbage *collector*. Ada dua konsep penting yang perlu diketahui terlebih dahulu sebelum menggunakan EmguCV. Pertama mengenailayerpada EmguCV. EmguCV terdiri dari 2 layer, yaitu *basic layer* dan *second layer. Basic layer* mengandung fungsi, struktur, dan enumerasi yang secara langsung merefleksikan apa yang ada di OpenCV. Dengan adanya layer inilah kita bisa memanggil fungsi-fungsi pada OpenCV dengan bahasa pemrograman C#. Sedangkan second layer mengandung kelas-kelas yang memanfaatkan keunggulan teknologi .NET (Muliawan, 2015).

2.11 Konsep Dasar Basis Data

Dari prosedur yang digunakan, ada tiga pengertian konsep basis data, sebagai berikut:

2.11.1 Pengertian *Database* Dan *Database Management System* (DBMS)

Basis data adalah jenis program komputer yang berfungsi sebagai pengorganisasian dan mengatur data. Basis data ini merupakan kumpulan file-file yang mempunyai kaitan satu file dengan file lainnya.

Pengertian basis data menurut Fathansyah dalam bukunya adalah : "Kumpulan dari data yang saling berhubungan dan disimpan secara bersama-sama sedemikian rupa dan tanpa pengulangan (*rududansi*) yang tidak perlu, untuk

memenuhi berbagai kebutuhan" (Fathansyah, Basis Data, 1999). Prinsip basis data adalah pengaturan data, tujuan dari basis data adalah untuk memudahkan mempercepat dalam pengambilan data.

Sedangkan *Database Management System* adalah untuk mendefinisikan, membuat, memelihara, dan mengontrol akses database, pengguna memerlukan seebuah sistem perangkat lunak. Berikut keuntungan menggunakan DBMS :

a. Menciptakan data yang lebih konsisten.
b. Memudahkan dalam mengontrol data yang redundan.
c. Mengetahui informasi dari data yang sama.
d. Meningkatkan integritas data.

e. Meningkatkan keamanan.

2.11.2 Kegunaan Basis Data

Untuk memanipulasi data yang sudah tersimpan pada perangkat keras komputer dan menggunakan perangkat lunak komputer, untuk keperluan penyediaan informasi lebih lanjut yang perlu diorganisasikan sedemikian rupa agar informasi yang dihasilkan berkualitas.

2.11.3 Komponen Basis Data

Komponen basis data terdiri dari:

1. Perangkat Lunak

   Merupakan komponen yang menghubungkan fisik basis data.

2. Struktur Relasional

   Struktur ini merupakan struktur basis data yang paling umum. Dalam model relasional, semua elemen data dalam basis data ditampilkan dalam bentuk tabel-tabel yang sederhana.

3. Struktur Berorientasi Objek

   Kemampuan *encapsulation* memungkinkan struktur ini dapat menangani dengan lebih tipe data yang lebih kompleks (grafik, gambar, suara, teks) daripada struktur basis data yang lainnya.

4. Perangkat Keras

   Merupakan komponen yang dibutuhkan untuk manajemen basis data. Basis data biasanya memiliki sebuah perangkat keras, diantaranya:

   a. Kompter
   b. Memory sekunder *On-Line* (Berupa *Hardisk*).
   c. Memory sekunder yang *Off-Line* keperluan *Back-up*.
   d. Perangkat komunikasi (Untuk Jaringan).

5. Basis Data

   Setiap basis data dapat memiliki sejumlah objek sistem basis data. Selain menyimpan data, juga untuk menyimpan definisi struktur.

6. Sistem Operasi

   Sistem Operasi merupakan program aplikasi yang mengaktifkan system komputer dan untuk mengendalikan seluruh sumber daya dalam komputer dan melakukan operasi dasar dalam komputer.

7. Sistem Pengolah Basis Data

   Pemakai tidak melakukan pengolahan secara fisik tetapi melakukan secara langsung dengan ditangani oleh sebuah perangkat lunak yang khusus, perangkat ini disebut DBMS yang menentukan bagaimana organisasi akan disimpan dan diubah kemudian dikembalikan untuk menerapkan mekanisme pengaman data.

8. Pemakai

   Sistem basis data ini terdiri dari beberapa pemakai yang kemudian dibedakan menjadi beberapa bagian, diantaranya :

   a. *Programmer* Aplikasi
   b. *User* Mahir
   c. *User* Umum
   d. *User* Khusus

2.12*Unified Modeling Language* (UML)

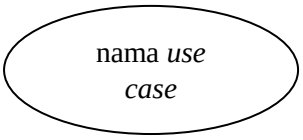
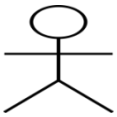
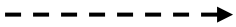
Menurut Rosa dan Salahuddin (2018) *Unified Modeling Language* (UML) adalah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemograman berorientasi objek. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

2.12.1 *Use Case* Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*Behavior*) dan mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Rosa dan Salahuddin, 2018). *Use case* ini bertujuan untuk mengetahui siapa saja yang dapat mengakses sistem dan apa saja

fungsi yang ada di dalam sistem tersebut. Berikut adalah simbol-simbol yang ada pada diagram *use case* :

TABEL: 2.1. Simbol-simbol *Use Case* Diagram (Rosa dan Salahuddin, 2018)

Simbol	Nama	Deskripsi
	*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case.*
 nama aktor	Aktor / *actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri. Biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
	Asosiasi / *Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
	*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan.
	Generalisasi / *Generalization*	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case*

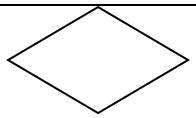
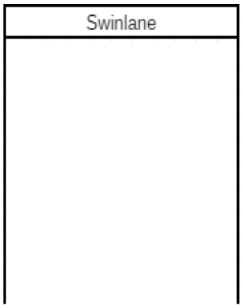
		dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
<<include>>	*Include*	*Include* berarti *use case* yang ditambahkan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan.

2.12.2 *Activity* Diagram

Menurut Rosa dan Salahuddin (2018) *Activity Diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. *Activity Diagram* ini banyak digunakan untuk mendefinisikan urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan. Berikut adalah simbol-simbol yang ada pada *Activity Diagram* :

TABEL : 2.2. Simbol-simbol *Activity Diagram* (Rosa dan Salahuddin, 2018)

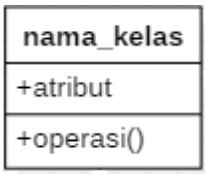
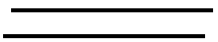
Simbol	Nama	Deskripsi
	Status awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas	Aktivitas	Aktivitas yang dilakukan sistem biasanya diawali dengan kata kerja.

	Percabangan / *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
	Penggabungan / *Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	Status akhir	Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
	*Swimlane*	Memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi

2.12.3 *Class* Diagram

Menurut Rosa dan Salahuddin (2018) *class Diagram* adalah diagram yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Sedangkan metode atau operasi adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Berikut adalah simbol-simbol yang ada pada *class diagram.*

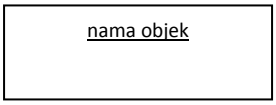
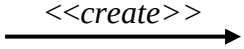
TABEL : 2.3 Simbol-simbol *Class* Diagram (Rosa dan Salahuddin, 2018)

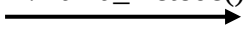
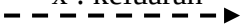
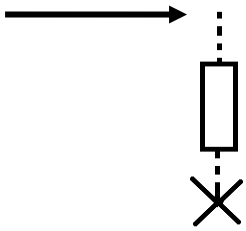
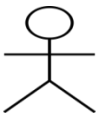
Simbol	Nama	Deskripsi
	Kelas	Kelas pada struktur sistem.
 nama_interface	Antarmuka / *Interface*	Sama dengan konsep *interface* dalam pemograman berorientasi objek.
	Asosiasi / *Association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
	Asosiasi berarah / *Directed Association*	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
	Generalisasi	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity.*
	Kebergantungan/ *Dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.
	Agregasi / *Aggregation*	Relasi antarkelas dengan makna semua-bagian (*whole-part*).

2.12.4 *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek (Rosa dan Salahuddin, 2018). Karenanya dalam membuat *sequence diagram* harus mengetahui terlebih dahulu objek-objek yang ada pada *use case,* semakin banyak *use case* yang didefinisikan maka *sequence diagram* yang harus dibuat juga semakin banyak. Dalam *sequence diagram* diperoleh pada tabel 2.4.

TABEL : 2.4. Simbol-simbol *Sequence Diagram* (Rosa dan Salahudiin, 2018)

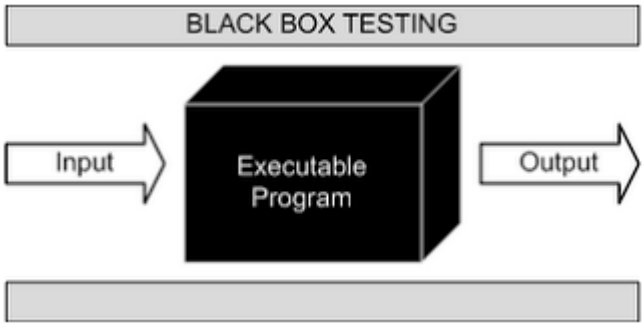
Simbol	Nama	Deskripsi
	Garis hidup / *Lifeline*	Menyatakan kehidupan suatu objek.
	Objek	Menyatakan objek yang berinteraksi pesan.
	Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.
	Pesan tipe *create*	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.

Simbol	Nama	Deskripsi
x : nama_metode() 	Pesan tipe *call*	Menyatakan suatu objek memanggil operasi/ metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode sehingga operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
x : masukan 	Pesan tipe *send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
x : keluaran 	Pesan tipe *return*	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu timbal balik ke objek tertentu.
	Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.
	Aktor / *actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem.

2.13 *Black Box testing*

Black Box Testing adalah untuk mencari kesalahan/kegagalan dalam operasi tingkat tinggi, yang mencakup kemampuan dari perangkat lunak, operasional/tata laksana, skenario pemakai. Fungsi dari pengujian ini berdasarkan kepada apa yang dapat dilakukan oleh sistem. Untuk melakukan pengujian perilaku seseorang harus mengerti lingkup dari aplikasi, solusi bisnis yang diberikan oleh aplikasi, dan tujuan sistem dibuat.

Contoh pengujian pada aplikasi internet banking, maka pengujian yang dilakukan adalah menjalankan aplikasi, memeriksa apakah semua fungsi pada aplikasi berjalan dengan baik serta mengecek tampilan dari aplikasi tersebut apakah sesuai dengan design yang sudah ditentukan atau belum.



Gambar 2.2 Simbol *Black Box Testing*

BAB III

ANALISIS DAN PERANCANGAN

.

Dalam penelitian ini digunakan metode pengembangan SDLC (*System Development life cycle*) dengan Model *Waterfall* (Pressman, 2015). Ada lima tahap dalam pengembangan namun dalam penelitian ini dibatasi sampai dengan tahap empat, keempat tahap tersebut yaitu, *Communication, Planning, modeling, dan Construction.*

3.1 *Communication*

Pada tahap ini dilakukan proses pengumpulan informasi berupa data yang berkaitan dengan penelitian.

3.1.1 Pengumpulan Data

Pengumpulan data merupakan salah satu rangkaian proses dalam penelitian ini yang bertujuan untuk memperoleh data-data serta informasi-informasi terhadap kasus yang menjadi permasalahan dalam tugas akhir ini. Informasi yang sangat butuhkan penulis dalam penelitian ini adalah informasi-informasi mengenai metode *Eigenface* dalam sebagai dasar dalam *face recognition* dan *algoritma Haar-like Feature* sebagai dasar dalam pengenalan wajah. Dalam memperoleh informasi atau pengumpulan data tersebut penulis melakukan dua pendekatan, diantaranya adalah :

a. Studi Pustaka

   Melakukan studi kepustakaan terhadap berbagai referensi yang berkaitan dengan penelitian yang dilakukan. Topik-topik yang akan di kaji antara lain meliputi: pengenalan wajah, pengolahan citra digital.

b. Diskusi

   Pendekatan ini dilakukan untuk berdiskusi secara langsung dengan orang-orang yang memahami baik sebagian maupun keseluruhan dari kasus yang dibahas dalam laporan ini tentang bagaimana masalah serta solusi untuk perancangan dan analisa dari aplikasi yang di teliti ini.

3.1.2 Studi Literatur

Peneliti melakukan studi literatur dengan mengumpulkan data-data dengan membaca, dan memahami referensi yang berasal dari buku-buku, jurnal-jurnal penelitian, dan sumber pustaka lainnya yang berkaitan dengan penelitian. Metode yang digunakan *Eigenface* berkaitan dengan penelitian, berikut adalah tabel 3.1 literatur yang menjadi referensi penelitian :

TABEL : 3.1 Studi Literatur

NO	Literatur	Pembahasan
1.	Sayeed Al-Aidid dan Daniel S. Pamungkas "Sistem Pengenalan Wajah dengan Algoritma Haar Cascade dan Local Binary Pattern Histogram" Jurnal Rekayasa elektrika Volume.14,1,2018	Penelitian ini dibuat untuk pengenalan wajah dengan Algoritma Haar cascade
2.	Panji Purwanto, Burhanudin Dirgantoro Ir.,M.T "Implementasi Face Identification dan Face Recognition Pada Kamera Pengawas Sebagai Pendeteksi Bahaya" Jurnal Fakultas teknik Vol.2,1,2015	Penelitian ini dibuat untuk identifikasi wajah pada kamera pengawas, dengan akurasi deteksi wajah 80% dan *face recognition* 100%
3.	Candra Noor Santi "Mengubah Citra Berwarna Menjadi GrayScale dan Citra Biner" junal Teknologi Informasi DINAMIK Vol 16,1,2011	Penelitian ini dibuat untuk mengubah citra berwarna menjadi *grayscale*

NO	Literatur	Pembahasan
4.	Dian Esti Pratiwi "Implementasi Pengenalan Wajah Menggunakan PCA" Jurnal Ilmu Komputer dan Elektronika Vol.3,2,2013	Penelitian ini dibuat untuk mengImplementasikan *Face Recognition* Menggunakan PCA. Tingkat Akurasi dari hasil penelitian ini sebesar 82,2%

3.1.3 Analisis Masalah

Analisis masalah merupakan salah satu tahapan penelitian dalam menentukan serta menganalisis masalah-masalah yang berhubungan dengan penelitian, rincian analisis masalah dalam penelitian ini adalah sebagai berikut:

a. Menganalisis metode *EigenFace* untuk perhitungan jarak dengan baik sehingga dapat mengenali wajah.
b. Menganalisis tingkat keberhasilan atau akurasi pengenalan wajah dengan menggunakan metode *Eigenface.*

3.1.4 Analisis Metode *EigenFace*

Eigenface adalah sekumpulan unsur wajah yang telah dibuat standar yang diambil dari analisis statistik dari banyak gambar wajah. *Algoritma eigenface* secara keseluruhan cukup sederhana. *Training image* direpresentasikan dalam sebuah *vector*

flat (gabungan *vektor*) dan digabung bersama-sama menjadi sebuah matriks tunggal. *Eigenface* dari masing-masing citra kemudian diekstraksi dan disimpan dalam *file temporary* atau *database. Test image* yang masuk didefinisikan juga nilai *eigenfaces*-nya dan dibandingkan dengan *eigenfaces* dari *image database* atau *file temporary* (Prasetyo dan Rahmatun,2008).

3.1.4.1 Proses *Image Test*

Langkah pertama setelah didapatkan citra wajah hasil capture, citra wajah akan dirubah dari bentuk RGB ke dalam bentuk *grayscale*, setelah didapatkan citra ke abuan, citra akan dirubah menjadi citra hitam putih dengan melakukan *tresshold* agar kompleksitas citra lebih sederhana.

1) *Grayscale*

   Grayscaling citra merupakan tahapan pertama dari proses penyelarasan, pada tahap ini terjadi pengkonversian citra warna RGB menjadi berwarna abu. Citra warna RGB terdiri dari 3 parameter warna yaitu merah (*red*), hijau (*green*) dan biru (*blue*), jika citra warna RGB dimasukan ke dalam proses ekstraksi maka proses tersebut akan sulit untuk itu dilakukan penyamaan parameter dengan melakukan tahap *grayscale*.

2) *Tresshold*

   Tressholding adalah proses pengubah citra berderajat keabuan menjadi citra biner atau hitam putih sehingga dapat diketahui daerah mana yang termasuk

objek dan *background* dari citra secara jelas. Citra hasil *tressholding* biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur.

3.1.4.2 Perhitungan *Eigenvalue* dan *Eigenvektor*

Jika A adalah matriks m x m, maka setiap skalar λ memenuhi persamaan Ax=λx untuk m×1 vektor x ≠ 0, disebut *eigenvalue* dari A. Vektor x disebut *eigenvektor* dari A yang berhubungan dengan *eigenvalue* λ, dan persamaan diatas disebut persamaan *eigenvalue-eigenvektor* A Kadang-kadang *eigenvalue* dan *eigenvektor* juga dinyatakan sebagai (*latents root and vectors*) atau karekteristik *roots* dan vektor. Persamaan dapat juga dituliskan sebagai

(A - λI)x = 0

Setiap nilai *eigenvalue* λ harus memenuhi persamaan determinan,

|A - λI| = 0

yang dikenal sebagai persamaan karakteristik A. Dengan menggunakan definisi suatu determinan, kita bisa mengamati bahwa persamaan karakteristik adalah sebuah polinomial derajat ke-m dalam λ. Karena itu, skalar α0, ..., αm -1 seperti halnya persamaan karakteristik diatas dapat juga dinyatakan sebagai

$$(-\lambda)^m + \alpha_{m-1}(-\lambda)^{m-1} + \cdots + \alpha_1(-\lambda) + \alpha_0 = 0$$

Karena polinomial derajat m memiliki m (*roots*), berarti suatu matriks m×m memiliki m *eigenvalue*, karena itu terdapat m skalar λ1, ..., λm yang memenuhi persamaan

karakteristik. Apabila semua *eigenvalue* A adalah *real*, kadang-kadang kita jumpai *eigenvalue* terbesar ke-I matriks A sebagai λI (A). Dengan kata lain *eigenvalue* A dapat juga dituliskan sebagai

$$\lambda_1(A) \geq \ldots \geq \lambda_m(A).$$

Persamaan karakteristik dapat digunakan untuk mencari *eigenvalue* matriks A. Kemudian dapat juga digunakan dalam persamaan *eigenvalue-eigenvektor* untuk mencari *eigenvektor*. Dari *eigenvektor* yang telah diperoleh, dalam bebarapa penerapan, seperti penguraian nilai singular dan spektral, yang digunakan adalah *eigenvektor* ternormalisasi. *Eigenvektor* ternormalisasi adalah *eigenvektor* dimana tiap-tiap elemen dibagi dengan panjang vektor tersebut. Untuk lebih jelasnya perhatikan contoh berikut :

Tentukan *eigenvalue* dan *eigenvektor* dari matriks A berukuran 3×3 sebagai berikut.

$$A = \begin{bmatrix} 5 & -3 & 3 \\ 4 & -2 & 3 \\ 4 & -4 & 5 \end{bmatrix}$$

Jawaban :

Dengan menggunkan definisi 5.3, persamaan karakteristik A adalah,

$$|A-\lambda I| = \begin{vmatrix} 5-\lambda & -3 & 3 \\ 4 & -2-\lambda & 3 \\ 4 & -4 & 5-\lambda \end{vmatrix}$$

$$= \begin{aligned} &-(5-\lambda)^2(2+\lambda)-3(4)^2-4(3)^2 \\ &+3(4)(2+\lambda)+3(4)(5-\lambda)+3(4)(5-\lambda) \end{aligned}$$

$$= -\lambda^3+8\lambda^2-17\lambda+10$$

$$= -(\lambda-5)(\lambda-2)(\lambda-1)=0$$

Jadi, dari hasil di atas diperoleh tiga *eigenvalue* A, yaitu : λ1=5 , λ2= 2 dan λ3=1 Untuk mendapatkan *eigenvektor* A yang bersesuaian dengan λ1= 5.

(Anton, H., 1987)

3.1.5 *Spesifikasi Hardware*

Spesifikasi minimum *hardware* yang digunakan untuk menjalankan aplikasi dengan baik adalah sebagai berikut :

a. *Prosessor Intel Core* i3 1.80 GHz
b. RAM 4GB
c. *Harddisk* 50GB
d. *Monitor*
e. *Keyboard*
f. *Mouse*
g. *Webcam*

3.1.6 *Spesifikasi Software*

Spesifikasi software yang dibutuhkan agar aplikasi berjalan dengan baik antara lain :

a. Sistem operasi : *Windows* 7

b. *Compiler* dan *script editing software* : *Microsoft Visual Studio* 2010.

3.2 *Planning* (*Estimating, Scheduling, Tracking*)

Dalam tahap ini penelitian memfokuskan pada penjadwalan pengerjaan penelitian, dan analisis kebutuhan perangkat lunak dan perangkat lunak yang digunakan untuk menjalankan aplikasi.

Pada penelitian ini terdapat beberapa proses yang harus dilakukan dari tahap *communication* hingga *implementation and testing*, maka dari itu di perlukan penjadwalan yang tepat agar penelitian ini dapat selesai tepat pada waktunya. Berikut penjadwalan penelitian berdasarkan aktivitas yang dilakukan dengan skala waktu 1 minggu

TABEL : 3.2 Penjadwalan Penelitian

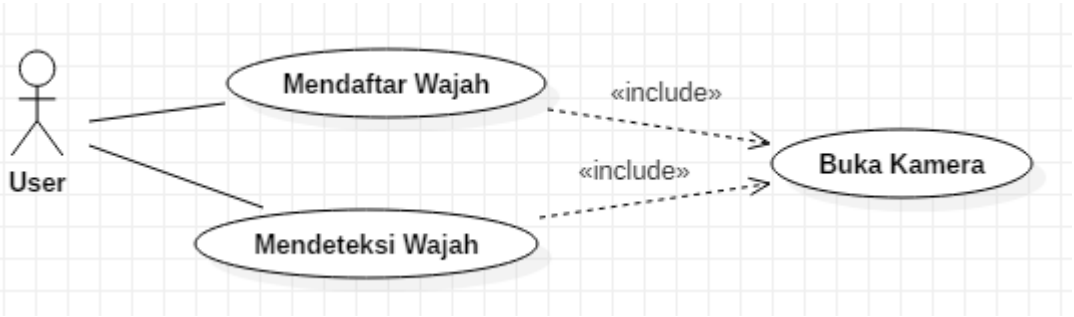
Tahapan	**No**	**Aktifitas**	**Juli**				**Agustus**				**September**			
Communication	1	Pengumpulan Data	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
	2	Studi Literatur												
	3	Analisis Masalah												
	4	Analisis Metode *EigenFace*												
	5	Spesifikasi *Hardware* dan *Software*												
Estimating Scheduling Tracking	1	Penjadwalan												
Perancangan (*Design*)	1	Perancangan UML												
	2	Perancangan *User Interface*												
Construction	1	Pengkodean												
	2	Pengujian(*BlackBox*)												

3.3 Perancangan (*Design*)

Pada tahap ini dilakukan perancangan aplikasi yang dibangun meliputi perancangan diagram dan perancangan antarmuka dari aplikasi yang akan di bangun.

3.3.1 *Use Case* Diagram

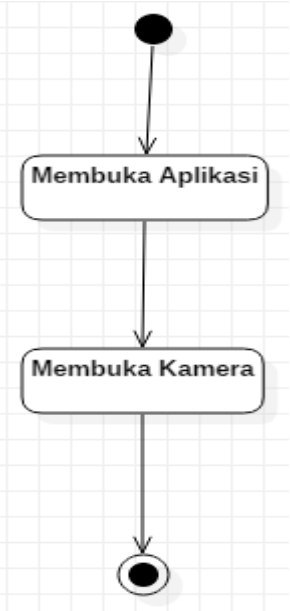
Use Case Mendefinisikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use Case* diagram digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan apa saja yang berhak menggunakan fungsi-fungsi itu. Berikut adalah *Use Case diagram* pada aplikasi yang dirancang digambarkan GAMBAR 3.1.



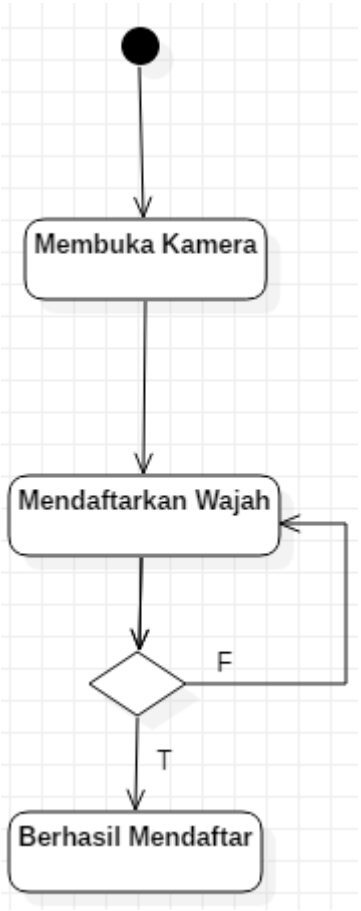
GAMBAR : 3.1 *Use Case* Diagram *Face Recognition*

3.3.2 *Activity* Diagram

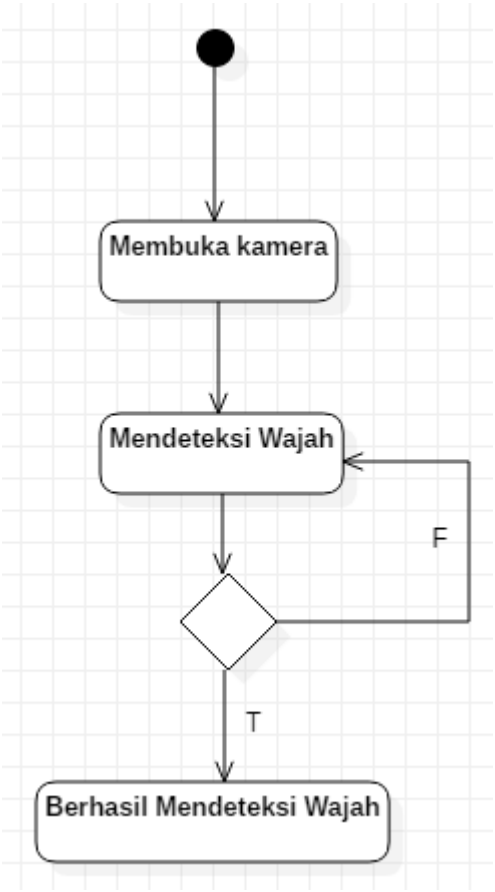
Activity Diagram adalah diagram yang menunjukan aktifitas dari setiap fungsi yang ada, biasanya menggambarkan aliran kerja (*workflow*) dari sebuah sistem atau proses bisnis, dapat juga menggambarkan mengenai aktifitas menu yang ada pada aplikasi. Berikut adalah *Activity diagram* pada aplikasi yang dirancang digambarkan pada GAMBAR 3.2 GAMBAR 3.3, GAMBAR 3.4.



GAMBAR 3.2 *Activity Membuka Kamera*



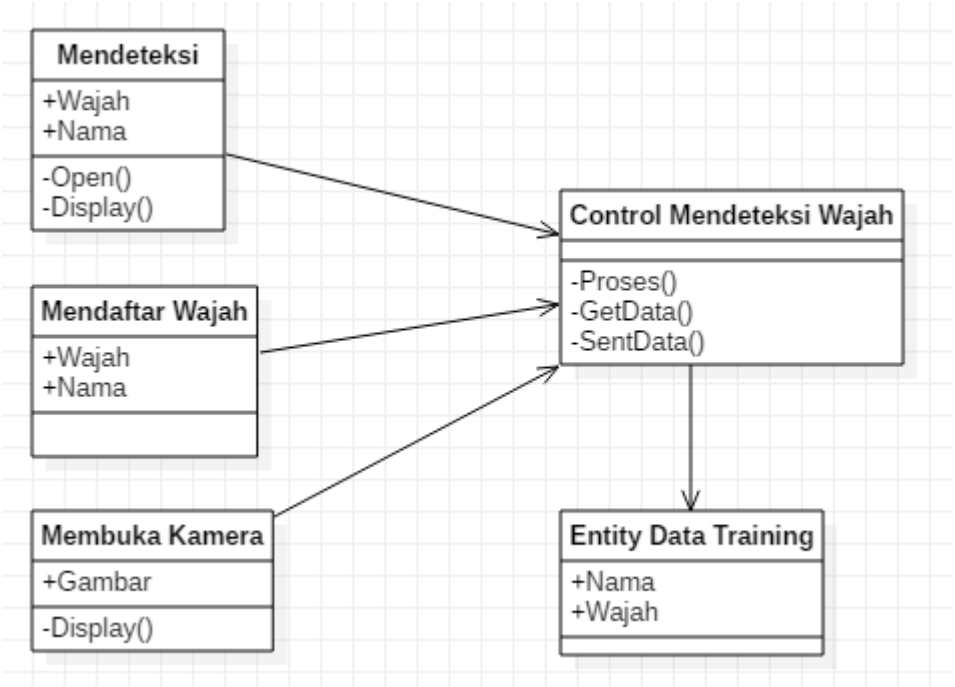
GAMBAR : 3.3 *Activity Mendaftarkan Wajah*



GAMBAR 3.4 *Activity* Pendeteksian Wajah

3.3.3 *Class Diagram*

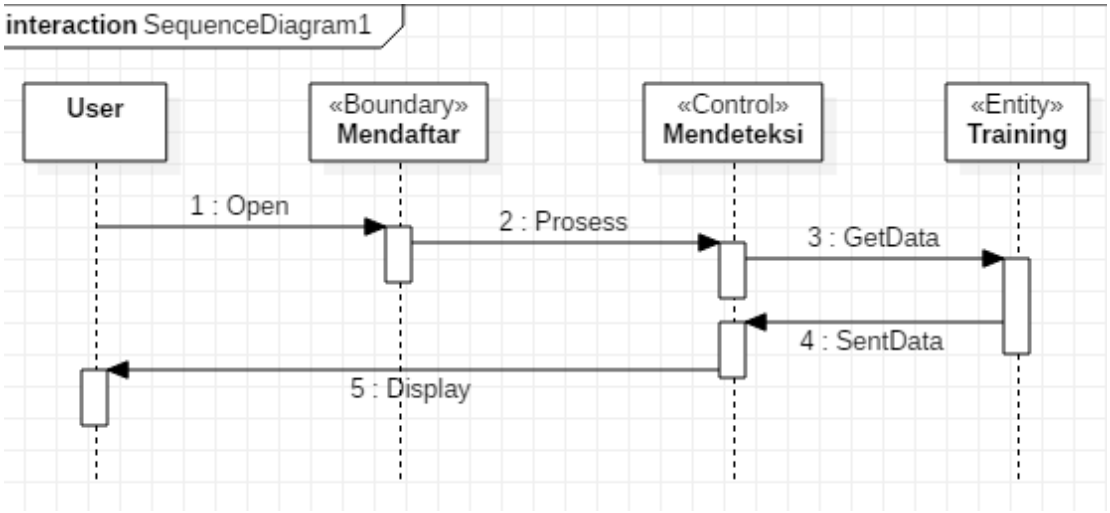
Class Diagram adalah sebuah spesifikasi yang jika di instansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (*atribut / property*) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda / fungsi). *Class Diagram* menggambarkan struktur dan deskrpsi *class*, *package* dan *object* beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi dan lain-lain. *Class Diagram* pada rancangan aplikasi ini digambarkan pada GAMBAR 3.5.



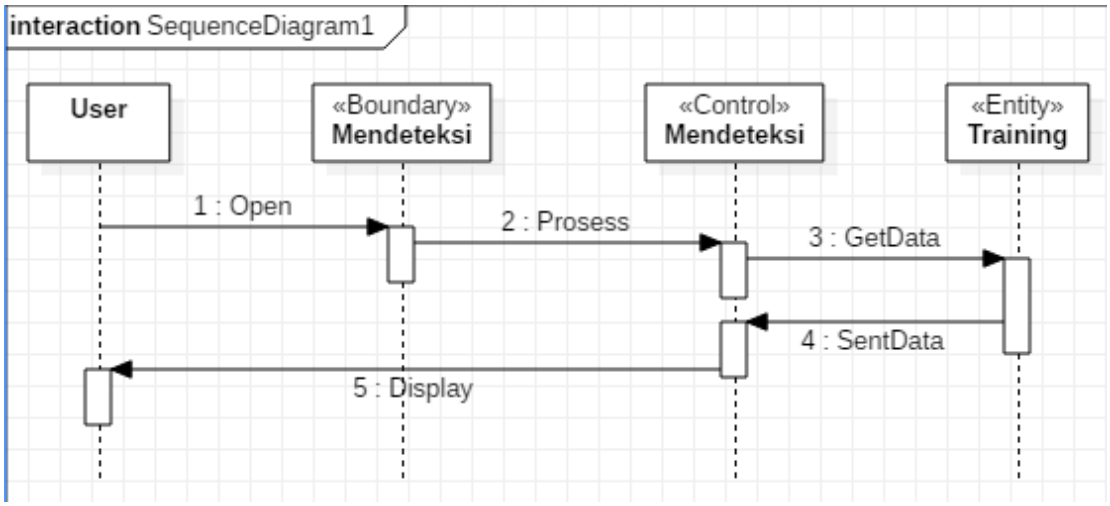
GAMBAR : 3.5 *Class* proses *face Recognition*

3.3.4 *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada setiap objek pada sebuah skenario. Diagram ini menunjukan sejumlah contoh objek dan pesan yang diletakan diantara objek-objek ini didalam use case. Komponen utama sequence diagram terdiri tas objek yang digambarkan dengan kotak bernama. Pesan diwakili oleh garis dengan tanda panah dan waktu yang ditunjukan dengan progress vertical. Berikut adalah *sequence diagram* pada aplikasi yanng dirancang digambarkan pada GAMBAR 3.6, GAMBAR 3.7.



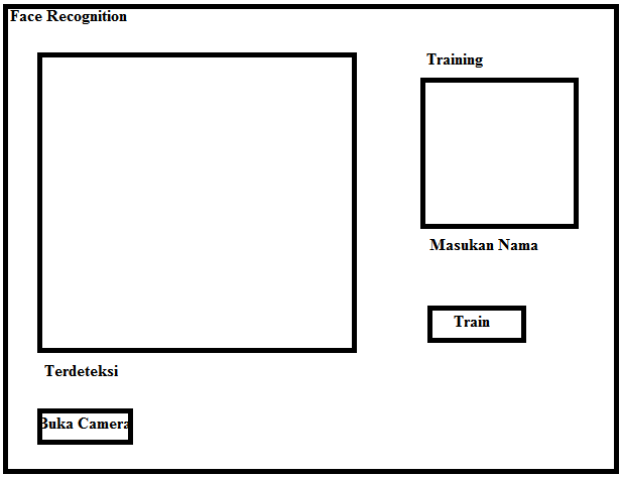
GAMBAR : 3.6 *Sequence Mendaftar*



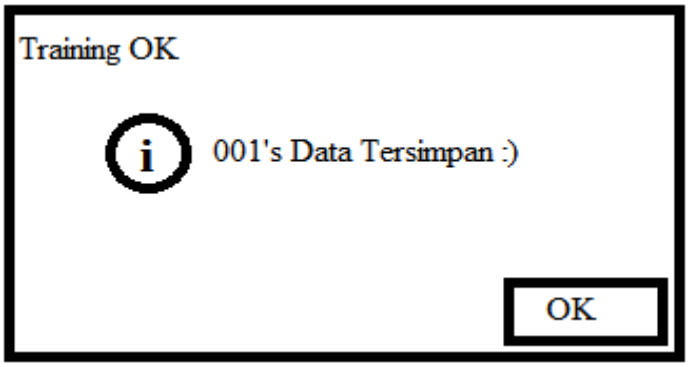
GAMBAR : 3.7 *Sequence Mendeteksi*

3.3.5 Perancangan *User Interface*

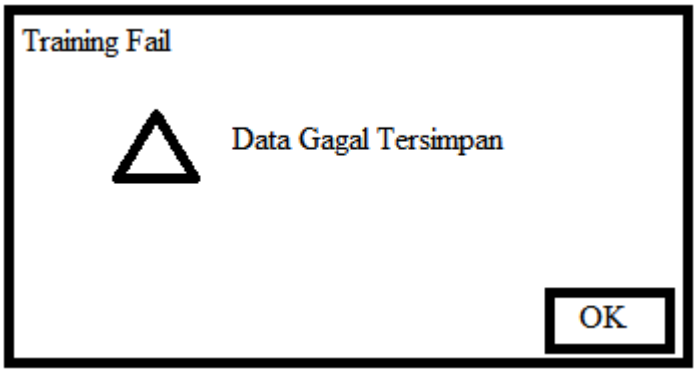
Setelah melakukan analisis diagram pada tahap kali ini akan dilakukan analisis desain interface yang akan dibangun, interface yang akan di bangun terdiri dari halaman *Face Recognition.* Digambarkan pada GAMBAR 3.8, GAMBAR 3..9, dan GAMBAR 3.10



GAMBAR : 3.8 Halaman *Home*



GAMBAR : 3.9 *Notifikasi* Berhasil Menyimpan



GAMBAR : 3.10 *Notifikasi* Gagal Menyimpan

BAB IV

IMPLEMENTASI DAN UJI COBA

4.1 *Construction* (*Code & Test*)

Dalam tahap penelitian ini berfokus pada pengkodean menggunakan bahasa pemrogram *Visusal Studio .NET* dan melakukan pengujian hasil menggunakan metode *black-box testing* setelah pengkodean selesai.

4.1.1 Implementasi

Berikut adalah *interface* aplikasi *Face Recognition* yang di buat pada tahap sebelumnya.

1. Halaman Saat Membuka kamera

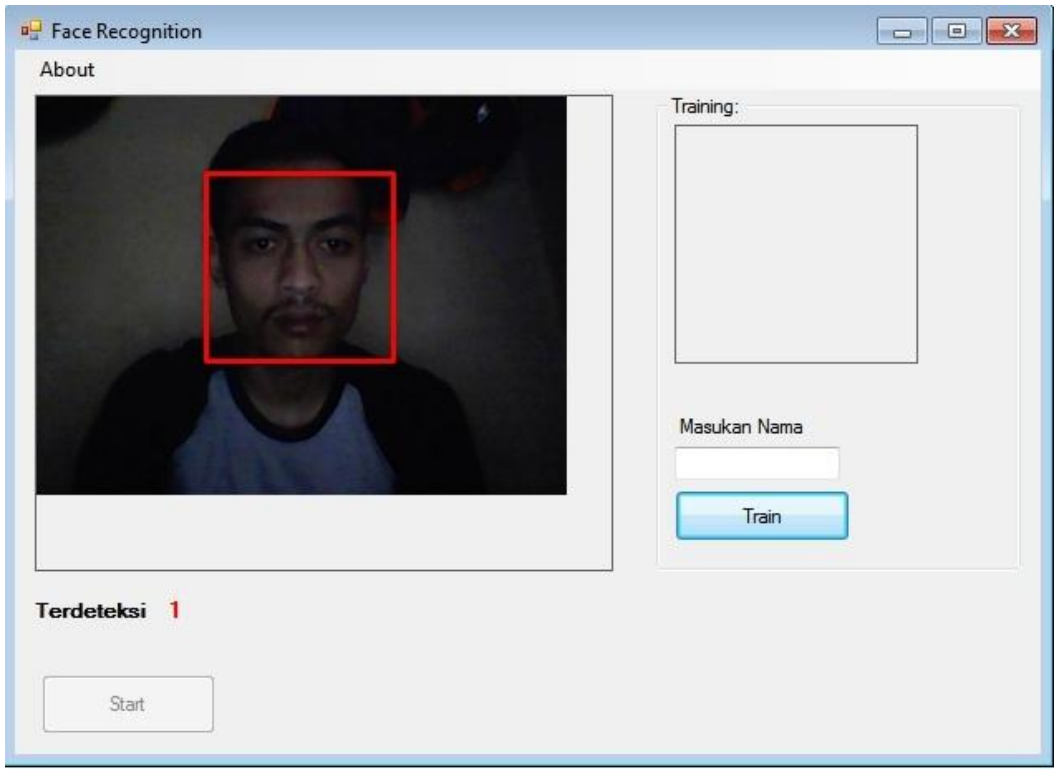
   Pada gambar 4.1 menunjukan bahwa sebelum training dan pendeteksian wajah terlebih dahulu untuk membuka kamera untuk mendapatkan proses training dan pendeteksian wajah.

2. Halaman Saat di *Training*

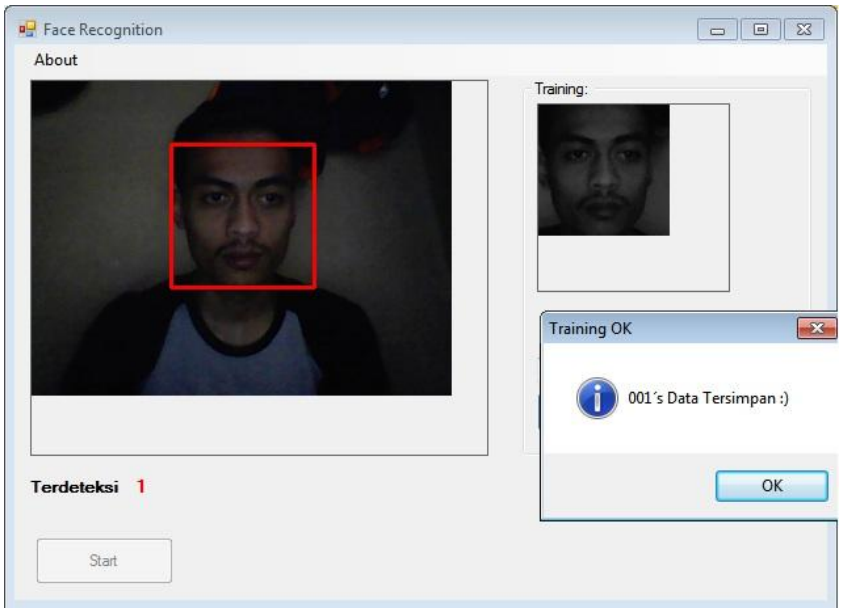
   Pada gambar 4.2 menunjukan hasil mengambil wajah dengan sesuai kotak merah dan menjadi bentuk *grayscale* dan menyimpan ke data hasil *training* ke dalam folder.

3. Halaman Pendeteksian

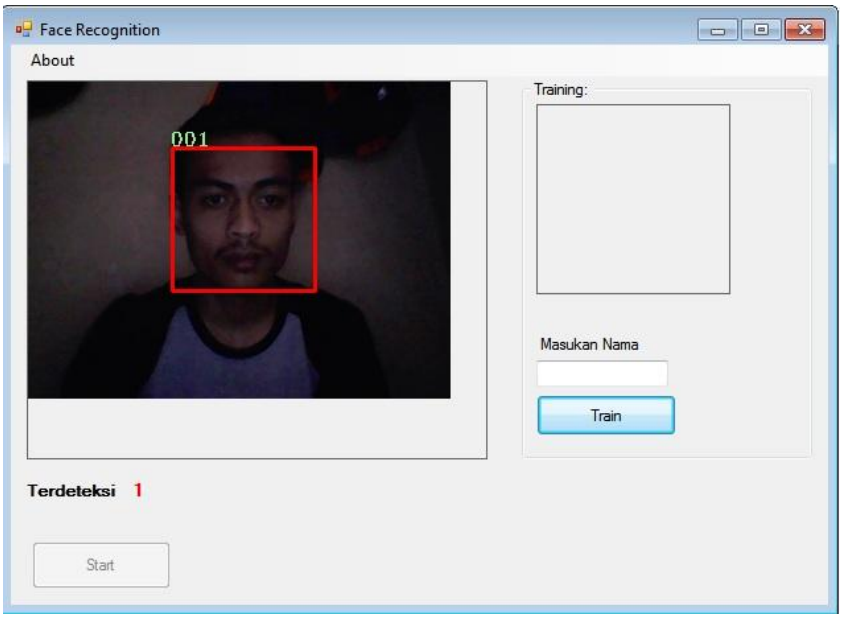
   Pada gambar 4.3 menunjukan pengenalan wajah hasil dari training yang sudah tersimpan dengan nama dan saat pengenalan wajah akan mengidentifikasi/memperifikasi wajah



GAMBAR : 4.1 Halaman Membuka Kamera



GAMBAR : 4.2 Halaman Saat Mendaftar Wajah

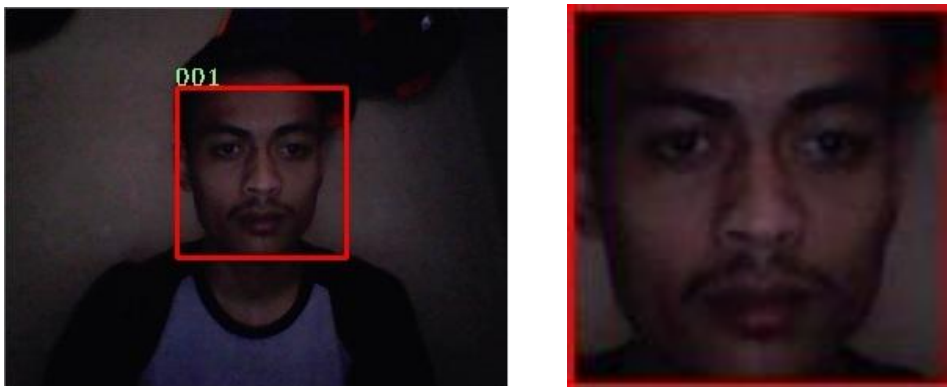


GAMBAR : 4.3 Halaman Mengenali Wajah

4.1.2 Pengujian *Training* wajah

Berikut adalah proses pengujian *training* wajah menggunakan cara, yaitu :

1. Pemotongan gambar



GAMBAR : 4.4 Pengujian Pemotongan Wajah

Pada gambar 4.4 merupakan proses pemotongan wajah hasil training dan pada hasil gambar sebelah kiri terdapat kotak merah untuk pendeteksian wajah sebelum untuk mengenali identitas wajah tersebut akan di proses pemotongan gambar wajah terlebih dahulu agar verifikasi hanya sekitar wajah. Pada gambar sebelah kanan merupakan hasil pemotongan gambar wajah terdapat hasil dari gambar sebelah kiri.

2. Mengubah citra wajah menjadi *grayscale*

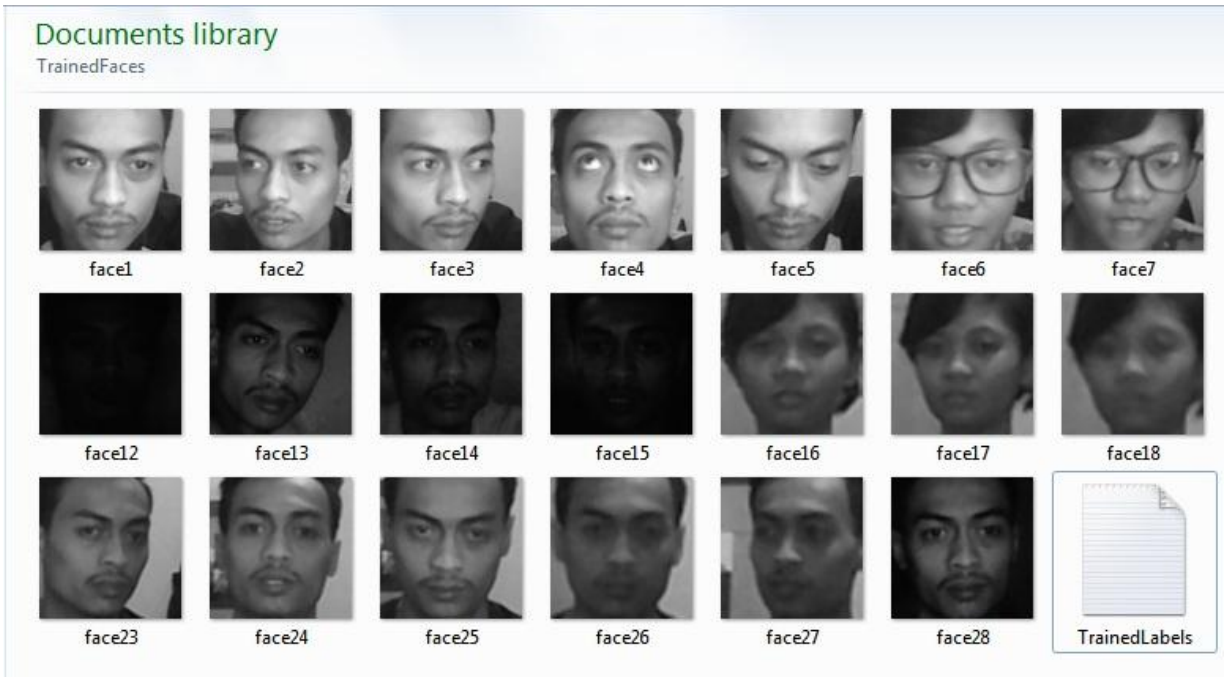
GAMBAR : 4.5 Proses citra asli ke *grayscale*

Pada tahap ini terdapat pada gambar 4.5 merupakan hasil dari citra wajah merubah menjadi *grayscale*

3. Mengubah citra *grayscale* menjadi *tresshold*

GAMBAR : 4.6 Proses merubah *grayscale* ke *tresshold*

4.1.3 Proses Penyimpanan Data



GAMBAR : 4.7 Proses Penyimpanan Data

Pada gambar 4.7 merupakan hasil training dalam tahapan pemotongan gambar wajah, merubah citra wajah menjadi *grayscale* dan merubah *grayscale* ke *tresshold* dan hasil dari tahapan ini disimpan ke dalam folder berupa hasil pada tahapan sebelumnya.

4.1.4 Proses *Eigenface*

Pada tahap ini adalah proses perhitungan *eigenvektor* dan *eigenvalues* dari matriks A berukuran 20 x 20, pada gambar 4.4 dimana pendeteksian hanya pada kotak merah dan kotak merah itu dirubah menjadi matrik dan dapat hasilnya di perlihatkan pada gambar 4.8,matrik A adalah *Eigenvektor* yang yang berisikan nilai piksel..

matriks A

178	182	186	189	193	199	202	203	204	208	210	209	207	210	212	215	216	216	215	215
180	184	188	193	198	202	203	203	205	209	210	210	208	210	212	215	218	219	216	216
183	186	191	197	202	203	204	204	205	209	210	210	209	210	210	212	215	217	215	212
184	187	192	198	202	204	204	203	206	209	210	209	209	210	210	209	213	215	212	208
183	188	194	201	204	203	203	203	206	209	209	209	210	210	211	211	213	212	209	208
179	187	193	200	204	203	202	203	204	205	205	206	209	209	211	212	211	207	207	210
168	178	187	193	200	200	200	202	202	202	202	205	207	209	209	209	209	207	205	208
146	155	170	182	192	196	198	197	198	200	200	204	205	206	205	206	208	207	205	205
114	130	147	162	175	185	192	191	194	197	197	199	201	205	204	204	205	205	204	204
71	84	96	111	127	142	166	184	189	189	190	193	198	200	202	201	201	199	199	200
69	75	78	80	85	93	111	132	163	174	177	173	181	189	193	197	196	195	191	189
62	66	68	69	68	69	61	72	93	124	151	164	172	174	183	189	194	193	191	188
94	75	67	70	69	74	74	81	87	82	111	134	157	164	173	178	183	183	183	183
147	131	107	89	79	77	82	88	95	94	103	120	147	162	169	172	175	173	172	173
162	153	142	124	111	95	85	88	94	96	100	111	129	152	165	167	167	167	167	168
159	143	133	130	124	117	107	99	96	95	99	107	122	139	154	166	166	166	166	167
150	146	134	129	123	117	119	118	110	100	99	101	116	127	143	161	167	168	168	168
90	114	129	138	135	123	120	121	117	109	100	100	108	122	136	151	163	167	167	168
58	123	117	101	125	131	132	130	128	124	107	98	99	112	128	148	162	165	168	177
81	192	194	123	106	116	133	143	145	138	119	101	97	108	128	155	170	175	181	190

GAMBAR : 4.8 Matriks *Eigenvektor*

matriks I																			
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

GAMBAR : 4.9 *Matriks Eigenvalues*

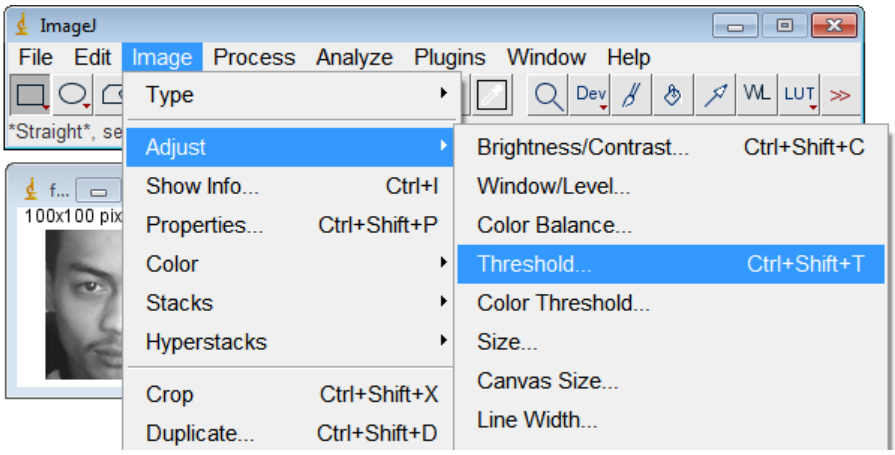
matrixA-lambda*I																			
178	182	186	189	193	199	202	203	204	208	210	209	207	210	212	215	216	216	215	215
180	184	188	193	198	202	203	203	205	209	210	210	208	210	212	215	218	219	216	216
183	186	191	197	202	203	204	204	205	209	210	210	209	210	210	212	215	217	215	212
184	187	192	198	202	204	204	203	206	209	210	209	209	210	210	209	213	215	212	208
183	188	194	201	204	203	203	203	206	209	209	209	210	210	211	211	213	212	209	208
179	187	193	200	204	203	202	203	204	205	205	206	209	209	211	212	211	207	207	210
168	178	187	193	200	200	200	202	202	202	202	205	207	209	209	209	209	207	205	208
146	155	170	182	192	196	198	197	198	200	200	204	205	206	205	206	208	207	205	205
114	130	147	162	175	185	192	191	194	197	197	199	201	205	204	204	205	205	204	204
71	84	96	111	127	142	166	184	189	189	190	193	198	200	202	201	201	199	199	200
69	75	78	80	85	93	111	132	163	174	177	173	181	189	193	197	196	195	191	189
62	66	68	69	68	69	61	72	93	124	151	164	172	174	183	189	194	193	191	188
94	75	67	70	69	74	74	81	87	82	111	134	157	164	173	178	183	183	183	183
147	131	107	89	79	77	82	88	95	94	103	120	147	162	169	172	175	173	172	173
162	153	142	124	111	95	85	88	94	96	100	111	129	152	165	167	167	167	167	168
159	143	133	130	124	117	107	99	96	95	99	107	122	139	154	166	166	166	166	167
150	146	134	129	123	117	119	118	110	100	99	101	116	127	143	161	167	168	168	168
90	114	129	138	135	123	120	121	117	109	100	100	108	122	136	151	163	167	167	168
58	123	117	101	125	131	132	130	128	124	107	98	99	112	128	148	162	165	168	177
81	192	194	123	106	116	133	143	145	138	119	101	97	108	128	155	170	175	181	190

GAMBAR : 4.10 Hasil Perhitungan Matriks

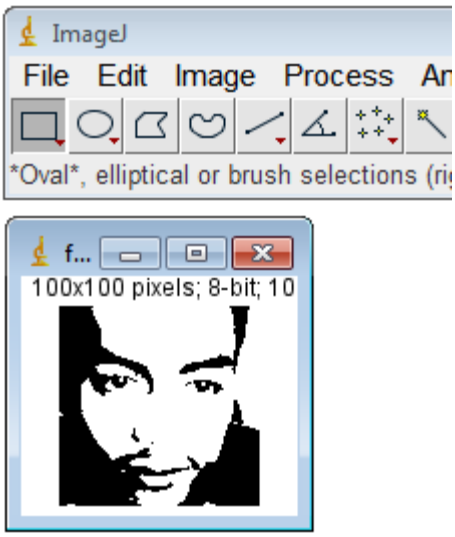
4.1.5 Proses *ImageJ*

Proses ini menjelaskan tahapan *imagej* yang menghasilkan keakurasian pada tabel 4.1 berikut adalah tahapan proses imagej yaitu :

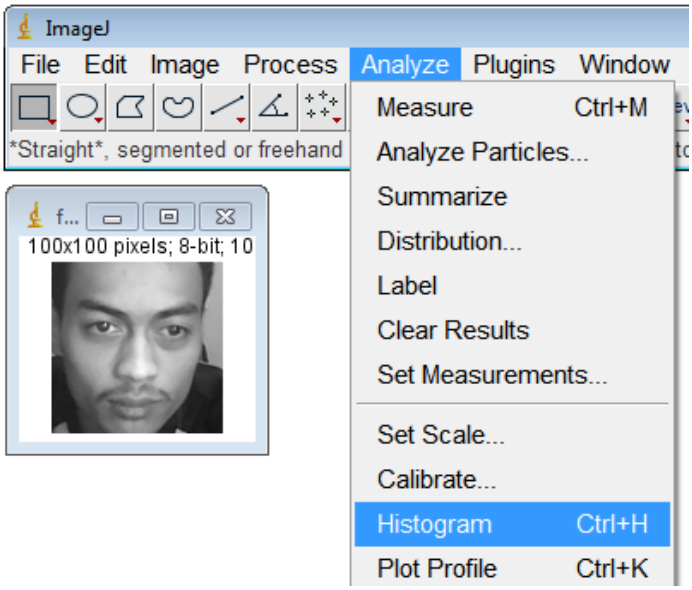
1. Pada gambar 4.11 dan 4.12 adalah proses yang digunakan untuk mengkonversi nilai *pixel* citra *grayscale* menjadi biner pada metode *threshold.*
2. Pada gambar 4.13 dan 4.14 adalah proses grafis untuk distribusi warna dari citra digital atau menggambarkan penyebaran nilai-nilai intensitas *pixel* dari suatu citra atau bagian tertentu di dalam citra. Dari sebuah *histogram* dapat diketahui *frekuensi* kemunculan *relative* dari *intensitas* pada citra, kecerahan, dan kontas dari sebuah gambar.
3. Pada gambar 4.15 dan 4.16 adalah fungsi untuk menghitung data rata-rata dari matrik.



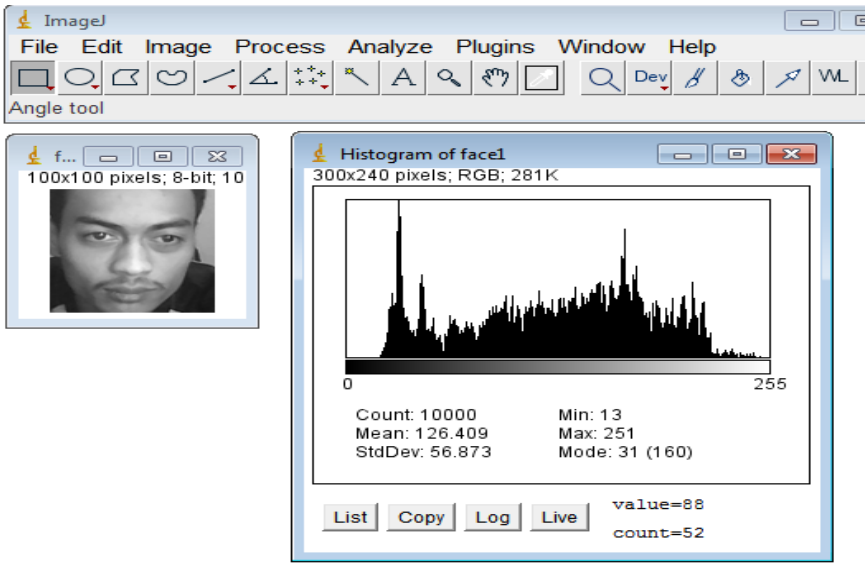
GAMBAR : 4.11 Proses citra *grayscale* ke *Threshold*



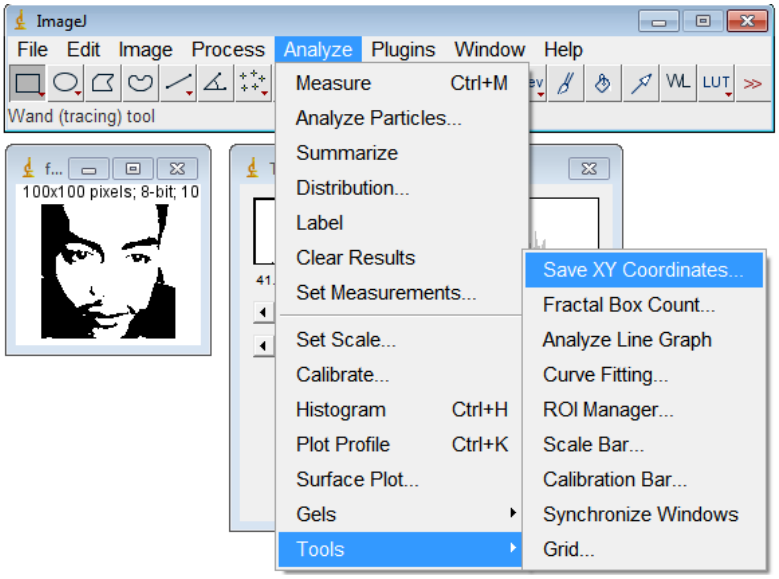
GAMBAR : 4.12 Hasil Proses citra *grayscale* ke *Threshold.*



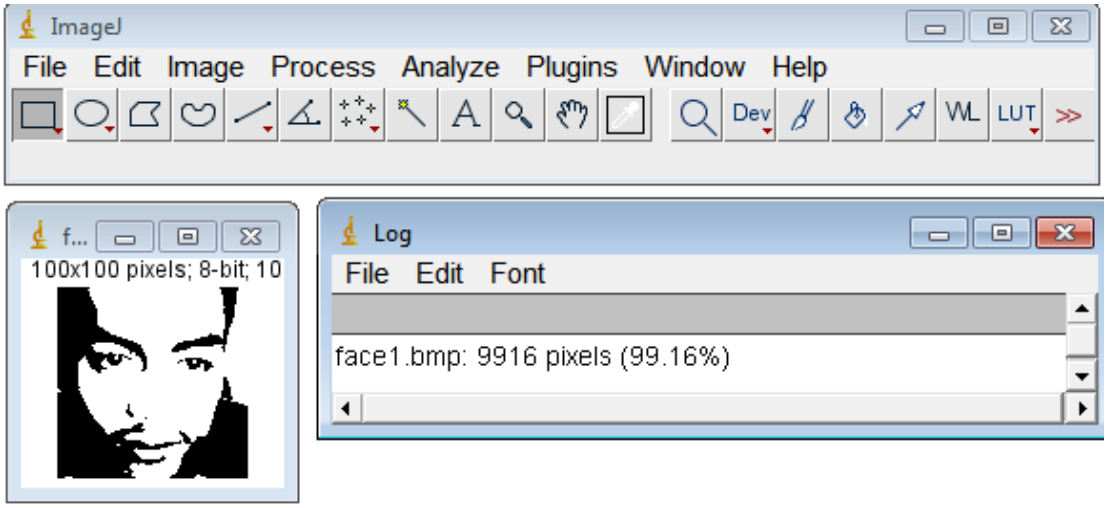
GAMBAR : 4.13 proses citra *grayscale* ke *Histogram*



GAMBAR : 4.14 Hasil Proses citra *grauscale* ke *Histogram*



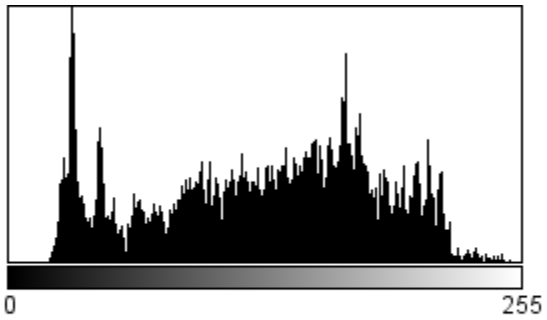
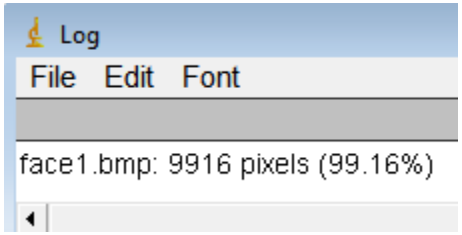
GAMBAR : 4.15 Proses Citra untuk Mengetahui *Coordinates*

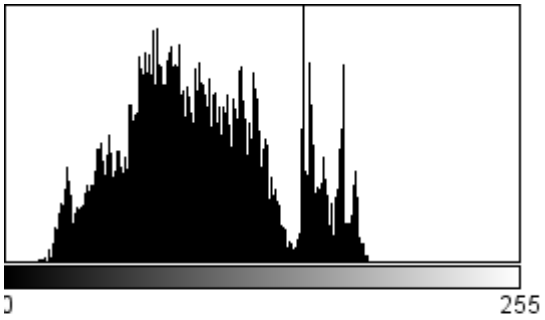
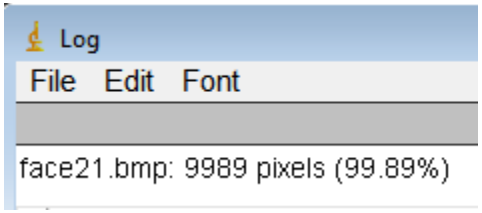
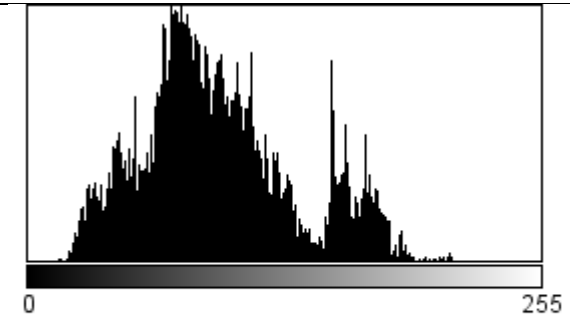
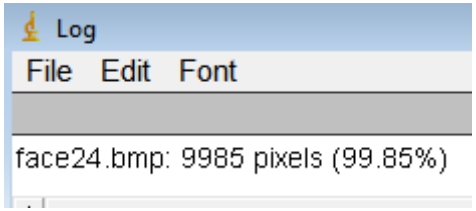


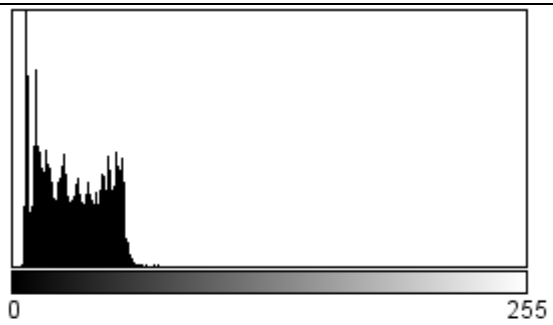
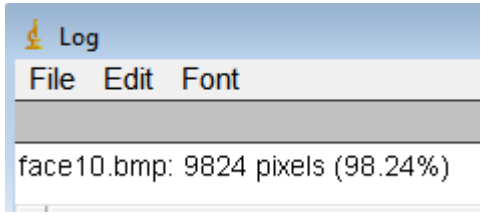
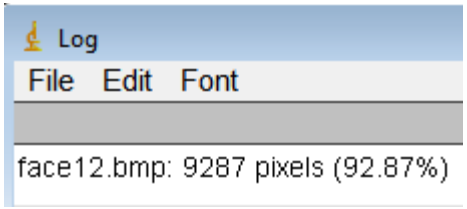
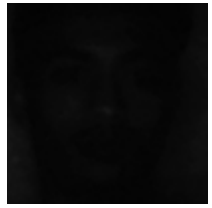
GAMBAR : 4.16 Hasil untuk Mengetehui *Coordinates*

Berikut adalah pengujian menggunakan *black-box* pada tahap *Verification,* Tabel 4.1 Rencana Pengujian

TABEL : 4.1. Pengujian Dengan Metode *Black-Box*

NO	Jarak	*Brightnes*	Keakurasian	Hasil *Training*
1.	100 CM			

NO	Jarak	*Brightnes*	Keakurasian	Hasil *Training*
2.	150 CM			
3.	200 CM			

NO	Jarak	*Brightnes*	Keakurasian	Hasil *Training*
4.	100 CM			
5.	150CM			

Setelah dilakukan pengujian dengan menggunakan metode *black box* didapatkan hasil dari tabel, 4.1 menunjukan hasil pengujian dengan metode *black box*. Dalam jarak yang di uji dengan 100 cm, 150, cm dan 200 cm, ditambah dengan rotasi 15 derajat yang berbeda mendapatkan keakurasian sebesar 99.85 % dan terkecil dengan 92.87 % .

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil perancangan sampai dengan pengujian didapatkan kesimpulan bahwa :

1. Jarak antara wajah dengan *webcam* sangat berpengaruh dalam proses pendeteksian wajah.
2. Tinggi rendahnya unsur pencahayaan yang berada di sekitar objek sangat mempengaruhi proses pendeteksian wajah.
3. Kombinasi- kombinasi dari metode *eigenface* dapat mendeteksi wajah dengan keakurasian sebesar 99.85%

5.2 Saran

Adapun saran-saran yang dapat disampaikan untuk pengembangan aplikasi lebih lanjut :

1. Untuk penelitian lebih lanjut, ciri menggunakan metode *eigenface* dapat digunakan untuk membuat informasi seperti sistem absensi, sistem keamanan rumah, sistem identifikasi seseorang dan lain-lain.

2. Untuk memudahkan pengguna sistem, citra wajah bisa terdapat obyek-obyek lain terdapat di dalam citra. Sehingga citra wajah yang diinputkan oleh pengguna dapat lebih sedikit mempunyai batasan.

DAFTAR PUSTAKA

Abdul Aziz Abdillah. (2014) : Uji Kinerja *Face Recognition* menggunakan *EigenFace*, Jakarta.

Anton, H. (1987) : Elementary Linear Algebra, John Wiley & Son, New York

Candra Noor Santi. (2011) : Mengubah Citra Berwarna Menjadi GrayScale dan Citra Biner, Semarang.

Dian Esti Pratiwi. (2013) : Implementasi Pengenalan Wajah Menggunakan PCA, Jurnal Ilmu Komputer dan Elektronika, Yogyakarta.

Fatta, H. A. (2009): Rekayasa Sistem Pengenalan Wajah.Yogyakarta: Andi Offset

Fathansyah.(1999) : Basis Data. Informatika Bandung, Bandung.

Jogiyanto, (1999) : Analisis dan Desain Sistem Informasi. Yogyakarta : Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis, ANDI Yogyakarta.

Minartiningtyas, Brigida Arie. (2013). *Teori Pengenalan Wajah (Face Recognition).*

Pavani, S.-K., Delgadoa, D., & Frangia, A. F, (2010) : Haar-like Features with Optimally Weighted Rectangles for Rapid Object Detection. The Journal of the Pattern Recognition Society.

Panji Purwanto.(2015) : Burhanudin Dirgantoro Ir.,M.T “Implementasi Face Identification dan Face Recognition Pada Kamera Pengawas Sebagai Pendeteksi Bahaya

Prasetyo, E., & Rahmatun, I. (2008): Desain System Pengenalan Wajah Dengan Variasi Ekspresi dan Posisi Menggunakan Metode EIGENFACE. Jurnal Ilmiah Informatika dan Komputer, Vol.11, No.01, pp.33.

Pressman, R.S. (2015) : Rekayasa Perangkat Lunak Pendekatan Praktisi Buku I. Yogyakarta : Andi.

Purwanto, Dirgantoro, & Jati, (2015) : Implementasi Face Identification Dan Face Recognition Pada Kamera Pengawas Sebagai Pendeteksi Bahaya. Universitas Telkom.

Rosa A.S dan Shalahuddin. M. (2018) : Rekayasa Perangkat Lunak Terstuktur dan Berorientasi Objek. Bandung : INFORMATIKA.

Sayeed Al-Aidid dan Daniel S. Pamungkas. (2018) : Sistem Pengenalan Wajah dengan Algoritma Haar Cascade dan Local Binary Pattern Histogram, Batam

Sugiyono, (2007): Metodologi Penelitian Bisnis, PT. Gramedia, Jakarta

Sutarman, (2012) : *Pengantar Teknologi Informasi*. Jakarta: PT. Bumi Aksara.

A.*Listing* Program

```
Imports System.Diagnostics
Imports Emgu.CV.Structure
Imports Emgu.CV
Imports Emgu.CV.CvEnum
Imports System.Collections.Generic
Imports System.Drawing
Imports System.Windows.Forms
Imports System.IO

Public Class Form1
   Dim currentFrame As Image(Of Bgr, [Byte])
   Dim grabber As Capture
   Dim face As HaarCascade
   Dim eye As HaarCascade
   Dim font As New MCvFont(CvEnum.FONT.CV_FONT_HERSHEY_TRIPLEX,
0.5, 0.5)
   Dim result As Image(Of Gray, Byte), TrainedFace As Image(Of Gray, Byte) =
Nothing
   Dim gray As Image(Of Gray, Byte) = Nothing
   Dim trainingImages As New List(Of Image(Of Gray, Byte))()
   Dim labels As New List(Of String)()
   Dim NamePersons As New List(Of String)()
   Dim ContTrain As Integer, NumLabels As Integer, t As Integer
   Dim name As String, names As String = Nothing

   <Serializable()> _
   Public Class EigenObjectRecognizer
      Private _eigenImages As Image(Of Gray, [Single])()
      Private _avgImage As Image(Of Gray, [Single])
      Private _eigenValues As Matrix(Of Single)()
      Private _labels As String()
      Private _eigenDistanceThreshold As Double

      Public Property EigenImages() As Image(Of Gray, [Single])()
         Get
            Return _eigenImages
         End Get
         Set(ByVal value As Image(Of Gray, [Single])())
            _eigenImages = value
         End Set
      End Property
```

```
    Public Property Labels() As [String]()
        Get
            Return _labels
        End Get
        Set(ByVal value As [String]())
            _labels = value
        End Set
    End Property
    Public Property EigenDistanceThreshold() As Double
        Get
            Return _eigenDistanceThreshold
        End Get
        Set(ByVal value As Double)
            _eigenDistanceThreshold = value
        End Set
    End Property
    Public Property AverageImage() As Image(Of Gray, [Single])
        Get
            Return _avgImage
        End Get
        Set(ByVal value As Image(Of Gray, [Single]))
            _avgImage = value
        End Set
    End Property
    Public Property EigenValues() As Matrix(Of Single)()
        Get
            Return _eigenValues
        End Get
        Set(ByVal value As Matrix(Of Single)())
            _eigenValues = value
        End Set
    End Property

    Private Sub New()
    End Sub
    Public Sub New(ByVal images As Image(Of Gray, [Byte])(), ByRef termCrit As
MCvTermCriteria)
        Me.New(images, GenerateLabels(images.Length), termCrit)
    End Sub

    Private Shared Function GenerateLabels(ByVal size As Integer) As [String]()
        Dim labels As [String]() = New String(size - 1) {}
        For i As Integer = 0 To size - 1
```

```
            labels(i) = i.ToString()
        Next
        Return labels
    End Function
    Public Sub New(ByVal images As Image(Of Gray, [Byte])(), ByVal labels As
[String](), ByRef termCrit As MCvTermCriteria)
        Me.New(images, labels, 0, termCrit)
    End Sub

    Public Sub New(ByVal images As Image(Of Gray, [Byte])(), ByVal labels As
[String](), ByVal eigenDistanceThreshold As Double, ByRef termCrit As
MCvTermCriteria)
        Debug.Assert(images.Length = labels.Length, "The number of images should
equals the number of labels")
        Debug.Assert(eigenDistanceThreshold >= 0.0, "Eigen-distance threshold
should always >= 0.0")

        CalcEigenObjects(images, termCrit, _eigenImages, _avgImage)

        _eigenValues = Array.ConvertAll(Of Image(Of Gray, [Byte]), Matrix(Of
Single))(images, Function(img As Image(Of Gray, [Byte])) New Matrix(Of
Single)(EigenDecomposite(img, _eigenImages, _avgImage)))

        _labels = labels

        _eigenDistanceThreshold = eigenDistanceThreshold
    End Sub

#Region "static methods"

    Public Shared Sub CalcEigenObjects(ByVal trainingImages As Image(Of Gray,
[Byte])(), ByRef termCrit As MCvTermCriteria, ByRef eigenImages As Image(Of
Gray, [Single])(), ByRef avg As Image(Of Gray, [Single]))
        Dim width As Integer = trainingImages(0).Width
        Dim height As Integer = trainingImages(0).Height

        Dim inObjs As IntPtr() = Array.ConvertAll(Of Image(Of Gray, [Byte]),
IntPtr)(trainingImages, Function(img As Image(Of Gray, [Byte])) img.Ptr)

        If termCrit.max_iter <= 0 OrElse termCrit.max_iter > trainingImages.Length
Then
            termCrit.max_iter = trainingImages.Length
        End If
```

```
        Dim maxEigenObjs As Integer = termCrit.max_iter
        eigenImages = New Image(Of Gray, Single)(maxEigenObjs - 1) {}
        For i As Integer = 0 To eigenImages.Length - 1
            eigenImages(i) = New Image(Of Gray, Single)(width, height)
        Next
        Dim eigObjs As IntPtr() = Array.ConvertAll(Of Image(Of Gray, [Single]),
IntPtr)(eigenImages, Function(img As Image(Of Gray, [Single])) img.Ptr)
        avg = New Image(Of Gray, [Single])(width, height)

        CvInvoke.cvCalcEigenObjects(inObjs, termCrit, eigObjs, Nothing, avg.Ptr)
    End Sub

    Public Shared Function EigenDecomposite(ByVal src As Image(Of Gray,
[Byte]), ByVal eigenImages As Image(Of Gray, [Single])(), ByVal avg As Image(Of
Gray, [Single])) As Single()
        Return CvInvoke.cvEigenDecomposite(src.Ptr, Array.ConvertAll(Of
Image(Of Gray, [Single]), IntPtr)(eigenImages, Function(img As Image(Of Gray,
[Single])) img.Ptr), avg.Ptr)
    End Function
#End Region

    Public Function EigenProjection(ByVal eigenValue As Single()) As Image(Of
Gray, [Byte])
        Dim res As Image(Of Gray, [Byte]) = New Image(Of Gray,
Byte)(_avgImage.Width, _avgImage.Height)
        CvInvoke.cvEigenProjection(Array.ConvertAll(Of Image(Of Gray, [Single]),
IntPtr)(_eigenImages, Function(img As Image(Of Gray, [Single])) img.Ptr),
eigenValue, _avgImage.Ptr, res.Ptr)
        Return res
    End Function

    Public Function GetEigenDistances(ByVal image As Image(Of Gray, [Byte]))
As Single()
        Using eigenValue As New Matrix(Of Single)(EigenDecomposite(image,
_eigenImages, _avgImage))
            Return Array.ConvertAll(Of Matrix(Of Single), Single)(_eigenValues,
Function(eigenValueI As Matrix(Of Single))
CSng(CvInvoke.cvNorm(eigenValue.Ptr, eigenValueI.Ptr,
Emgu.CV.CvEnum.NORM_TYPE.CV_L2, IntPtr.Zero)))
        End Using
    End Function
```

```
        Public Sub FindMostSimilarObject(ByVal image As Image(Of Gray, [Byte]),
ByRef index As Integer, ByRef eigenDistance As Single, ByRef label As [String])
            Dim dist As Single() = GetEigenDistances(image)

            index = 0
            eigenDistance = dist(0)
            For i As Integer = 1 To dist.Length - 1
               If dist(i) < eigenDistance Then
                  index = i
                  eigenDistance = dist(i)
               End If
            Next
            label = Labels(index)
        End Sub

        Public Function Recognize(ByVal image As Image(Of Gray, [Byte])) As
[String]
            Dim index As Integer
            Dim eigenDistance As Single
            Dim label As [String]
            FindMostSimilarObject(image, index, eigenDistance, label)

            Return If((_eigenDistanceThreshold <= 0 OrElse eigenDistance <
_eigenDistanceThreshold), _labels(index), [String].Empty)
        End Function
    End Class

    Public Sub New()
        InitializeComponent()
        face = New HaarCascade("haarcascade_frontalface_default.xml")
        Try
            Dim Labelsinfo As String = File.ReadAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt")
            Dim Labels__1 As String() = Labelsinfo.Split("%"c)
            NumLabels = Convert.ToInt16(Labels__1(0))
            ContTrain = NumLabels
            Dim LoadFaces As String

            For tf As Integer = 1 To NumLabels
               LoadFaces = "face" & tf & ".bmp"
               trainingImages.Add(New Image(Of Gray, Byte)(Application.StartupPath +
"/TrainedFaces/" & LoadFaces))
               labels.Add(Labels__1(tf))
```

```
            Next
        Catch e As Exception
            MessageBox.Show("Nothing in database, please add at least a face.",
"Triained faces load", MessageBoxButtons.OK, MessageBoxIcon.Information)

        End Try
    End Sub

    Private Sub button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button1.Click
        grabber = New Capture()
        grabber.QueryFrame()
        Timer1.Start()
        button1.Enabled = False
    End Sub

    Private Sub button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles button2.Click
        Try
            ContTrain = ContTrain + 1
            gray = grabber.QueryGrayFrame().Resize(320, 240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC)
            Dim facesDetected As MCvAvgComp()() = gray.DetectHaarCascade(face,
1.2, 10, Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
New Size(40, 40))
            For Each f As MCvAvgComp In facesDetected(0)
                TrainedFace = currentFrame.Copy(f.rect).Convert(Of Gray, Byte)()
                Exit For
            Next
            TrainedFace = result.Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC)
            trainingImages.Add(TrainedFace)
            labels.Add(TextBox1.Text)
            imageBox1.Image = TrainedFace
            File.WriteAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString() &
"%")
            For i As Integer = 1 To trainingImages.ToArray().Length
                trainingImages.ToArray()(i - 1).Save(Application.StartupPath +
"/TrainedFaces/face" & i & ".bmp")
                File.AppendAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", labels.ToArray()(i - 1) + "%")
```

```
        Next

        MessageBox.Show(TextBox1.Text + "´s Data Tersimpan :)", "Training OK",
MessageBoxButtons.OK, MessageBoxIcon.Information)
        TextBox1.Text = ""
        imageBox1.Image = Nothing
      Catch
        MessageBox.Show("Data Gagal Tersimpan", "Training Fail",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
      End Try

   End Sub

   Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
      label3.Text = "0"

      NamePersons.Add("")
      currentFrame = grabber.QueryFrame().Resize(320, 240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC)
      gray = currentFrame.Convert(Of Gray, [Byte])()

      Dim facesDetected As MCvAvgComp()() = gray.DetectHaarCascade(face, 1.2,
10, Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
New Size(20, 20))
      For Each f As MCvAvgComp In facesDetected(0)
        t = t + 1
        result = currentFrame.Copy(f.rect).Convert(Of Gray, Byte)().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC)
        currentFrame.Draw(f.rect, New Bgr(Color.Red), 2)

        If trainingImages.ToArray().Length <> 0 Then
          Dim termCrit As New MCvTermCriteria(ContTrain, 0.001)
          Dim recognizer As New
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 5000, termCrit)

          name = recognizer.Recognize(result)

          currentFrame.Draw(name, font, New Point(f.rect.X - 2, f.rect.Y - 2),
color:=New Bgr(Color.LightGreen))
        End If

        NamePersons(t - 1) = name
```

```
            NamePersons.Add("")

            label3.Text = facesDetected(0).Length.ToString()
        Next
        t = 0

        For nnn As Integer = 0 To facesDetected(0).Length - 1
            names = names + NamePersons(nnn) + ", "
        Next
        imageBoxFrameGrabber.Image = currentFrame
        NamePersons.Clear()
    End Sub
End Class
```