BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional menggambarkan keadaan sistem yang akan di bangun. Diantaranya perangkat keras, perangkat lunak, sebagai bahan analisis kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan.

3.1.1 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak merupakan faktor-faktor yang harus dipenuhi untuk merancang sebuah perangkat lunak sehingga perangkat lunak tersebut sesuai dengan maksud dan tujuan perangkat lunak perangkat lunak tersebut dibuat. perangkat lunak yang dibutuhkan antara lain :

1. Miscrosoft Visual C++ 6.0
2. Open CV 3.1
3. Power Designer
4. Windows platform SDK
5. Directx SDK

3.1.1.1 Kebutuhan Antarmuka Eksternal

Perangkat lunak yang dibangun membutuhkan perangkat lunak lain sebagai penunjang agar aplikasi yang dibangun berjalan dengan fungsinya. Kebutuhan tersebut antara lain :

1. Antarmuka Komunikasi

Sebagai penunjang antarmuka komunikasi dari perangkat lunak, diperlukan *Framework.Net* 2.0 (teknologi inti yang menyediakan berbagai *library*) untuk digunakan oleh aplikasi di atasnya yang mendukung aplikasi, atau yang lebih tinggi lagi.

3.1.2 Kebutuhan Perangkat Keras

Analisis Perangkat keras merupakan proses analisis yang lebih menekankan kepada aspek pemanfaatan perangkat keras yang selama ini telah dimiliki serta perangkat-perangkat apa saja yang dibutuhkan untuk membuat sistem ini berjalan. Perangkat keras yang dibutuhkan sebagai berikut:

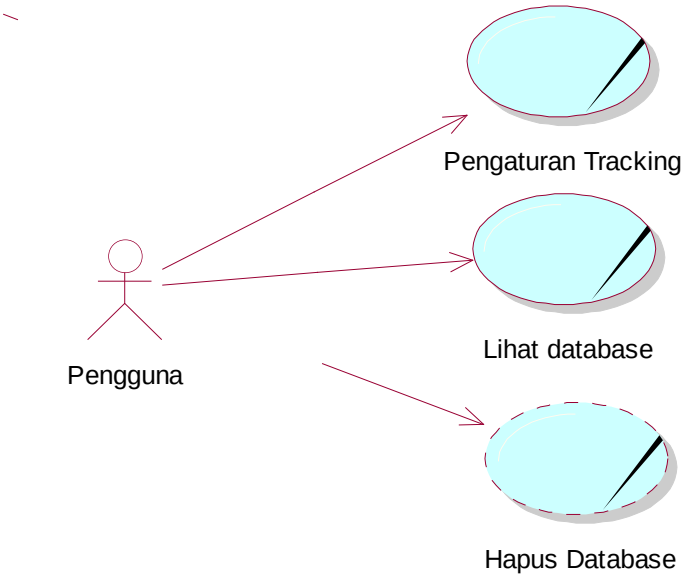
1. Monitor
2. Processor Pentium 4 2,8 Ghz
3. Harddisk 80 GB
4. RAM 512 GB
5. DVD - ROM
6. Mouse dan keyboard

3.2 Analisis Kebutuhan Fungsional

Merupakan kebutuhan secara fungsional yang harus dipenuhi oleh perangkat lunak yang dibangun. Kebutuhan fungsional tersebut akan dideskripsikan, sebagai berikut :

3.2.1 Use Case Diagram

Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem itu digunakan.



Gambar 3.1 *use case* diagram pengguna

3.2.2 Flow Of Event (skenario)

Skenario merupakan penjelasan lebih detai dari kasus (*case*) dari awal hingga akhirnya diperoleh sebuah *output*.

Tabel 3.1 flow of event pengguna ke 1

Identifikasi	
Nomor	1.0.1
Nama	Pengaturan Tracker
Tujuan	Untuk pendeteksian wajah dan mata
Deskripsi	Sistem menerima masukkan yang dipilih pengguna dan berdasarkan masukkan tersebut maka akan terlihat pada saat melakukan pendeteksian wajah dan mata.
Tipe	Primer, esensial
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Menu tampilan di layar
Aksi Aktor	**Reaksi Sistem**
Pengguna mengklik Pendefenisian HUE untuk mendefinisian hue pada algoritma camshift, kemudian masukkan berapa lebar dan panjang untuk proses bagian yang akan di capture	USB kamera aktif dan akan menampilkan citra masukkan. Pada saat proses capture, sistem akan menyimpan hasil capture pada lokasi yang telah ditentukan.
Aktifkan checkbox periksa Cek varience dan isi nilai variance.	Menghitung dekomposisi koefisien vektor kemudian sistem akan mendapatkan koordinat (lokasi mata)
Untuk penentuan lokasi database pengguna klik browse	Menampilkan direktori yang nantinya dipilih lokasi database yang tersimpan sebelumnya untuk proses pendeteksian yang akan dipakai.
Isi Jumlah Gambar di database dan Jumlah EigenEyes	memeriksa dan mencocokkan gambar berdasarkan masukkan yang ditentukan
Masukkan nilai Max Jarak Dari Ruang Mata	menghitung vektor-vektor dari ruang mata

Masukkan nilai Rasio perbandingan Lebar Wajah & lebar template mata	Menghitung/mengkalkulasikan nilai yang di inputkan dan mencocokkan dengan template mata yang sudah ada.
Aktifkan checkbox Deteksi Pupil, Tandai Mata dengan active snakes, Pencocokkan bentuk Lonjong Mata. Lalu klik Simpan/Tutup.	Dapatkan koordinat mata kemudian akan mendeteksi pupil mata dengan menggunakan algoritma template matching, untuk menandai bola mata sistem akan mengkalkulasikan dan melakukan perhitungan dengan active countor dan pencocokan bentuk lonjong mata akan terlihat pada saat proses tracking dilakukan
Kondisi akhir	Mendeteksi wajah dan mata

Tabel 3.2 flow of event pengguna ke 2

Identifikasi	
Nomor	1.0.2
Nama	Mulai tracking
Tujuan	Melakukan pendeteksian mata
Deskripsi	Sistem melakukan pendeteksian mata
Tipe	Primer, esensial
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Menu tampilan di layar
Aksi Aktor	**Reaksi Sistem**
Pengguna memilih menu Mulai tracking	Sistem akan mengirim dan memeriksa Pengaturan tracking yang sudah di pilih oleh pengguna. Kemudian keluarannya akan ditampilkan pada saat pendeteksian.
Klik icon Capture	Mengcapture citra masukkan pada saat proses

	pendeteksian sedang berlangsung dan hasil capture akan di simpan di lokasi yang telah di tentukan.
Kondisi akhir	Aplikasi akan mendeteksi mata kemudian hasil capture akan tersimpan dilokasi yang telah ditentukan

Tabel 3.3 flow of event pengguna ke 3

Identifikasi	
Nomor	1.0.3
Nama	Stop tracking
Tujuan	Menghentikan proses pendeteksian yang sedang berjalan
Deskripsi	Menghentikan proses pendeteksian yang sedang berjalan
Tipe	Primer, esensial
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Menu tampilan di layar
Aksi Aktor	**Reaksi Sistem**
Pengguna memilih menu stop tracking	Sistem akan merespon dan akan me non aktifkan USB kamera.
Kondisi akhir	Pendeteksian akan berhenti dan kamera USB tidak aktif

Tabel 3.4 flow of event pengguna ke 4

Identifikasi	
Nomor	2.0.1
Nama	Lihat Database
Tujuan	Mengetahui template mata dan lokasi database mata yang akan digunakan
Deskripsi	Menampilkan database mata dan direktori database yang disimpan
Tipe	Primer, esensial

Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Menu tampil di layar	
Aksi Aktor		**Reaksi Sistem**
Pengguna memilih menu Lihat Database		Mencari dan menampilkan lokasi database mata yang sudah disimpan dan ditentukan sebelumnya.
Kondisi akhir	Menampilkan database mata beserta direktorinya yang akan digunakan untuk pendeteksian mata	

Tabel 3.5 flow of event pengguna ke 5

Identifikasi		
Nomor	2.0.2	
Nama	Hapus Database	
Tujuan	Menghapus database mata yang sebelumnya sudah tersimpan	
Deskripsi	Menghapus database mata	
Tipe	Primer, esensial	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Menu tampil di layar	
Aksi Aktor		**Reaksi Sistem**
Pengguna memilih menu Hapus database		Mencari lokasi database mata yang sudah ditentukan, setelah ditemukan database mata akan dihapus. Jika batal untuk menghapus databse maka akan kembali ke menu utama.
Kondisi akhir	Database mata akan terhapus	

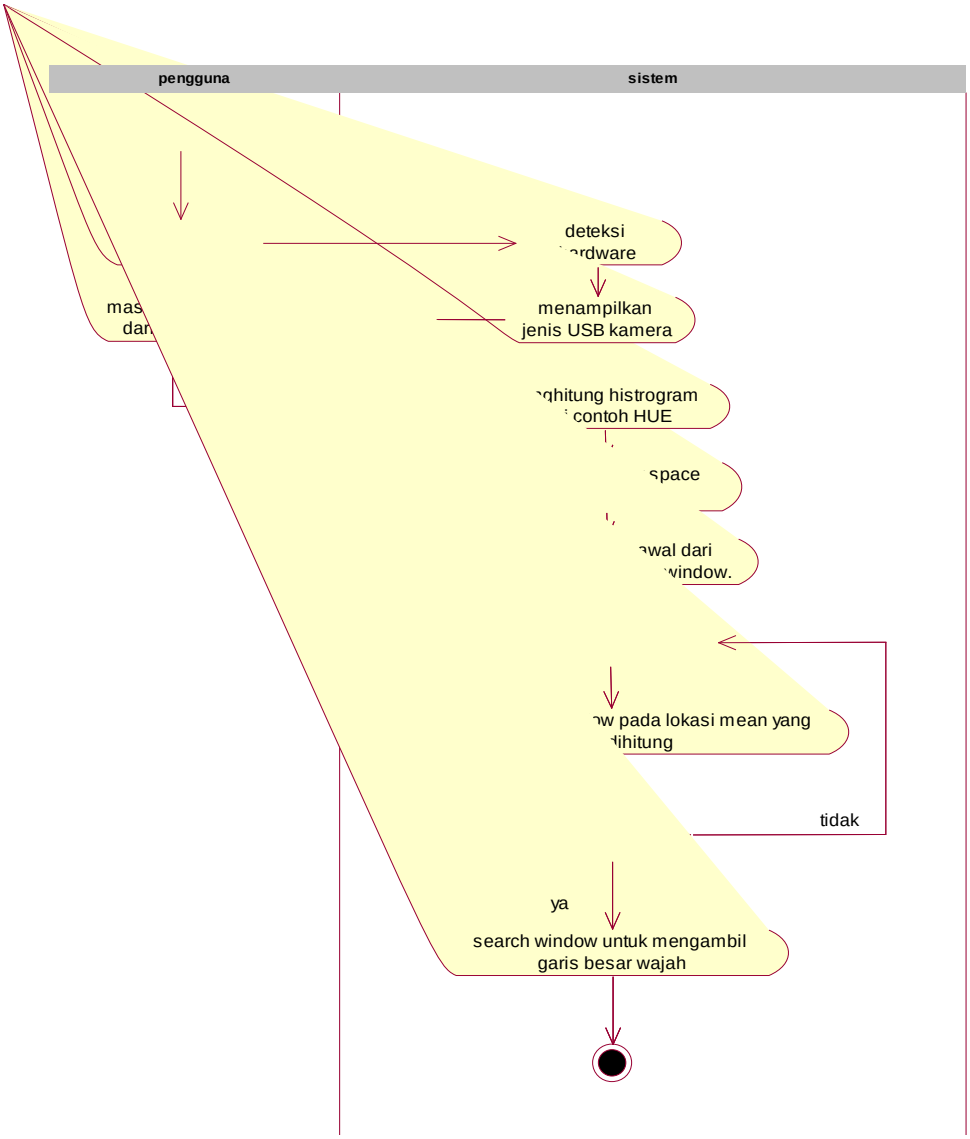
Tabel 3.6 flow of event pengguna ke 6

Identifikasi	
Nomor	3.0.1
Nama	Info

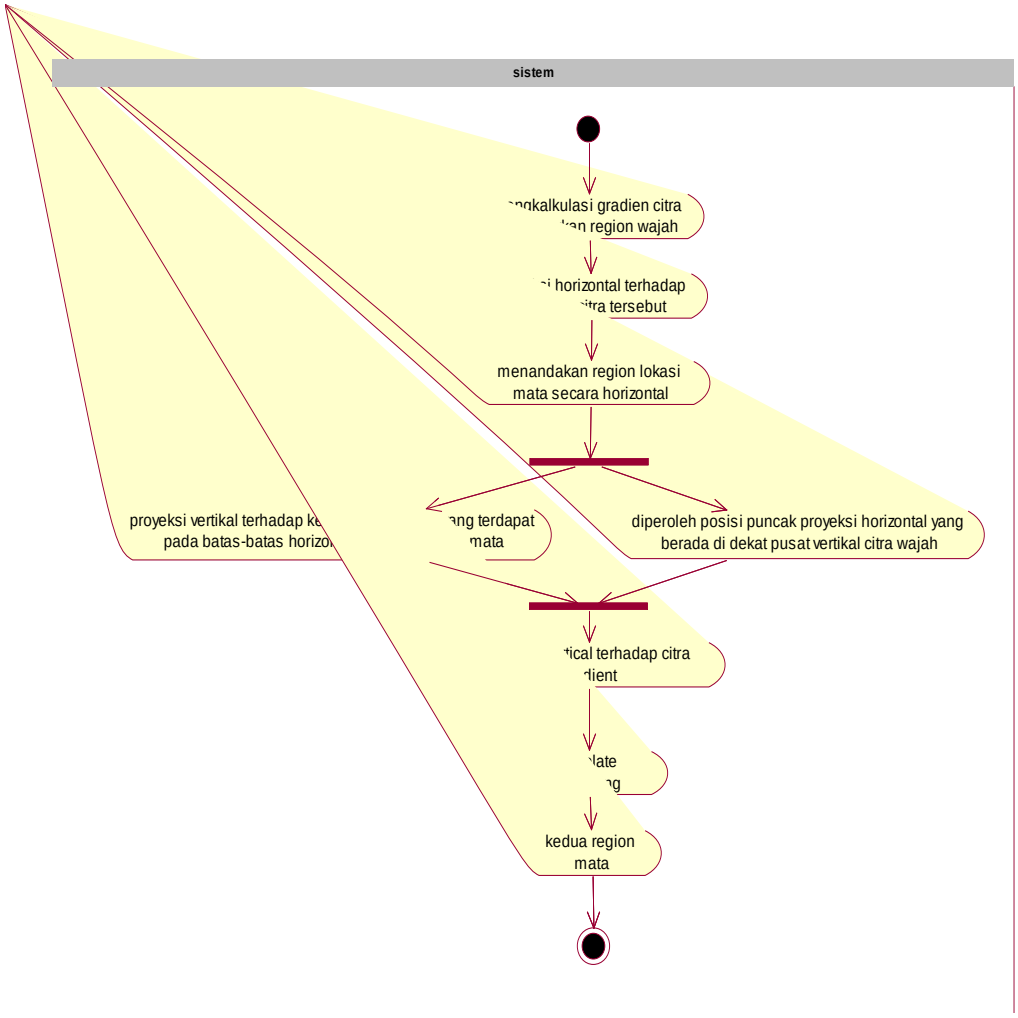
Tujuan	Mengetahui Informasi tentang pembuat	
Deskripsi	Tentang pembuat	
Tipe	Primer, esensial	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Menu tampil di layar	
Aksi Aktor		**Reaksi Sistem**
Pengguna memilih menu info		Sistem Menampilkan form tentang pembuat
Kondisi akhir	Form tentang pembuat tampil di layar	

3.2.3 Actifity Diagram

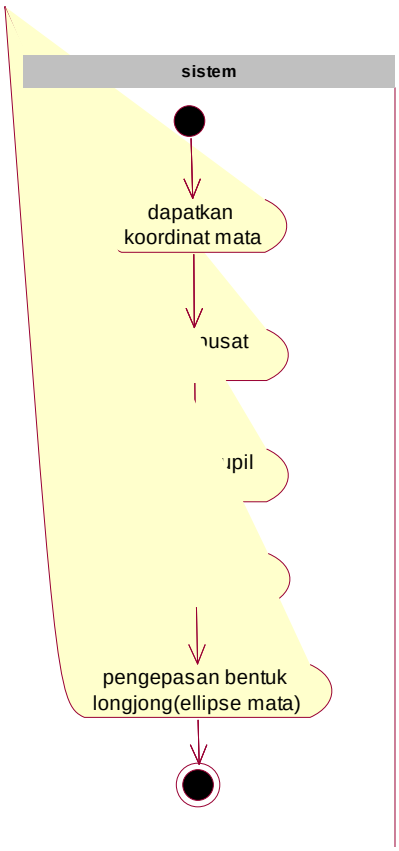
Activity Diagram digunakan untuk mengilustrasikan aliran fungsional dalam sebuah sistem. Suatu teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior* paralel.



Gambar 3.2 *activity diagram* deteksi wajah



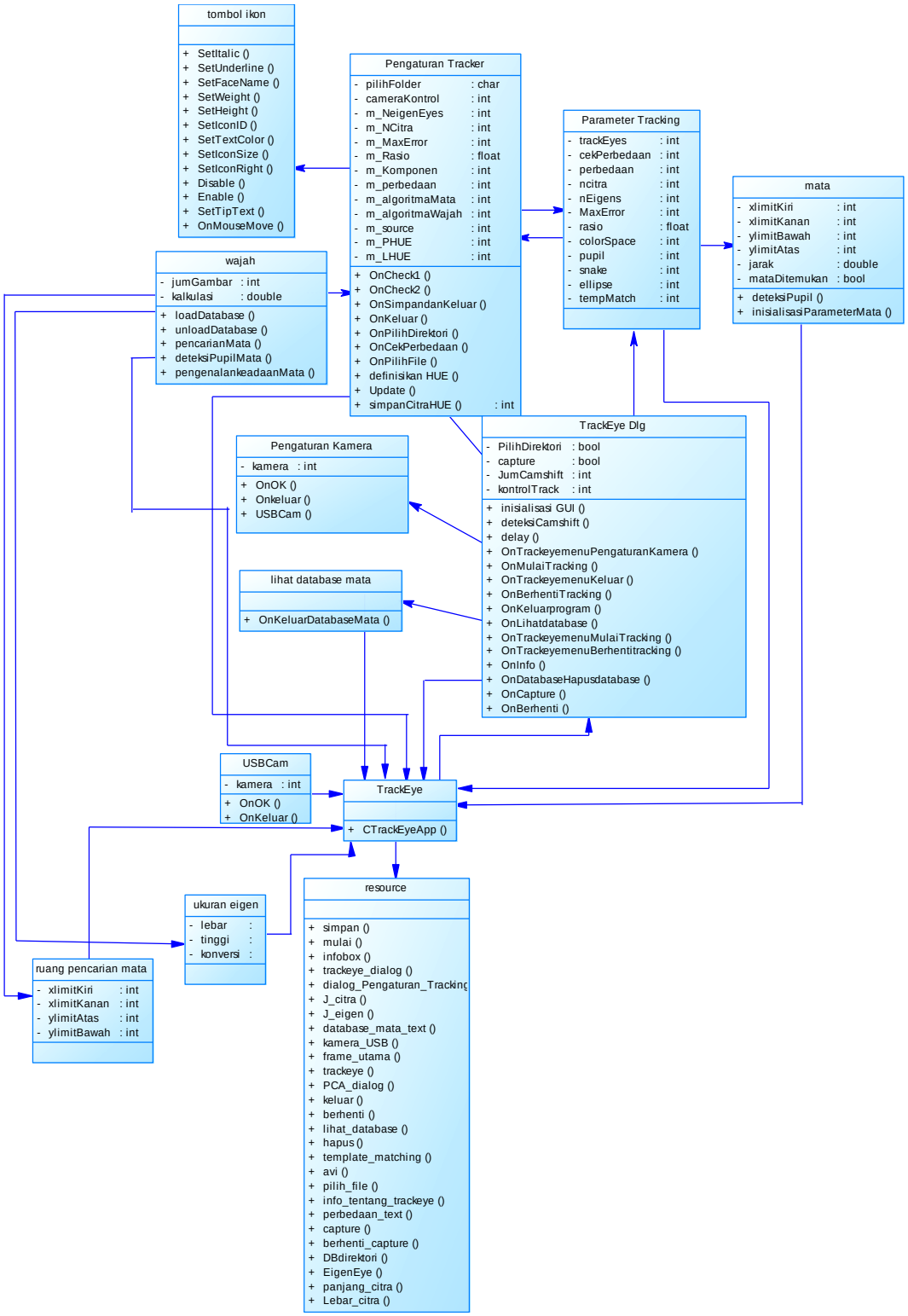
Gambar 3.3 *activity diagram* deteksi mata



Gambar 3.4 *activity diagram* deteksi pupil

3.2.4 Class Diagram

Class Diagram menggambarkan keterkaitan antar kelas dan mempresentasikan struktur dari sistem. Mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka.



Gambar 3.5 *class diagram trackeye*

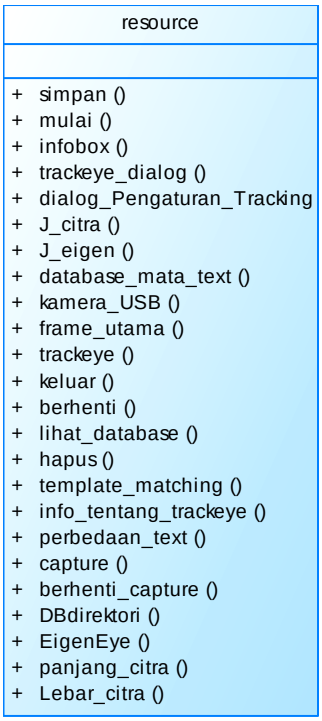
3.2.4.1 Spesifikasi Class

3.2.4.1.1 Class Resource

Class resouce adalah *class* yang digunakan sebagai *base class* dari kelas *trackeye.*

3.2.4.1.1.1 Deskripsi Atribut

Tidak ada atribut dalam kelas.



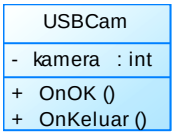
3.2.4.1.1.2 Deskripsi Layanan

Nama Layanan	***Keterangan***
Simpan	Menyimpan inputan pengaturan tracker dari user
Mulai	Mulai tracking
Infobox	Pengambilan objek info
Trackeye_dialog	Pengesetan objek trackeye Dlg
Pengaturan_tracker	Pengesetan pengaturan tracker
J_citra	Pengesetan jumlah citra
J_eigen	Pengesetan jumlah eigen
Database_mata_text	Direktori database mata
Kamera_USB	Pengambilan objek USBCam
Frame utama	Frame masukkan citra untuk pendeteksian
Trackeye	Pengambilan objek trackeye

Keluar	Keluar dari aplikasi
Berhenti	Berhenti tracking
Lihat_database	Pengambilan objek lihat database mata
Hapus	Menghapus database mata
Template_matching	Algoritma yang dipakai pada saat pendeteksian
Info_tentang_trackeye	Pengambilan objek tentang trackeye
Text_perbedaan	Pengesetan cek perbedaan/variance
Capture	Pengambilan gambar dari citra masukkan per frame
Berhenti_capture	Menghentikan proses capture
DB_direktori	Pengesetan direktori database
Eigeneye	Algoritma yang dipakai pada saat pendeteksian
Panjang_citra	Pengesetan untuk capture definisikan hue untuk algoritma camshaft
Lebar_citra	Pengesetan untuk capture definisikan hue untuk algoritma camshaft
Update	Mengupdate panjang dan lebar bagian citra untuk proses capture

3.2.4.1.2 Class USB Kamera

Class yang berguna untuk pendeteksian USB kamera yang aktif



3.2.4.1.2.1 Deskripsi Atribut

Nama	Tipe
Kamera	Integer

3.2.4.1.2.2 Deskripsi Layanan

Nama Layanan	*Keterangan*
OnOK	Menyetujui hasil pendeteksian kamera oleh sistem
Onbatal	Keluar dari form jika kamera tidak terdeteksi

3.2.4.1.3 Class Ukuran Eigen

Class yang digunakan untuk menghitung atau mengkalkulasi ukuran eigen pada saat proses pendeteksian



3.2.4.1.3.1 Deskripsi Atribut

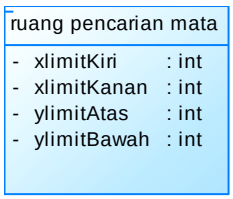
Nama	Tipe
Lebar	integer
Tinggi	Integer
Konversi	integer

3.2.4.1.3.2 Deskripsi Layanan

Tidak ada layanan dalam class

3.2.4.1.4 Class Ruang Pencarian Mata

Class untuk menghitung batas-batas ruang untuk pencarian mata.



3.2.4.1.4.1 Deskripsi Atribut

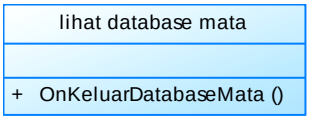
Nama	Tipe
XlimitKiri	integer
XlimitKanan	integer
YlimitAtas	integer
YlimitBawah	integer

3.2.4.1.4.2 Deskripsi Layanan

Tidak ada layanan yang di deskripsikan dalam class ini.

3.2.4.1.5 Class Lihat Database Mata

Class yang digunakan untuk menampilkan database mata



3.2.4.1.5.1 Deskripsi Atribut

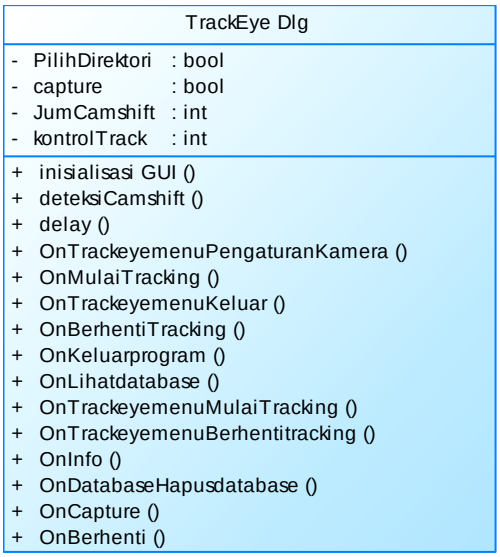
Tidak ada atribut yang dideskripsikan dalam *class* ini.

3.2.4.1.5.2 Deskripsi Layanan

Nama Layanan	***Keterangan***
OnKeluarDatabaseMata	Keluar dari database mata yang sudah ditampilkan

3.2.4.1.6 Class Trackeye Dlg

Class yang digunakan untuk aktivasi *trackeye*



3.2.4.1.6.1 Deskripsi Atribut

Nama	**Tipe**
PilihDirektori	Bool
Capture	Bool
JumlahCamshift	Integer
KontrolTrack	Integer

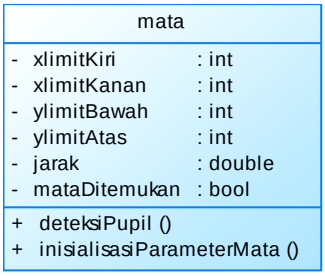
3.2.4.1.6.2 Deskripsi Layanan

Nama Layanan	***Keterangan***
Inisialisasi GUI	Tampilan interface
DeteksiCamshift	Pendeteksian wajah menggunakan algoritma camshift

Delay	Inisialisasi perframe pada saat proses pendeteksian
OnTrackeyeMenuPengaturanKamera	Mengeset pengaturan kamera
OnMulaiTracking	Melakukan proses pendeteksian mata
OnTrackeyeMenuKeluar	Keluar dari aplikasi
OnBerhentiTracking	Menghentikan pendeteksian yang sedang berjalan
OnKeluarProgram	Keluar dari aplikasi
OnLihatDatabase	Menampilkan database mata
OnTrackeyeMenuMulaiTracking	Melakukan proses pendeteksian mata
OnTrackeyemenuBerhentiTracking	Menghentikan pendeteksian yang sedang berjalan
OnInfo	Menampilkan tentang trackeye
OnDatabaseHapusDatabase	Menghapus database mata yang tersimpan
OnCapture	Mengambil gambar pada saat proses pendeteksian
OnBerhenti	Menghentikan proses capture

3.2.4.1.7 Class Mata

Class yang digunakan untuk proses pendeteksian mata



3.2.4.1.7.1 Deskripsi Atribut

Nama	Tipe
XlimitKiri	Integer
XlimitKanan	Integer
YlimitBawah	Integer
YlimitAtas	Integer
Jarak	Double
mataDitemukan	Bool

3.2.4.1.7.2 Deskripsi Layanan

Nama Layanan	*Keterangan*
deteksiPupil	Melakukan proses pendeteksian pupil
InisialisasiParameterMata	Menginisialisasikan parameter variable mata

3.2.4.1.8 Class Parameter Tracking

Class yang digunakan untuk melakukan perhitungan algoritma pada saat akan melakukan pendeteksian



3.2.4.1.8.1 Deskripsi Atribut

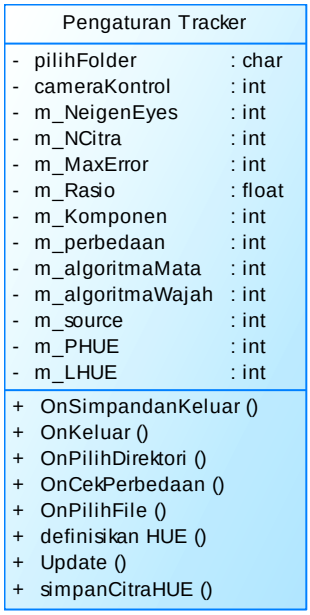
Nama	Tipe
Trackeyes	Integer
cekPerbedaan	Integer
Perbedaan	Integer
nCitra	Integer
nEigens	Integer
maxError	Integer
Rasio	Float
Colorspace	Integer
Pupil	Integer
Snakes	Integer
Ellipse	Integer
tempMatch	Integer

3.2.4.1.8.2 Deskripsi Layanan

Tidak ada layanan yang dideskripsikan dalam *class* ini

3.2.4.1.9 Class Pengaturan Tracker

Class yang digunakan untuk pengaturan pendeteksian mata yang di inputkan oleh pengguna.



3.2.4.1.9.1 Deskripsi Atribut

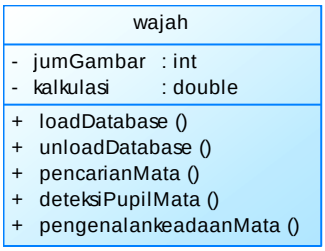
Nama	Tipe
pilihFolder	Char
kameraKontrol	Integer
m_NeigenEyes	Integer
m_NCitra	Integer
m_MaxError	Integer
m_rasio	Float
m_komponen	Integer
m_perbedaan	Integer
m_algoritmaMata	Integer
m_algoritmaWajah	Integer
M_source	Integer
M_PHUE	Integer
M_LHUE	Integer

3.2.4.1.9.2 Deskripsi Layanan

Nama Layanan	***Keterangan***
OnSimpan/Keluar	Hasil masukkan dari user disimpan untuk proses pendeteksian yang nanti akan dilakukan sekaligus keluar dari menu pengaturan trackeye
OnKeluar	Keluar dari form definisikan HUE
OnpilihDirektori	Menentukan direktori lokasi database yang tersimpan
OnCekPerbedaan	Melakukan perhitungan untuk warna kulit dan juga color space
OnDefinisikanHUE	Mengambil contoh dari inputan citra untuk pendeteksian wajah
OnUpdate	Mnegupdate inputan ukuran panjang dan lebar yang akan dicapture untuk pendefinisian HUE
OnSimpanCitraHUE	Menyimpan hasil citra yang telah dicapture sebelumnya

3.2.4.1.10 Class Wajah

Class wajah ini mempunyai hubungan asosiasi ke *class* pencarian ruang *eigen* dan *class* ukuran *eigen.*



3.2.4.1.10.1 Deskripsi Atribut

Nama	Tipe
jumGambar	Integer
Kalkulasi	Double

3.2.4.1.10.2 Deskripsi Layanan

Nama Layanan	***Keterangan***
loadDatabase	Menampilkan database
unLoadDatabase	Batal menampilkan database
pencarianMata	Proses pencarian posisi mata
deteksiPulpilMata	Proses Pendeteksian pupil mata
pengenalanKeadaanMata	Menentukan region-region mata

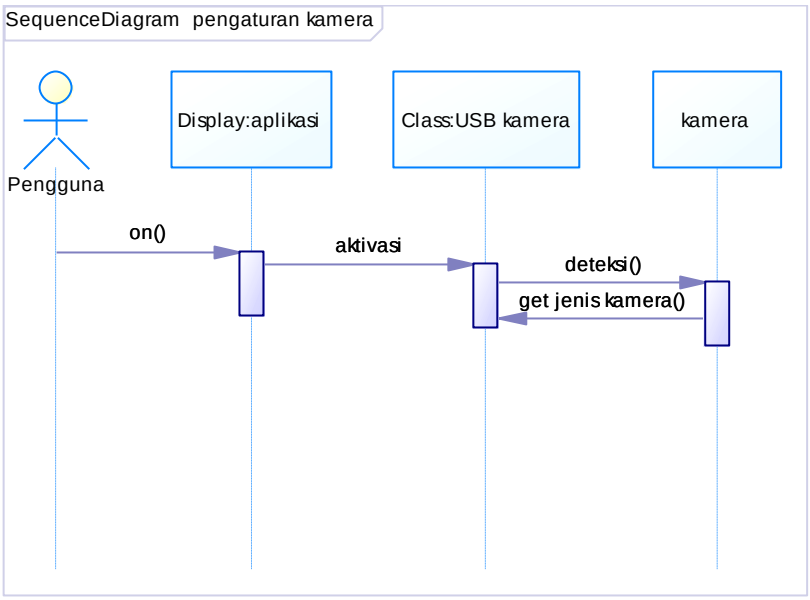
3.3 Perancangan

Tahap ini terdiri dari pemodelan kasus menggunakan *sequence diagram*, *communication diagram*, beserta rancangan antar muka pengguna.

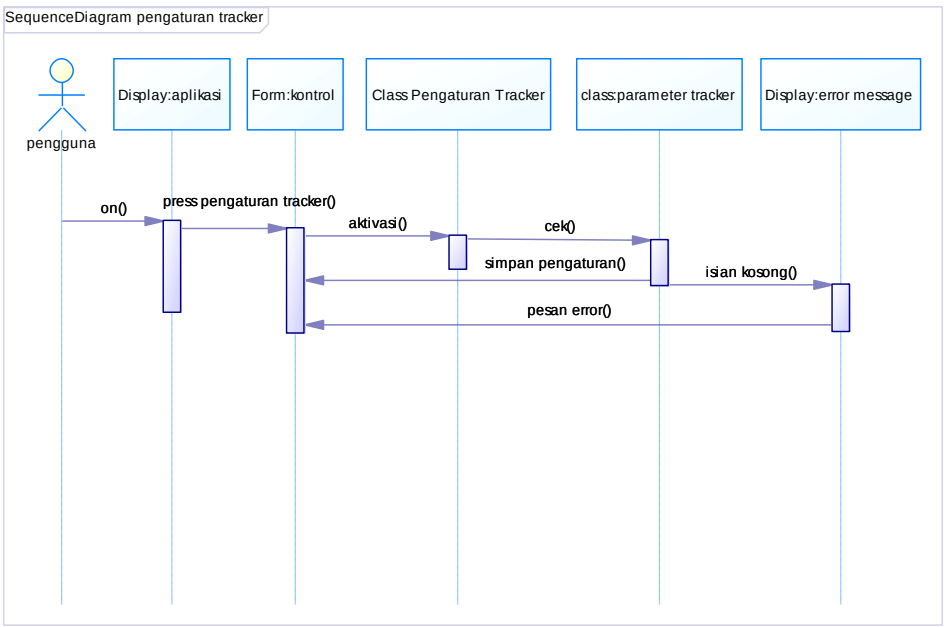
3.3.1 Sequence Diagram

Sequance Diagram merupakan diagram untuk menggambarkan perilaku sistem terhadap suatu interaksi yang dilakukan pada sistem tersebut. Secara khusus, menjabarkan *behavior* skenario tunggal. Diagram tersebut menunjukkan

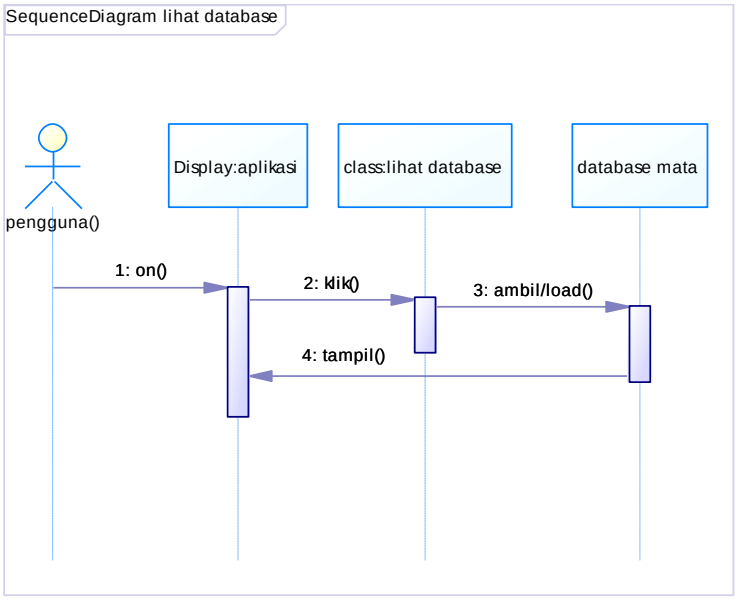
sejumlah objek contoh dan pesan-pesan yang melewati objek-objek di dalam *use case*.



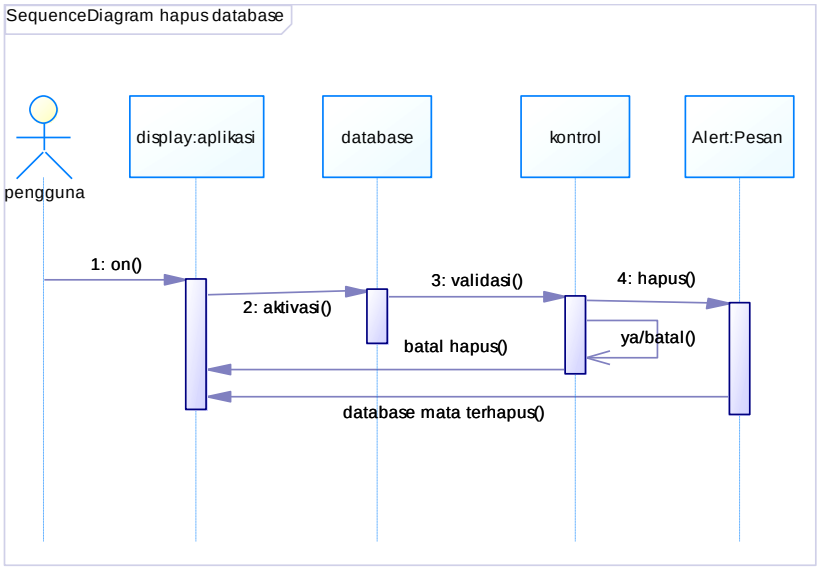
Gambar 3.6 *sequence diagram* pengaturan kamera



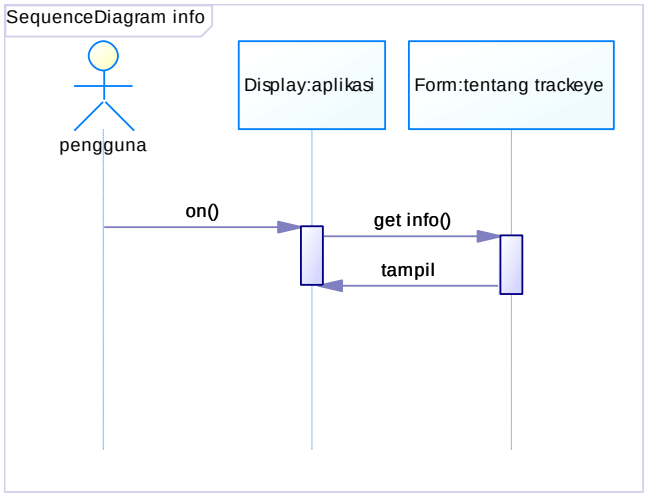
Gambar 3.7 *sequence diagram* pengaturan tracking



Gambar 3.8 *sequence diagram* lihat database



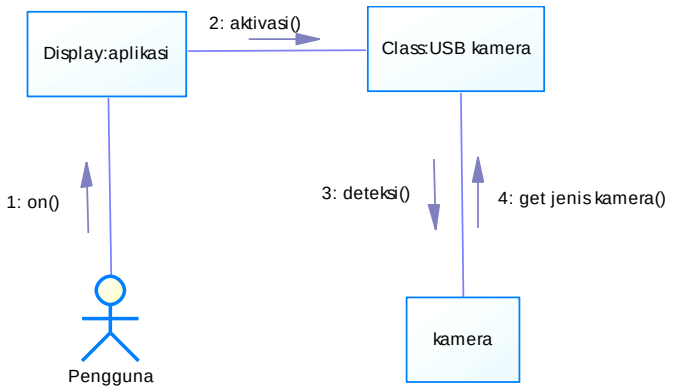
Gambar 3.9 *sequence diagram* hapus database



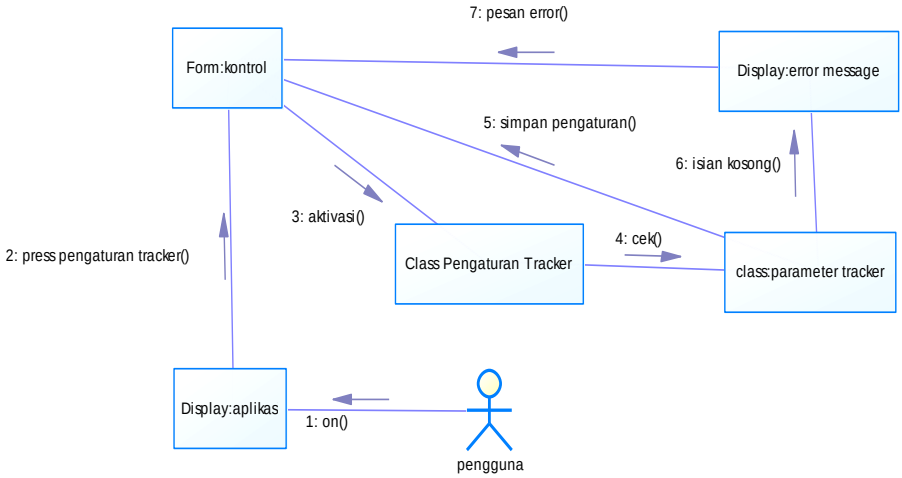
Gambar 3.10 *sequence diagram* info

3.3.2 Communication Diagram

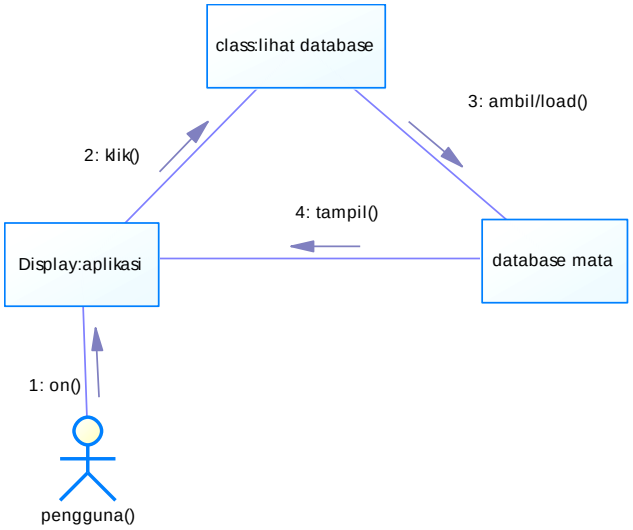
Communication Diagram memberi tekanan pada hubungan-hubungan data antar partisipan yang berbeda dalam sebuah interaksi. *Communication diagram* tidak menggambarkan setiap partisipan sebagai suatu garis alir dan menunjukkan pesan-pesan dengan arah vertical yang dilakukan *sequence diagram*, tetapi memungkinkan untuk menempatkan partisipan-partisipan secara bebas, untuk menggambarkan hubungan-hubungan yang menunjukkan bagaimana partisipan berhubungan dan menggunakan nomorisasi untuk menunjukkan bagian dari pesan-pesan.



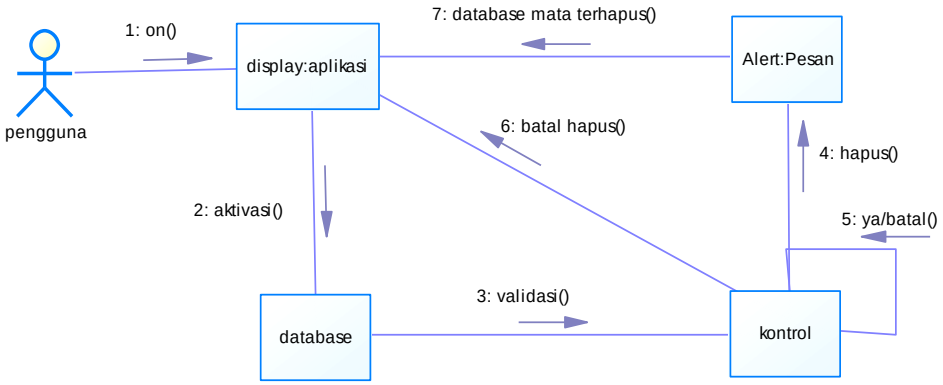
Gambar 3.11 communication diagram pengaturan kamera



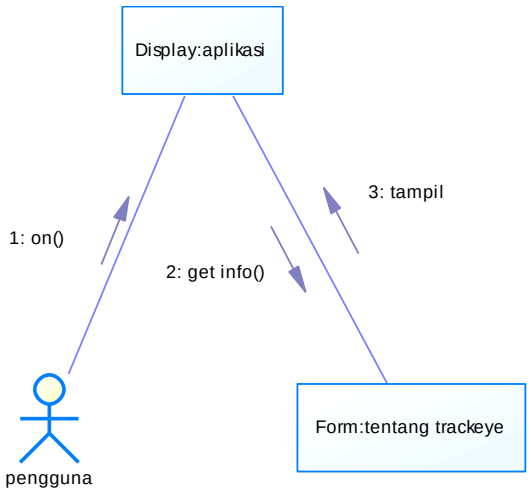
Gambar 3.12 *communication diagram* pengaturan *tracking*



Gambar 3.13 *communication diagram* lihat database



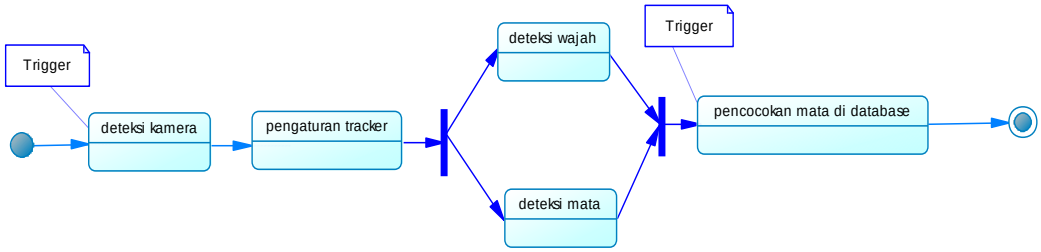
Gambar 3.14 *communication diagram* hapus database



Gambar 3.15 *communication diagram* info

3.3.3 State Chart Diagram

Teknik yang umum digunakan untuk menggambarkan *behavior* sebuah sistem.



Gambar 3.16 *state chart diagram* proses pendeteksian mata

3.4 Implementasi Algoritma

3.4.1 Algoritma Camshift

Berikut ini adalah implementasi dari penggunaan algoritma Camshift pada proses pendeteksian wajah manusia untuk menentukan posisi mata dengan menggunakan bahasa pemrograman Visual C++.

Tabel 3.7. Algoritma Camshift untuk pendeteksian wajah

```
// Inputs for CamShift algorithm

IplImage* HUE = cvCreateImage(cvGetSize(SampleForHUE),
IPL_DEPTH_8U, 1);

extractHUE(SampleForHUE, HUE);    //  **  Extract HUE information

int hist_size      = 20;

float ranges[]      = { 0, 180 };

float* pranges[]  = {ranges};

hist = cvCreateHist( 1, &hist_size, CV_HIST_ARRAY, pranges, 1 );

cvCalcHist(&HUE, hist);    // Calculate histogram of HUE part

hueFrame = cvCreateImage(cvGetSize(CameraFrame), IPL_DEPTH_8U, 1);

backProject = cvCreateImage(cvGetSize(CameraFrame), IPL_DEPTH_8U,
1);

extractHUE(CameraFrame, hueFrame);

while (trackControl != 0)

{

    extractHUE( CameraFrame, hueFrame );

    cvCalcBackProject( &hueFrame, backProject, hist ); //
Probability is formed

    //cvShowImage("Tester2", backProject);

    cvCamShift( backProject, searchWin, cvTermCriteria(
```

```
CV_TERMCRIT_EPS |

            CV_TERMCRIT_ITER, 15, 0.1 ), &comp, &faceBox );

    searchWin = comp.rect;

}
```

3.4.2 Algoritma Template Matching

Berikut ini adalah implementasi dari penggunaan algoritma Camshift pada proses pendeteksian wajah manusia untuk menentukan posisi mata dengan menggunakan bahasa pemrograman Visual C++.

Tabel 3.8. Algoritma Template Matching untuk pendeteksian mata

```
// Template Matching for Eye detection

void Face::findEyes_TM(IplImage* faceImage, TrackingSettings*
settings)

{

    CvSize faceSize; faceSize = cvGetSize(faceImage);

    // Load Template from the eye database

    CString fileName;

    // Name of the template for left eye

    fileName.Format("%s\\eye%d.jpg", settings->params-
>DBdirectory, 0);

    IplImage* eyeImage_Left    = cvLoadImage(fileName, -1);

    // Name of the template for left eye

    fileName.Format("%s\\eye%d.jpg", settings->params-
>DBdirectory, 1);

    IplImage* eyeImage_Right = cvLoadImage(fileName, -1);

    IplImage* tempTemplateImg_Left; IplImage*
tempTemplateImg_Right;
```

```
    IplImage* templateImg_Left; IplImage* templateImg_Right;

    if (eyeImage_Left == NULL || eyeImage_Right == NULL)

    {

        MessageBox(NULL, "Templates can not be loaded.\n

            Please check your eye database folder", "Error",
MB_OK||MB_ICONSTOP);

        exit(1);

    }

    else

    {

        //    Change color space according to the settings entered
by the user

        tempTemplateImg_Left =
cvCreateImage(cvGetSize(eyeImage_Left), IPL_DEPTH_8U, 1);

        changeColorSpace(settings, eyeImage_Left,
tempTemplateImg_Left);

        tempTemplateImg_Right =

                    cvCreateImage(cvGetSize(eyeImage_Right),
IPL_DEPTH_8U, 1);

        changeColorSpace(settings, eyeImage_Right,
tempTemplateImg_Right);

       float idealWidth = faceSize.width * settings->params-ratio;

        float conversionRatio =
idealWidth/(float)tempTemplateImg_Left->width;



        CvSize newSize;

        newSize.width = (int)idealWidth;

        newSize.height = (int)(tempTemplateImg_Left-
>height*conversionRatio);
```

```
templateImg_Left = cvCreateImage(newSize, IPL_DEPTH_8U, 1);

 cvResize(tempTemplateImg_Left, templateImg_Left,
CV_INTER_LINEAR); // was NN

 cvReleaseImage(&eyeImage_Left);

 cvReleaseImage(&tempTemplateImg_Left);



templateImg_Right = cvCreateImage(newSize, IPL_DEPTH_8U,1);

 cvResize(tempTemplateImg_Right, templateImg_Right,
CV_INTER_LINEAR); // was NN

 cvReleaseImage(&eyeImage_Right);

 cvReleaseImage(&tempTemplateImg_Right);
```

3.4.3 Algoritma EigenEye

Berikut ini adalah implementasi dari penggunaan algoritma Eigeneye untuk menentukan nilai-nilai eigen.

Tabel 3.9 Algoritma Eigeneyes untuk pendeteksian mata

```
void Face::findEyes(IplImage* faceImage, TrackingSettings* settings)

{

    IplImage**images=(IplImage**)malloc(sizeof(IplImage*)*numOfImages);

    IplImage**eigens = (IplImage**)malloc(sizeof(IplImage*)*numOfImages);

    IplImage* averageImage;

    IplImage* projection;

    CvSize faceSize; faceSize = cvGetSize(faceImage);

    eigenSize newEigenSize;
```

```
    newEigenSize.width = faceSize.width * settings->params->ratio;

    newEigenSize.conversion=((float)newEigenSize.width)/
((float)database[0]->width);

    newEigenSize.height=((float)database[0]->height)*
newEigenSize.conversion;

    CvSize newSize;

    newSize.width = (int)newEigenSize.width;

    newSize.height = (int)newEigenSize.height;

    IplImage* tempImg = cvCreateImage( newSize, IPL_DEPTH_8U, 1);

    //    **********Initializations**********

    for (int i=0; iparams->nImages; i++)

    {

        images[i] = cvCreateImage(newSize, IPL_DEPTH_8U, 1);

        cvResize(database[i], images[i], CV_INTER_LINEAR); // was NN

    }

    cvShowImage("Eigen", images[0]);

    cvReleaseImage(&tempImg);



    //    Create space for EigenFaces

    for (i=0; iparams->nImages; i++)

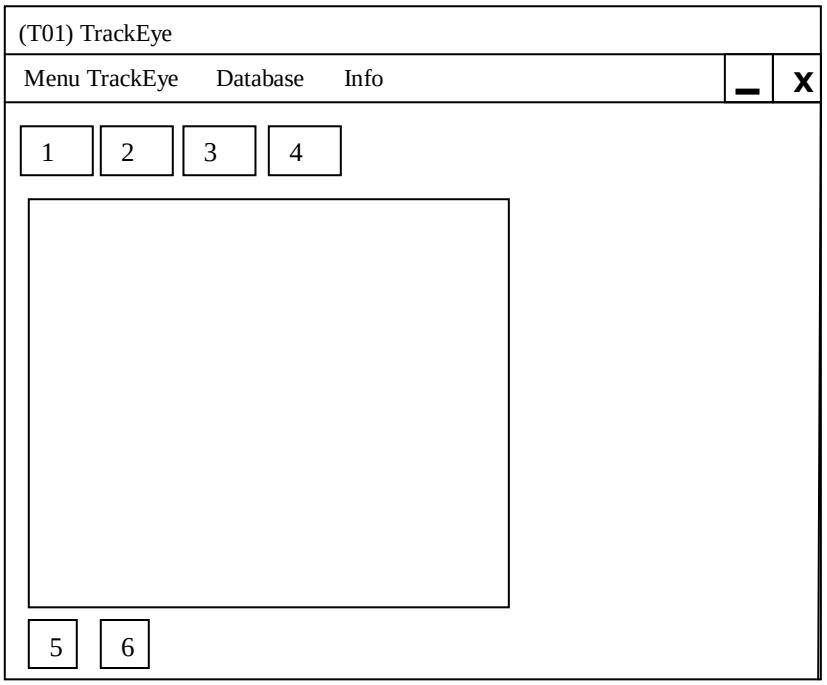
        eigens[i] = cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_32F,
1);
```

```
averageImage = cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_32F, 1);

projection = cvCreateImage(cvGetSize(images[0]), IPL_DEPTH_8U, 1);
```

3.5 Rancangan Antarmuka Pengguna

Rancangan antarmuka pengguna merupakan draft dari antarmuka yang akan diterapkan pada perangkat lunak deteksi mata. Form-form apa saja yang terdapat dalam aplikasi akan dijelaskan disini.

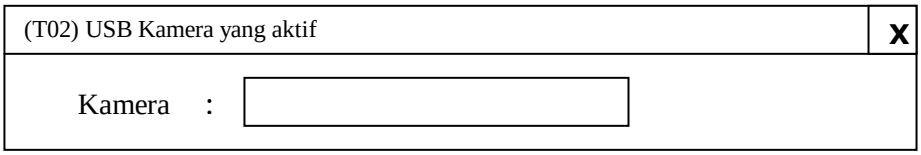


Gambar 3.17 antarmuka form main

Pada form main terdapat 6 tombol shortcut, diantaranya adalah :

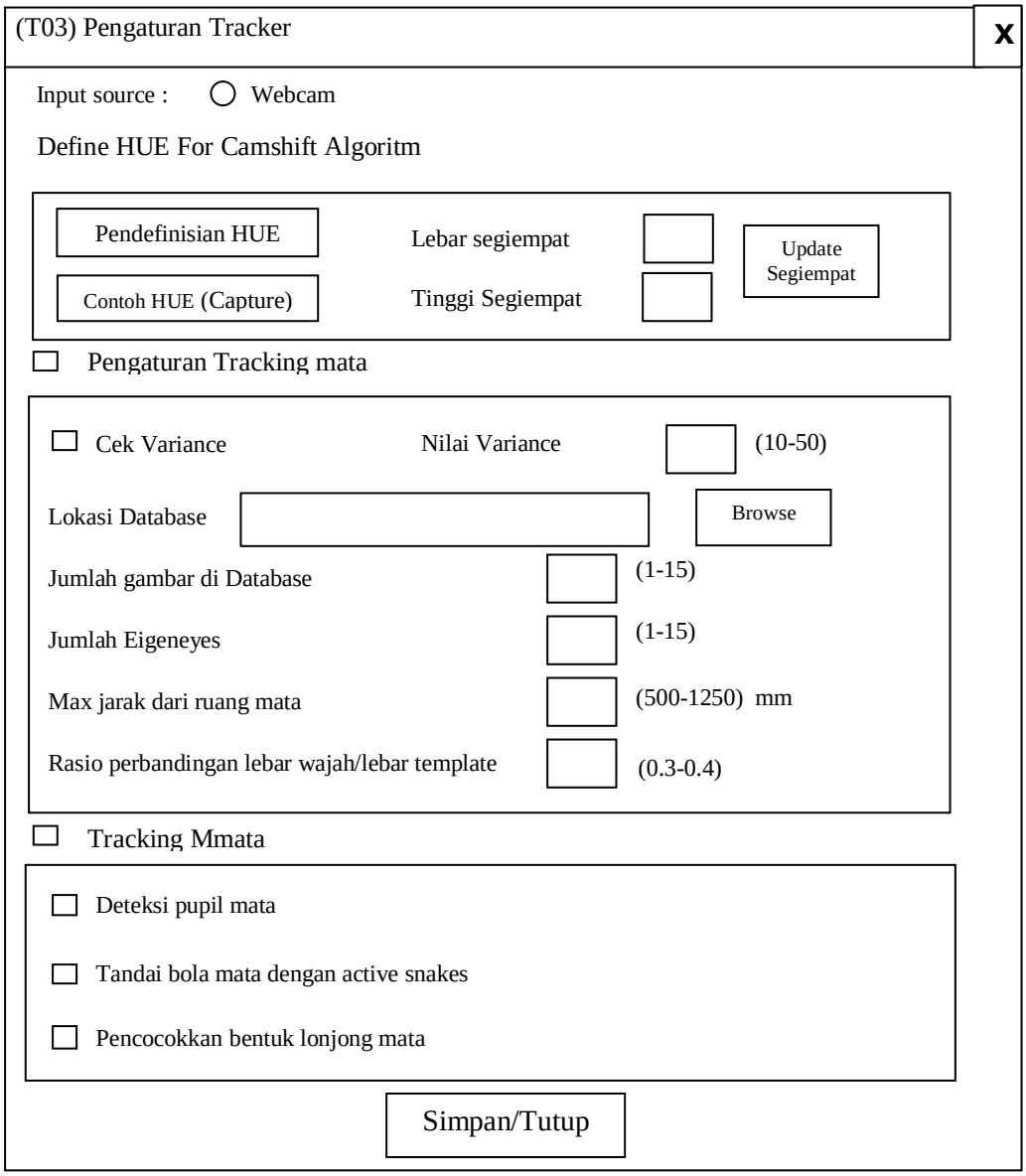
1. Tombol 1 adalah tombol untuk menampilkan form pengaturan *Tracking*.
2. Tombol 2 adalah tombol untuk Mulai *tracking*
3. Tombol 3 untuk *Stop tracking*.

4. Tombol 4 untuk keluar dari aplikasi.
5. Tombol 5 untuk meng*capture* video pada saat aplikasi *track eye* di jalankan.
6. Tombol 6 untuk menghentikan semantara pendeteksian mata (*pause*) pada saat *track eye* dijalankan.



Gambar 3.18 Antarmuka form USB kamera

Form USB kamera adalah untuk menampilkan jenis kamera yang terdeteksi, ini untuk memastikan bahwa kamera terhubung dan aktif.



Gambar 3.19 antarmuka form pengaturan tracking

Didalam form pengaturan *tracking* terdapat, 1 *radio button* 4 checkbox, 3 tombol button dan 7 textbox, diantaranya :

Radio Button

1. Webcam

Checkbox

1. Checkbox cek varience
2. Checkbox deteksi pupil mata
3. Checkbox tandai bola mata dengan active snakes
4. Checkbox pencocokkan bentuk lonjong mata

Tombol Button

1. Pendefinisian HUE

   Memilih bagian kulit wajah yang akan di definisikan.

2. Contoh HUE (capture)

   Menyimpan hasil citra hue pada saat calon area hue sudah ditentukan.

3. Update segi empat

   Tombol Untuk merubah lebar dan panjang pada saat akan melakukan pengcapturan pendefinisian hue.

4. Browse

   Merupakan tombol untuk menentukan lokasi database yang akan digunakan.

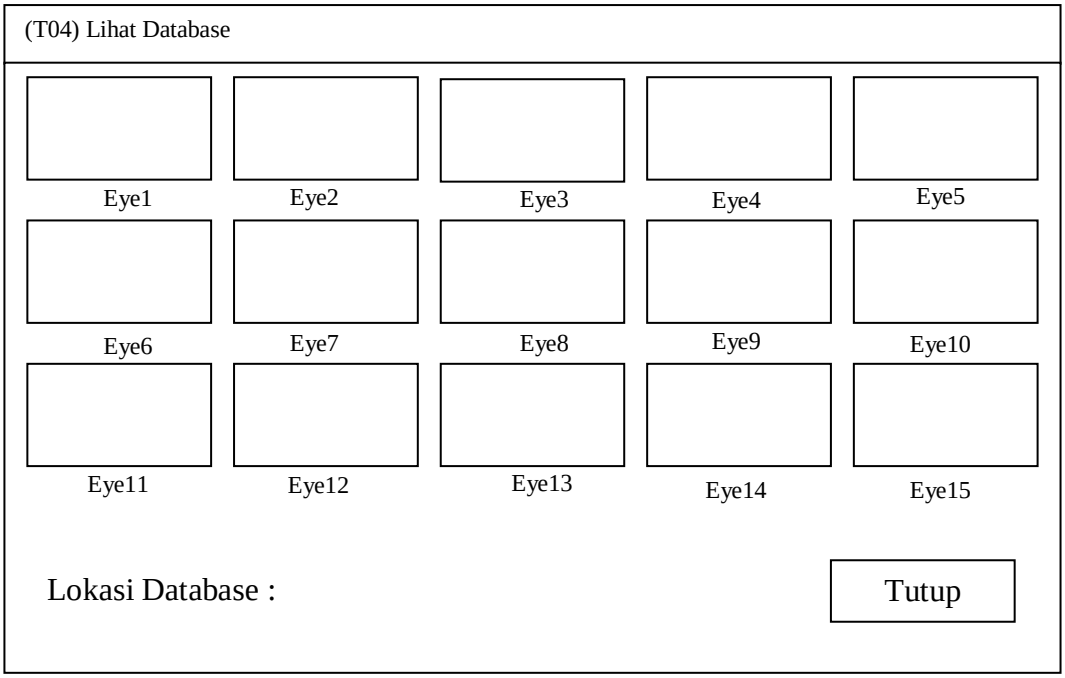
5. Sinpan/tutup

   Merupakan tombol untuk menyimpan semua pengaturan yang telah kita tentukan semua, dan sekaligus keluar dari pengaturan *tracker*.

Textbox

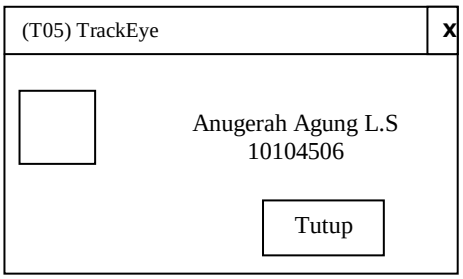
1. Nilai Variance
2. Lebar segiempat
3. Tinggi segiempat
4. Jumlah gambar di database

5. Jumlah *eigeneyes*
6. Max jarak dari ruang mata
7. Rasio perbandingan lebar wajah/lebar template mata

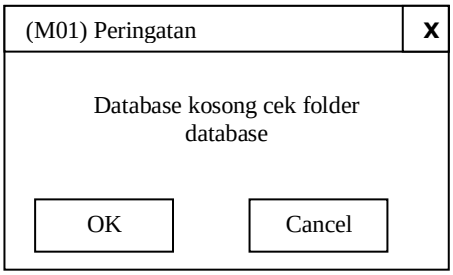


Gambar 3.20 antarmuka form lihat database mata

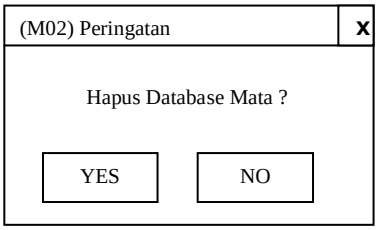
Dalam menu database terdapat sub menu lihat database, hapus database. Terdiri dari 15 panel, 1 tombol button dan 1 label.



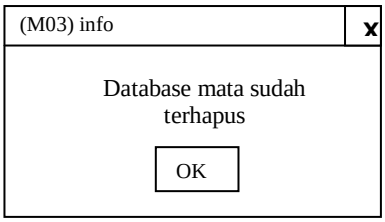
Gambar 3.21 antarmuka form info



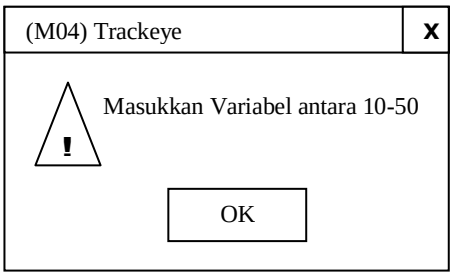
Gambar 3.22 pesan database kosong



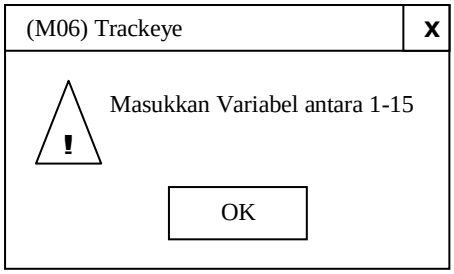
Gambar 3.23 pesan peringatan hapus database mata



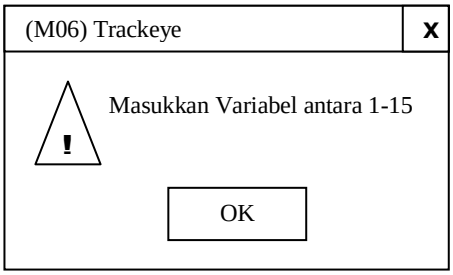
Gambar 3.24 pesan database terhapus



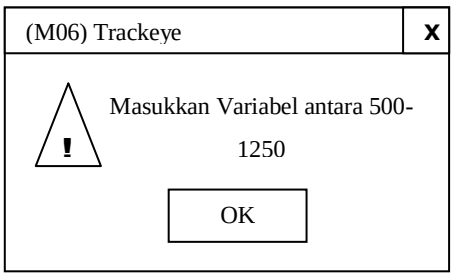
Gambar 3.25 Pesan peringatan jika nilai variance inputannya tidak sesuai



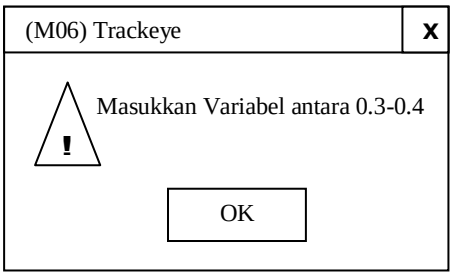
Gambar 3.26 Pesan peringatan jika inputan jumlah gambar di database tidak sesuai



Gambar 3.27 Pesan peringatan jika inputan jumlah eigeneye tidak sesuai

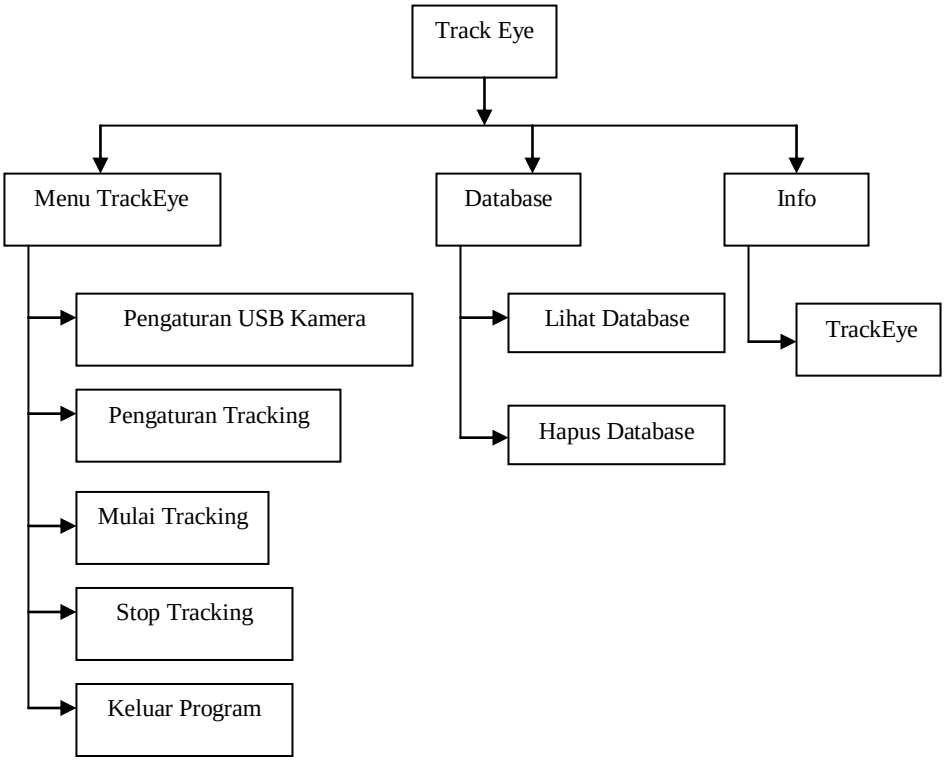


Gambar 3.28 Pesan peringatan jika inputan max jarak dari ruang mata tidak sesuai



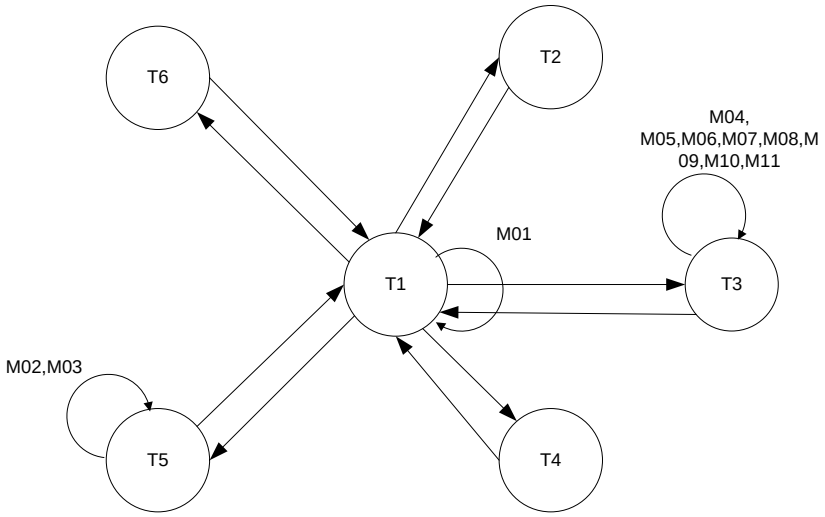
Gambar 3.29 Pesan peringatan jika inputan Rasio lebar wajah/lebar template mata tidak sesuai

3.6 Struktur Menu



Gambar 3.30 Struktur Menu

3.7 Jaringan Semantik



Gambar 3.31 jaringan semantik Trackeye