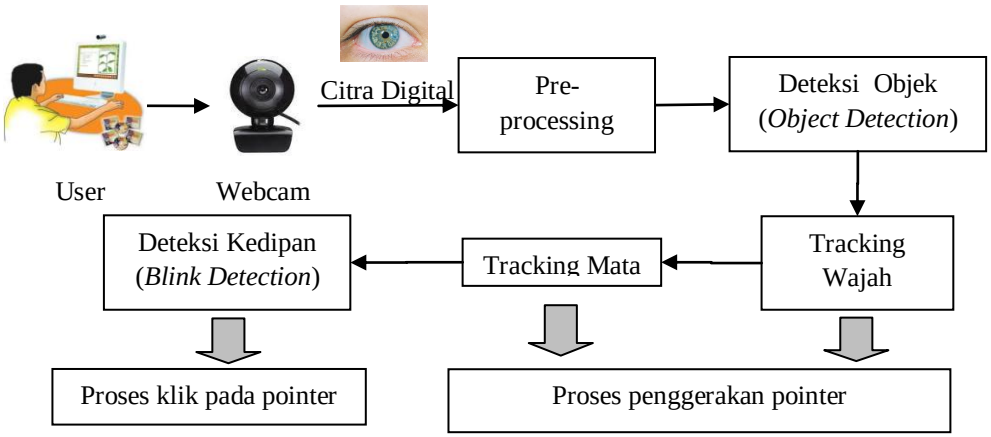
BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Deskripsi Umum Sistem

Proses pengendalian pointer dengan mata dapat dijelaskan sebagai berikut pertama kamera akan mengkalibrasi gambar dari *user* yang sedang berinteraksi dengan komputer, kemudian gambar tersebut akan diproses untuk dilakukan pengenalan apakah pada gambar tersebut terdapat mata atau tidak. Apabila pada gambar tersebut terdapat mata maka kemudian akan dilakukan deteksi pergerakan wajah dan deteksi pergerakan bola mata. Deteksi pergerakan bola mata ini akan digunakan untuk proses pergerakan *pointer.* Kemudian dilakukan deteksi kedipan pada kedua mata untuk proses klik pada *pointer*. Apabila yang berkedip adalah mata sebelah kiri, maka sistem akan melakukan proses klik kiri. Dan apabila yang berkedip adalah mata sebelah kanan, maka sistem akan melakukan proses klik kanan. Proses pengendalian pointer dengan mata digambarkan sebagai berikut :



Gambar 3.1 Proses pengendalian *pointer* dengan mata

Tahap – tahap pengendalian *pointer* dengan mata :

1. Tahap Kalibrasi

   Tahap awal posisi *user* harus tegak lurus dengan kamera. Kemudian kamera akan mengkalibrasi gambar dari user secara *real time* dan akan diubah menjadi gambar digital. Gambar tersebut akan digunakan untuk tahap selanjutnya.

2. Tahap *Pre-processing Image*

   Pada tahap *pre-processing image* dilakukan proses *grayscaling*, *tresholding* dan *scaling* untuk dilakukan pengenalan pola.

3. Tahap Deteksi Objek

   Pada tahap ini akan dilakukan pendeteksian objek wajah dan mata. Untuk mendeteksi objek wajah dan mata digunakan metode *Haar Cascade Classifier*.

4. Tahap *Tracking* Wajah

   Tahap berikutnya adalah *tracking* wajah. *Output* dari deteksi dan *tracking* wajah akan digunakan untuk tahap *tracking* mata.

5. Tahap *Tracking* Mata

   Tahap *tracking* mata dilakukan untuk menemukan lokasi dari mata. Untuk *tracking* mata digunakan metode *template machine*, yang akan digunakan untuk proses menggerakkan *pointer*. Hasil dari *tracking* mata akan digunakan pada tahap deteksi kedipan (*blink detection*).

6. Tahap Deteksi Kedipan

Pada tahap ini mata yang sedang di-*tracking* pada tahap sebelumnya akan dilakukan deteksi kedipan dengan menggunakan operasi morfologi yang akan digunakan untuk proses klik pada *pointer*.

3.2 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh kedalam bagian-bagian komponennnya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

3.2.1 Analisis Deteksi Objek

Analisis deteksi Objek ini dibagi menjadi 4 tahap, yaitu :

1. Tahap pertama adalah tahap kalibrasi yaitu proses pengambilan citra secara *real time* yang kemudian akan di ubah menjadi citra digital.
2. Tahap kedua adalah tahap pengenalan pola mata yang terdiri dari proses penskalaan (*scaling), grayscale* dan *tresholding.*
3. Tahap ketiga adalah integral proyeksi untuk mencari daerah lokasi dari mata.
4. Tahap keempat metode Haar Cascade Classifier untuk mendeteksi mata.

3.2.1.1 Kalibrasi

Pada analisis ini inputan yang digunakan adalah citra langsung yang dikalibrasi oleh *webcam* secara *real time* dan hasilnya berupa citra digital yang akan digunakan untuk tahap berikutnya. Berikut ini adalah proses kalibrasi citra realtime :



Gambar 3.2 Menentukan koordinat-koordinat citra

Misalkan koordinat citra A(3,2,1), kemudian konversi koordinat-koordinat citra ke dalam dunia nyata dengan matriks 3 x 4 seperti berikut :

$$(3,2,1,1)\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 1 & 1\end{bmatrix} = (u, v, t)$$

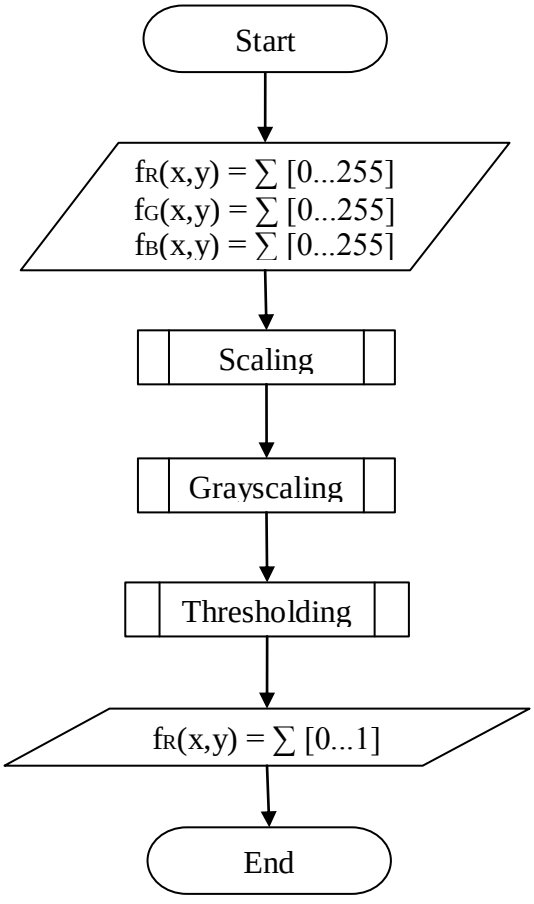
Jika dilanjutkan maka akan diperoleh :

u = 3.1 + 2.0 + 1.0 + 1 = 4
v = 3.0 + 2.1 + 1.3 + 1 = 6
t = 3.0 + 2.0 + 1.1 + 1 = 2

Maka koordinat citra adalah : (U,V) dimana U = 4/2 = 2 , V = 6/2 = 3, jadi koordinat citra pada computer yaitu (2,3).

3.2.1.2 Analisis Pengenalan Pola Mata

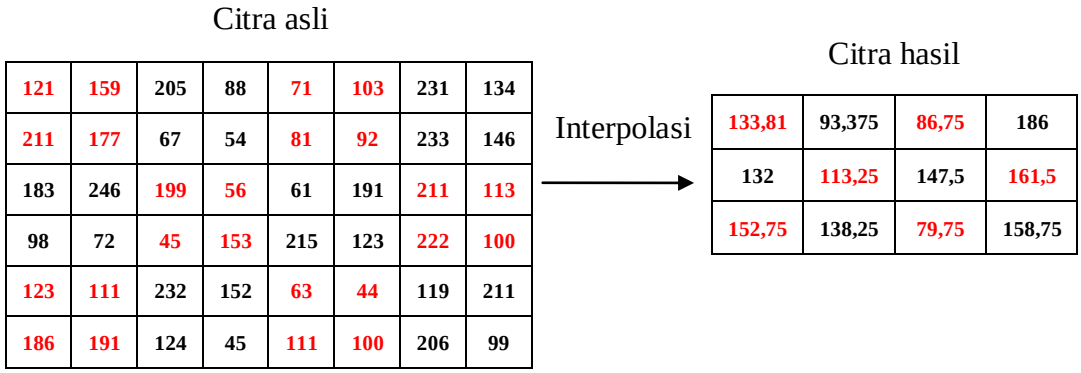
Dalam analisis processing image ini dibagi menjadi 3 tahap, tahap pertama yaitu penskalaan(*scaling*), tahap kedua proses *grayscaling*, dan tahap ketiga proses *tresholding.*



Gambar 3.3 Alur proses pengenalan pola mata

1. Tahap Penskalaan (*Scaling)*

Citra digital yang telah dikalibrasi secara *real time* oleh *webcam* akan diperkecil dengan menggunakan metode interpolasi. Metode ini menggunakan nilai rata – rata suatu region untuk mewakili region tersebut.



Gambar 3.4 Metode interpolasi untuk memperkecil gambar

Nilai piksel pada koordinat pada citra hasil interpolasi diperoleh dengan menghitung nilai rata – rata dari 4 nilai piksel pada citra asli, yaitu :

Tabel 3-1 Perhitungan nilai piksel hasil interpolasi

Nilai Piksel Citra Asli	Nilai Piksel Citra Hasil Interpolasi
(121+159+211+177) / 4	133,81
(205+88+67+54) / 4	93,375
(71+103+81+92) / 4	86,75
(231+134+233+146) / 4	186
(183+246+98+72) / 4	132
(199+56+45+153) / 4	113,25
(61+191+215+123) / 4	147,5
(211+113+222+100) / 4	161,5
(123+111+186+191) / 4	152,75
(232+152+124+45) / 4	138,25
(63+44+111+100) / 4	79,75
(119+211+206+99) / 4	158,75

Berikut ini adalah citra hasil interpolasi :

Gambar Asli



400 x 200 *pixel*

Gambar Hasil *Scaling*



200 x 100 *pixel*

Gambar 3.5 Penskalaan Citra Menggunakan Metode Interpolasi

2. Tahap *Grayscaling*

Citra digital yang telah melalui proses penskalaan kemudian diubah menjadi citra dua warna dengan proses *grayscaling*. Proses pengubahan citra RGB menjadi citra *grayscale* adalah sebagai berikut :

Misalkan suatu citra mata memiliki nilai : R = 152,75 G = 132 B = 133,81

Maka nilai grayscale dari citra tersebut dapat dihitung seperti di bawah ini :

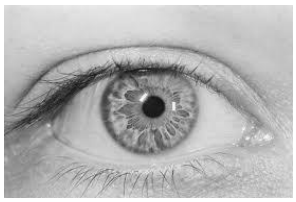
$$w = (0{,}2989 \times 152{,}75) + (0{,}5870 \times 132) + (0{,}1140 \times 133{,}81) = 138{,}395$$

Berikut ini adalah citra hasil *grayscaling* :

Gambar Asli



Gambar G*rayscale*



Gambar 3.6 Proses pengubahan citra RGB menjadi citra Grayscale

3. Tahap *Tresholding*

Selanjutnya adalah tahap *tresholding* yang digunakan untuk mengubah gambar hasil *grayscale* menjadi gambar biner. Nilai *Treshold* dihitung dengan membagi nilai hasil *grayscaling* pada tahap sebelumnya dengan nilai jumlah derajat keabuan (0 sampai 255 = 256) dibagi dengan 256 (nilai derajat keabuan yang diinginkan). Proses penghitungan nilai *treshold* untuk citra mata adalah sebagai berikut :

$$\mathbf{x = \frac{w}{b} = \frac{138{,}395}{1} = 138{,}395 \approx 138}$$

Dimana :

x = nilai pembanding *threshold*

w = nilai hasil *grayscaling*

b = 256/a (a = 256)

Untuk mengubah citra RGB menjadi citra biner menggunakan aturan sebagai berikut :

1. Jika nilai piksel citra ≥ x maka nilai piksel menjadi 1
2. Jika nilai piksel citra ≤ x maka nilai piksel menjadi 0

Berikut ini adalah proses pengubahan citra RGB menjadi citra biner :

Citra asli

121	159	205	88	71	103	231	134
211	177	67	54	81	92	233	146
183	246	199	56	61	191	211	113
98	72	45	153	215	123	222	100
123	111	232	152	63	44	119	211
186	191	124	45	111	100	206	99

Citra biner

0	1	1	0	0	0	1	0
1	1	0	0	0	0	1	1
1	1	1	0	0	1	1	0
0	0	0	0	1	1	1	0
0	0	1	1	0	0	0	1
1	1	0	0	0	0	1	0

Gambar 3.7 Proses pengubahan citra Grayscale menjadi citra Biner

Berikut ini adalah gambar hasil pengubahan citra grayscale menjadi citra biner :

Gambar Grayscale **Gambar Biner**



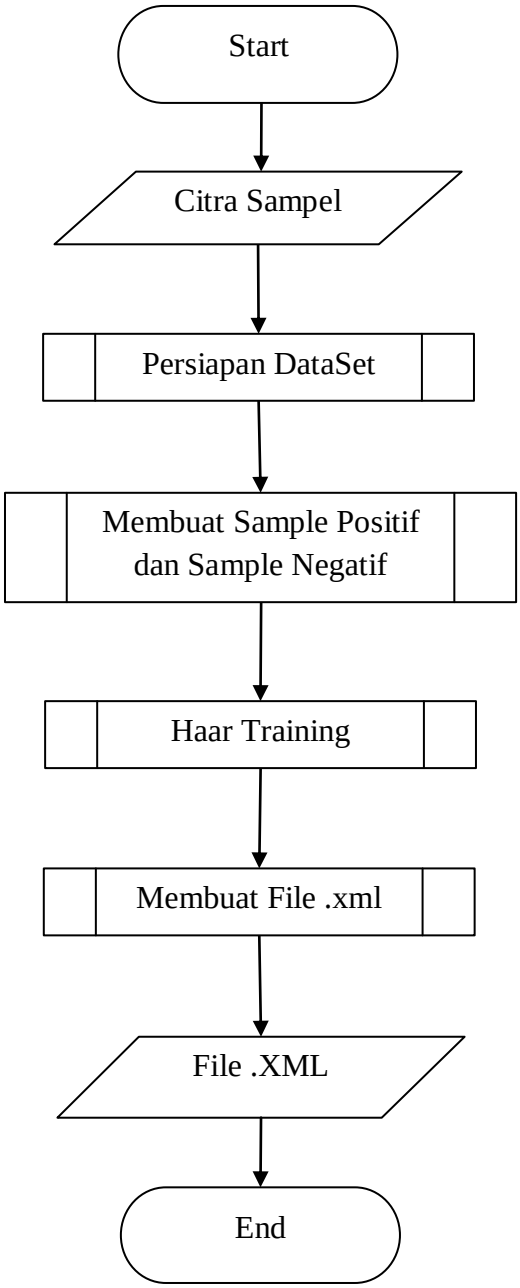
Gambar 3.8 Citra hasil thresholding

3.2.1.3 Deteksi Mata

Metode yang digunakan untuk mendeteksi mata adalah metode Haar Cascade Classifier. Metode ini merupakan metode yang menggunakan *statistical model* (*classifier*). Adapun tahapan-tahapan pada proses deteksi mata adalah sebagai berikut :

1. *Training* Data Pada *Haar*

File xml dibuat dengan suatu *training* yang dikenal dengan *Haar Training*. Proses *training* secara garis besar dapat dilihat dengan melalui bagan pada gambar di bawah. *Folder* aplikasi berada secara *default* di Program Files/OpenCV/bin/.



Gambar 3.9 Alur *Training* Data Pada Haar

Penjelasan alur dapat dijelaskan sebagai berikut :

a. Persiapan Data Set

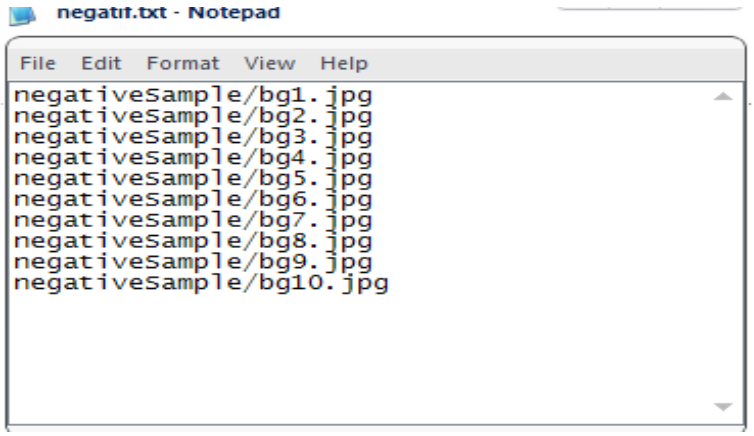
Data Set terdiri dari 2 buah *sample*, yaitu :

1. *Sample* positif, yaitu gambar yang mengandung obyek yang akan dideteksi. Jika kita menginginkan mata untuk dideteksi maka *sample* positif berisi gambar – gambar mata.
2. *Sample* negatif yaitu gambar yang tidak mengandung obyek yang akan dideteksi. Seperti gambar latar belakang, wajah dan sebagainya. Masukkan *sample* positif pada 1 direktori, misalnya *positiveSample/rawd*. *ata*Sedangkan *sample* negatif, dimasukkan pada */negativeSample*. *Sample minimal* 10 buah untuk mata dalam berbagai posisi. *File* gambar berupa *file* *.jpg.

b. Membuat *Sample* Negatif

Sampel negatif berisi gambar obyek selain obyek yang ingin dikenali. Resolusi untuk sampel negatif memiliki resolusi yang sama dengan resolusi kamera. Sampel negative berupa file text yang dibuat menggunakan create_list.bat pada folder /negativeSample untuk mencatatkan nama file sampel negatif pada file negatif.txt.

Contoh pencatatan nama file sampel negative pada file negative.txt :



Gambar 3.10 Pencatatan nama file pada sampel negatif

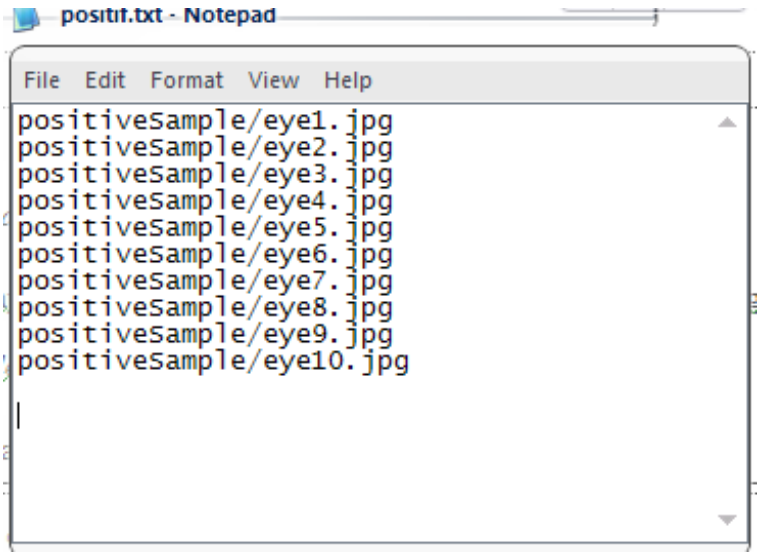
c. Membuat *Sample* Positif

Sampel positif berisi gambar obyek mata yang dibuat menggunakan *createsample utility*, data gambar dimasukkan ke dalam file positif.txt.

Berikut ini adalah berbagai pilihan *createsample utility :*

```
Usage: ./createsamples
  [-info <collection_file_name>]
  [-img <image_file_name>]
  [-vec <vec_file_name>]
  [-bg <background_file_name>]
  [-num <number_of_samples = 10>]
  [-bgcolor <background_color = 0>]
  [-inv] [-randinv] [-bgthresh
   <background_color_threshold = 70>]
  [-maxidev <max_intensity_deviation = 40>]
  [-maxxangle <max_x_rotation_angle = 1.100000>]
  [-maxyangle <max_y_rotation_angle = 1.100000>]
  [-maxzangle <max_z_rotation_angle = 0.500000>]
  [-show [<scale = 4.000000>]]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
```

Contoh pencatatan nama file sampel positif pada file positif.txt :



Gambar 3.11 Pencatatan nama file pada sampel positif

d. Haar Training

Sampel data yang telah dibuat dilatih menggunakan *haartraining utility*. Berikut ini adalah *listing* program pelatihan data dengan *haartraining utility* :

```
Usage: haartraining
  -data <dir_name>
  -vec <vec_file_name>
  -bg <background_file_name>
  [-npos <number_of_positive_samples = 10>]
  [-nneg <number_of_negative_samples = 20>]
  [-nstages <number_of_stages = 14>]
  [-nsplits <number_of_splits = 1>]
  [-mem <memory_in_MB = 200>]
  [-minhitrate <min_hit_rate = 0.995000>]
  [-maxfalsealarm <max_false_alarm_rate = 0.500000>]
  [-weighttrimming <weight_trimming = 0.950000>]
  [-mode <BASIC (default) | CORE | ALL>]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
  [-bt <DAB | RAB | LB | GAB (default)>]
  [-err <misclass (default) | gini | entropy>]
  [-maxtreesplits <max_number_of_splits_in_tree_cascade = 0>]
  [-minpos <min_number_of_positive_samples_per_cluster = 500>]
```

e. Membuat *file *.xml*

Setelah melakukan pelatihan data kemudian membuat *file *.xml*. Berikut ini adalah file .xml hasil *training* pada citra mata :

```
<opencv_storage>
<haarcascade_lefteye type_id="opencv-haar-classifier">
  <size>
    20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  8 12 3 8 -1.</_>
                <_>
                  8 16 3 4 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>0.027325896785021</threshold>
            <left_val>-0.9060062170028687</left_val>
            <right_node>1</right_node></_>
          <_>
            <!-- node 1 -->
            <feature>
              <rects>
                <_>
                  5 11 8 9 -1.</_>
                <_>
                  7 11 4 9 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-7.0568458177149296e-03</threshold>
            <left_val>0.9338570833206177</left_val>
            <right_val>-0.4585995972156525</right_val></_></_>
        <_>
```

2. Mendeteksi adanya wajah atau tidak pada gambar

Fungsi utama untuk mendeteksi wajah cukup sederhana, yaitu dengan menginisialisasi beberapa variabel, memasukkan gambar, mengubah gambar

menjadi *grayscale*, membuat fungsi *deteksi wajah* dan menampilkan rectangle di sekitar area citra wajah.

Gambar masukan diubah menjadi gambar *grayscale* seperti gambar dibawah ini *:*



Citra Masukan



Citra Grayscale

Gambar 3.12 Proses *Grayscale* pada citra wajah

Berikut ini adalah algoritma untuk mendeteksi wajah :

```
procedure deteksi wajah
{I.S : Citra realtime }
{F.S : Rectangle pada citra wajah }

kamus

img, face : Iplimage
ScaleFactor : double
MinNeighbors : integer
Minsize : integer

algoritma

  for  face -> total == 0
    begin
      face <- CvHaarClassifierCascade.FromFile("face.xml");
      if (faces->total == 0) then
        img.Rectangle(face.Rect, CvColor.Red, 2,
        LineType.AntiAlias);
      endif
    end
  endfor
endprocedure
```

3. Menentukan Region Of Interest

Region Of Interest adalah daerah persegi panjang pada gambar untuk memproses gambar lebih lanjut.

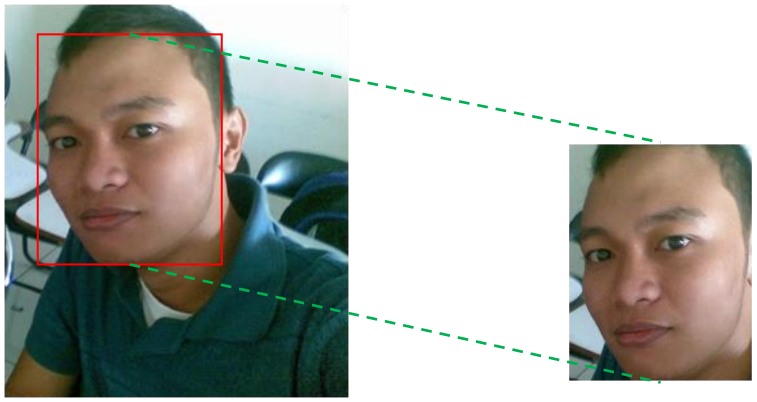


Image Sub Image

Gambar 3.13 Menentukan *Region Of Interest*

Berikut ini adalah algoritma untuk menentukan ROI pada citra wajah :

```
procedure Region Of Interest
{I.S : Rectangle pada citra yang terdeteksi }
{F.S : Citra di dalam rectangle }

kamus

faceroi, img, faces : IplImage
x : integer
y : integer

algoritma

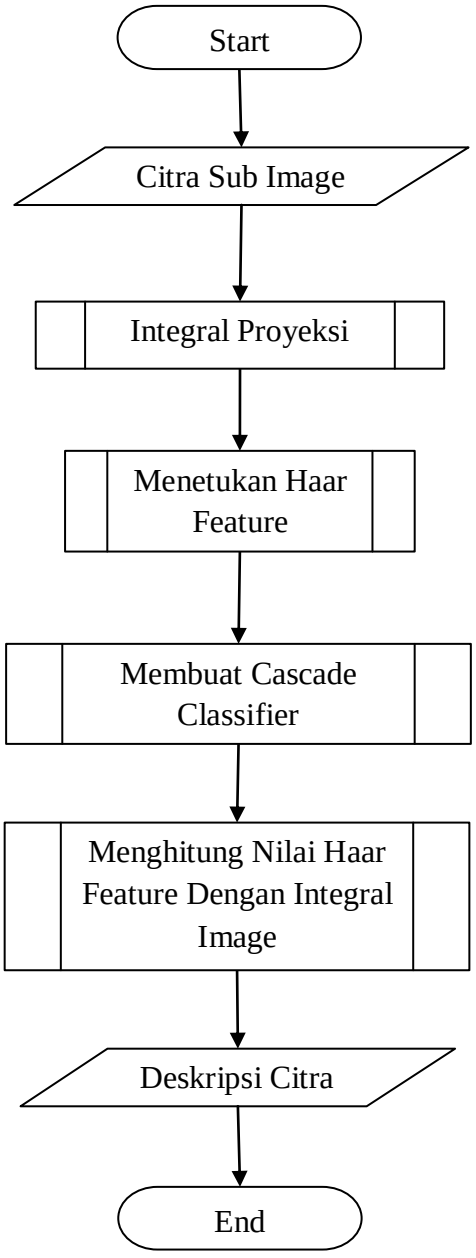
     if (img.Rectangle == face.Rect )then

        faceroi <- Cv.SetImageROI(face.Rect);
                   faceroi.CvtColor(lefteye,ColorConversion.Rg
                   bToGray);
                    face.Threshold(lefteye, x, y,
                   ThresholdType.Binary);

     endif
endprocedure
```

4. Sistem kerja metode *Haar Cascade Classifier*

Alur cara kerja metode Haar Cascade Classifier untuk mendeteksi mata adalah sebagai berikut :



Gambar 3.14 Alur Cara Kerja Metode *Haar Cascade Clasifier*

a. Integral Proyeksi

Setelah wajah terdeteksi, kemudian dilakukan proses integral proyeksi untuk menentukan daerah lokasi dari mata. Di sini diperkirakan lokasi mata 1/3 dari puncak wajah (beberapa pikel dari atas). Pencarian obyek dimulai pada sub window dengan ukuran skala 24x24 pada seluruh daerah gambar grayscale. Pencarian obyek diulangi kembali dengan ukuran sub window yang telah diskala ulang.

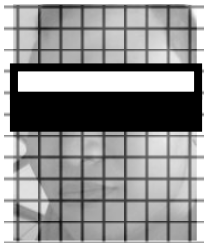
Gambar Grayscale **Perkiraan Daerah Lokasi Mata**



Gambar 3.15 Perkiraan Daerah Lokasi Mata

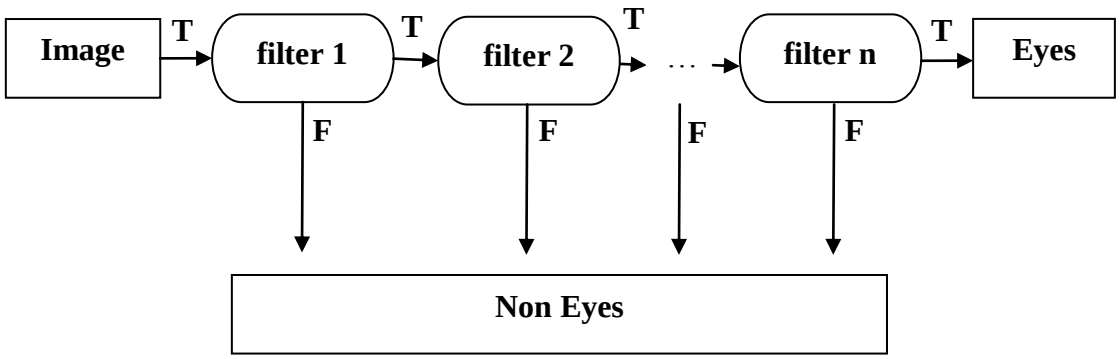
b. Haar Fitur

Setelah menentukan daerah lokasi dari mata kemudian dilakukan pencarian obyek mata. yaitu dengan cara mencari fitur-fitur yang memiliki tingkat pembeda yang tinggi. Hal ini dilakukan dengan mengevaluasi setiap fitur terhadap data latih dengan menggunakan nilai dari fitur tersebut. Fitur yang memiliki batas terbesar antara mata dan bukan mata dianggap sebagai fitur terbaik.



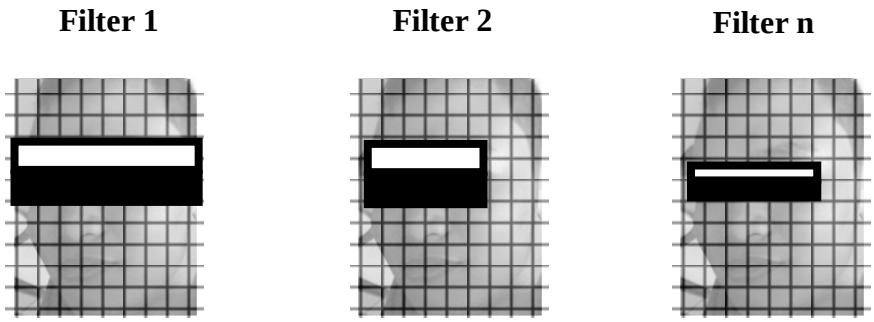
Gambar 3.16 Menentukan Haar fitur

Berikut ini adalah cara kerja algoritma *haar cascade classifier* :



Gambar 3.17 *Cascade Classifier*

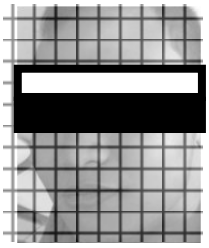
Pada klasifikasi tingkat pertama (filter 1), tiap subcitra akan diklasifikasi menggunakan satu fitur. Klasifikasi ini kira-kira akan menyisakan 50% subcitra untuk diklasifikasi di tahap kedua (filter 2). Seiring dengan bertambahnya tingkatan klasifikasi, maka diperlukan syarat yang lebih spesifik sehingga fitur yang digunakan menjadi lebih banyak. Jumlah subcitra yang lolos klasifikasi pun akan berkurang hingga mendekati citra yang ada pada sample (data .xml) atau hingga klasifikasi menyisakan 2% subcitra.



Gambar 3.18 Pencarian haar feature

d. *Integral Image*

Kotak Haar feature dapat dihitung menggunakan "*integral image*". Nilai dari *feature* dihitung menggunakan integral image pada gambar dibawah ini :



1	3	3	2
1	4	4	5
1	**7**	**5**	**2**
2	**3**	**4**	**6**
1	3	2	1
1	3	5	8
1	2	3	1
3	5	3	1

1	4	7	9
2	9	16	23
3	**17**	**29**	**38**
5	**22**	**38**	**53**
6	26	44	60
7	30	53	77
8	33	59	84
11	41	70	96

Gambar 3.19 Penghitungan Nilai Haar *Feature*

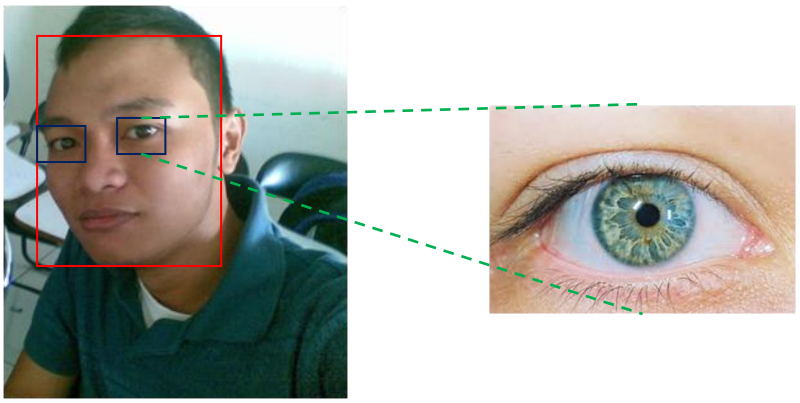
Dari perhitungan integral image diatas, maka diperolah nilai integral fitur hitam = { 5,22,38,53 }dan nilai integral fitur putih = { 3,17,29,38 }.

Maka nilai haar fitur tersebut adalah :

$$\mathbf{f(x) = (5 + 22 + 38 + 53) – (3 + 17 + 29 + 38) = 31}$$

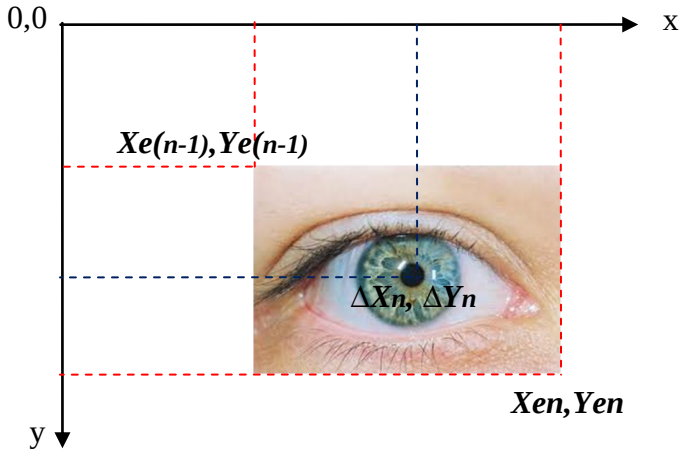
3.2.2 Analisis *Tracking* Mata

Tracking mata dilakukan untuk proses pergerakan *pointer mouse*. *Tracking* mata dilakukan menggunakan metode *eye tracking*. Sistem akan mencari koordinat *pointer* , kemudian nilai koordinat tersebut akan dijumlahkan dengan titik tengah koordinat mata. Titik tengah mata diambil dari citra yang ada di dalam *region of interest* di area mata. Berikut ini adalah citra mata yang dijadikan sebagai acuan untuk menentukan lokasi titik tengah mata :



Gambar 3.20 citra mata yang digunakan sebagai acuan

Berikut ini adalah gambar koordinat titik tengah mata :



Gambar 3.21 Menentukan koordinat titik tengah mata

Kemudian untuk posisi *pointer* dihitung menggunakan persamaan **2-17.** Perhitungan posisi pointer dapat dijabarkan seperti di bawah ini :

Misalkan koordinat titik tengah mata (300,500) , maka posisi pointer adalah (300 + 0,500 + 0) = 300 , 500.

3.2.3 Analisis Deteksi Kedipan

Pada tahap ini sistem akan mendeteksi kedipan mata pengguna untuk digunakan sebagai proses klik pada *pointer*. Deteksi kedipan dilakukan dengan cara memberikan nilai *grayscale* dari gambar mata dari tahap sebelumnya, kemudian dilakukan proses *tresholding* untuk mendapatkan citra biner, sehingga dapat membedakan perubahan piksel yang terjadi pada gambar.

Tahap berikutnya yang penting adalah operasi morfologi. Operasi morfologi digunakan untuk menghilangkan *noise* yang ada. Operasi morfologi yang digunakan adalah operasi erosi dan dilasi.

1. Erosi

   Erosi merupakan proses penghapusan titik-titik batas objek menjadi bagian dari latar, berdasarkan *structuring element* yang digunakan. Pada operasi ini, ukuran obyek diperkecil dengan mengikis sekeliling objek.

   Contoh operasi Erosi pada citra mata :

Citra asli

1	**1**	**0**	**0**	**0**	**0**	**1**	**1**
1	**0**	**1**	**1**	**1**	**1**	**0**	**1**
1	**1**	**0**	**0**	**0**	**0**	**1**	**1**
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**
1	**1**	**0**	**0**	**0**	**0**	**1**	**1**

Dierosi dengan

1	**1**
1	**1**

Hasil erosi

1	**1**	**1**	**0**	**0**	**1**	**1**	**1**
1	**1**	**0**	**1**	**1**	**0**	**1**	**1**
1	**1**	**1**	**0**	**0**	**1**	**1**	**1**
1	**1**	**0**	**0**	**0**	**0**	**1**	**1**
1	**1**	**1**	**0**	**0**	**1**	**1**	**1**

Gambar 3.22 Proses erosi pada citra mata

2. Dilasi

Dilasi merupakan proses penggabungan titik-titik latar menjadi bagian dari objek, berdasarkan *structuring element* yang digunakan. Proses ini adalah kebalikan dari erosi, yaitu merubah latar disekeliling objek menjadi bagian dari objek tersebut.

Citra asli

1	1	1	0	0	1	1	1
1	1	0	1	1	0	1	1
1	1	1	0	0	1	1	1
1	1	0	0	0	0	1	1
1	1	1	0	0	1	1	1

Didilasi dengan

1	1
1	1

Hasil dilasi

1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1
1	0	0	0	0	0	0	1
1	1	0	0	0	0	1	1

Gambar 3.23 Proses dilasi pada citra mata

3.3 Deskripsi Kebutuhan Sistem

Sebelum membangun sebuah sistem perlu dilakukan analisis kebutuhan sistem untuk menjamin bahwa sistem yang dibuat sesuai dengan kebutuhan pengguna dan layak untuk dikembangkan. Tahapan analisis kebutuhan sistem dapat dirinci menjadi beberapa tahap guna mempermudah proses analisis secara keseluruhan. Tahapan-tahapan ini sangat penting untuk menjamin keberhasilan pengembangan sistem secara keseluruhan.

3.3.1 Deskripsi Kebutuhan Data *Eksternal*

Kebutuhan data *eksternal* akan diuraikan secara rinci untuk keperluan perancangan perangkat lunak. Data *eksternal* yang digunakan pada perangkat lunak pengendali pointer yang akan dibangun yaitu :

1. Face.xml

File face.xml berisi data hasil dari *training* citra wajah yang digunakan untuk mendeteksi wajah. Adapun isi dari *file* face.xml adalah sebagai berikut :

```
<opencv_storage>
<haarcascade_ alt type_id="opencv-haar-classifier">
  <size>20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>2 7 14 4 -1.</_>
                <_>2 9 14 2 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>3.7895569112151861e-003</threshold>
            <left_val>-0.9294580221176148</left_val>
            <right_val>0.6411985158920288</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>1 2 18 4 -1.</_>
                <_>7 2 6 4 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>0.0120981102809310</threshold>
            <left_val>-0.7181009054183960</left_val>
            <right_val>0.4714100956916809</right_val></_></_>
        <_>
          <!-- tree 2 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>5 5 9 5 -1.</_>
                <_>8 5 3 5 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>1.2138449819758534e-003</threshold>
            <left_val>-0.7283161282539368</left_val>
            <right_val>0.3033069074153900</right_val></_></_></
            trees>
      <stage_threshold>-1.3442519903182983</stage_threshold>
      <arent>-1</parent>
      <next>-1</next></_>
    <_>
```

2. Lefteye.xml

File lefteye.xml berisi data hasil dari *training* citra mata kiri yang digunakan untuk mendeteksi mata kiri. Adapun isi dari *file* lefteye.xml adalah sebagai berikut :

```
<opencv_storage>
<haarcascade_lefteye type_id="opencv-haar-classifier">
  <size>
    20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  8 12 3 8 -1.</_>
                <_>
                  8 16 3 4 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>0.027325896785021</threshold>
            <left_val>-0.9060062170028687</left_val>
            <right_node>1</right_node></_>
          <_>
            <!-- node 1 -->
            <feature>
              <rects>
                <_>
                  5 11 8 9 -1.</_>
                <_>
                  7 11 4 9 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-7.0568458177149296e-03</threshold>
            <left_val>0.9338570833206177</left_val>
            <right_val>-0.4585995972156525</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  8 7 11 12 -1.</_>
                <_>
                  8 11 11 4 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-0.1253869980573654</threshold>
            <left_val>0.7246372103691101</left_val>
            <right_node>1</right_node></_>
          < >
```

3. Righteye.xml

File righteye.xml berisi data hasil dari *training* citra mata kanan yang digunakan untuk mendeteksi mata kanan. Adapun isi dari *file* righteye.xml adalah sebagai berikut :

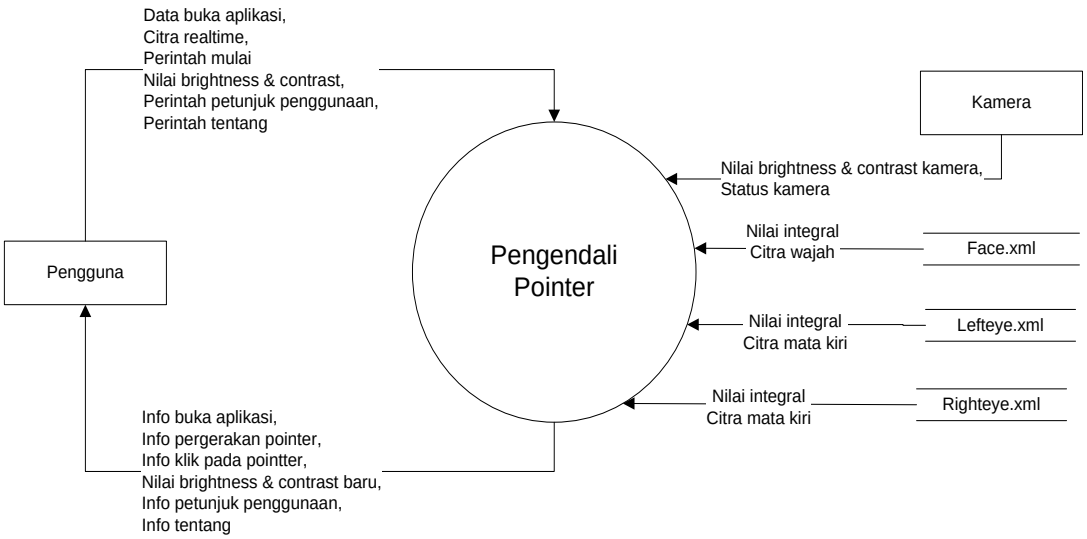
```
<opencv_storage>
<haarcascade_righteye type_id="opencv-haar-classifier">
  <size>
    20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  8 7 3 12 -1.</_>
                <_>
                  8 11 3 4 3.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>-0.0482105500996113</threshold>
            <left_node>1</left_node>
            <right_val>-0.8614044785499573</right_val></_>
          <_>
            <!-- node 1 -->
            <feature>
              <rects>
                <_>
                  8 7 8 3 -1.</_>
                <_>
                  10 9 4 3 2.</_></rects>
              <tilted>1</tilted></feature>
            <threshold>-0.0415761992335320</threshold>
            <left_val>0.9176905751228333</left_val>
            <right_val>-0.2128400951623917</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>
                  9 13 2 6 -1.</_>
                <_>
                  9 16 2 3 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>9.3528684228658676e-03</threshold>
            <left_val>-0.6978576779365540</left_val>
            <right_node>1</right_node></_>
          < >
```

3.3.2 Deskripsi Kebutuhan Fungsional

Kebutuhan fungsional dianalisis dengan memodelkan sistem. Pemodelan yang digunakan untuk memodelkan perangkat lunak pengendali *pointer* ini adalah pemodelan terstruktur. Perangkat lunak ini dimodelkan menggunakan DFD (*Data Flow Diagram*).*Tools* yang digunakan adalah Diagram Konteks, *Data Flow Diagram* dan Spesifikasi Proses yang dibuat menggunakan *Microsoft Visio 2007* sebagai perangkat lunak yang digunakan.

3.3.2.1 Diagram Konteks Pengendali Pointer

Diagram konteks adalah diagram yang menggambarkan masukan, proses dan keluaran secara umum yang terjadi pada sistem. Berikut ini adalah diagram konteks perangkat lunak pengendali pointer :



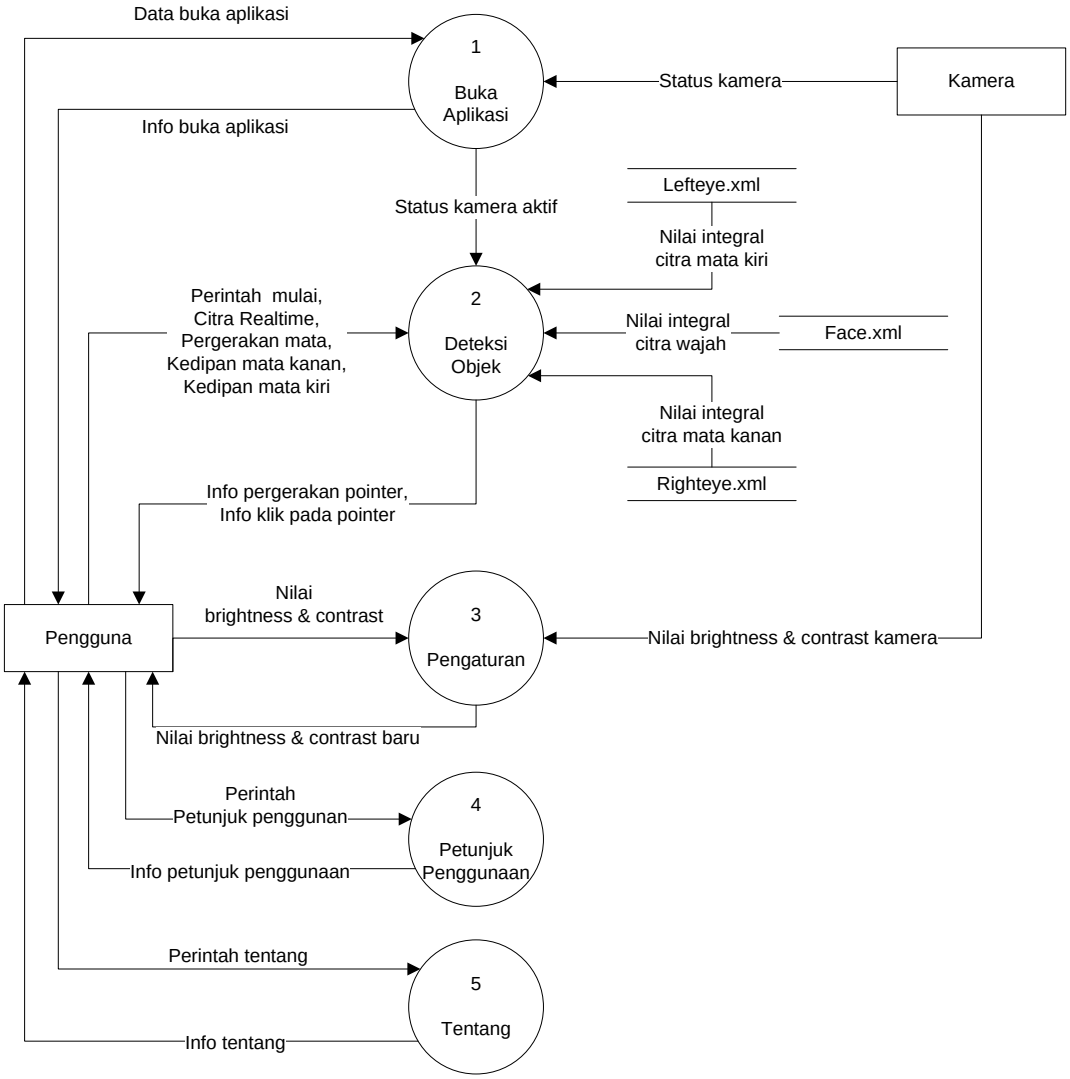
Gambar 3.24 Diagram Konteks Pengendali Pointer

3.3.2.2 *Data Flow Diagram (DFD)*

Data flow diagram merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil. DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau lingkungan fisik dimana data tersebut akan disimpan. Salah satu keuntungan DFD adalah memudahkan pemakai atau *user* yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan.

1. DFD Level 1 Pengendali Pointer

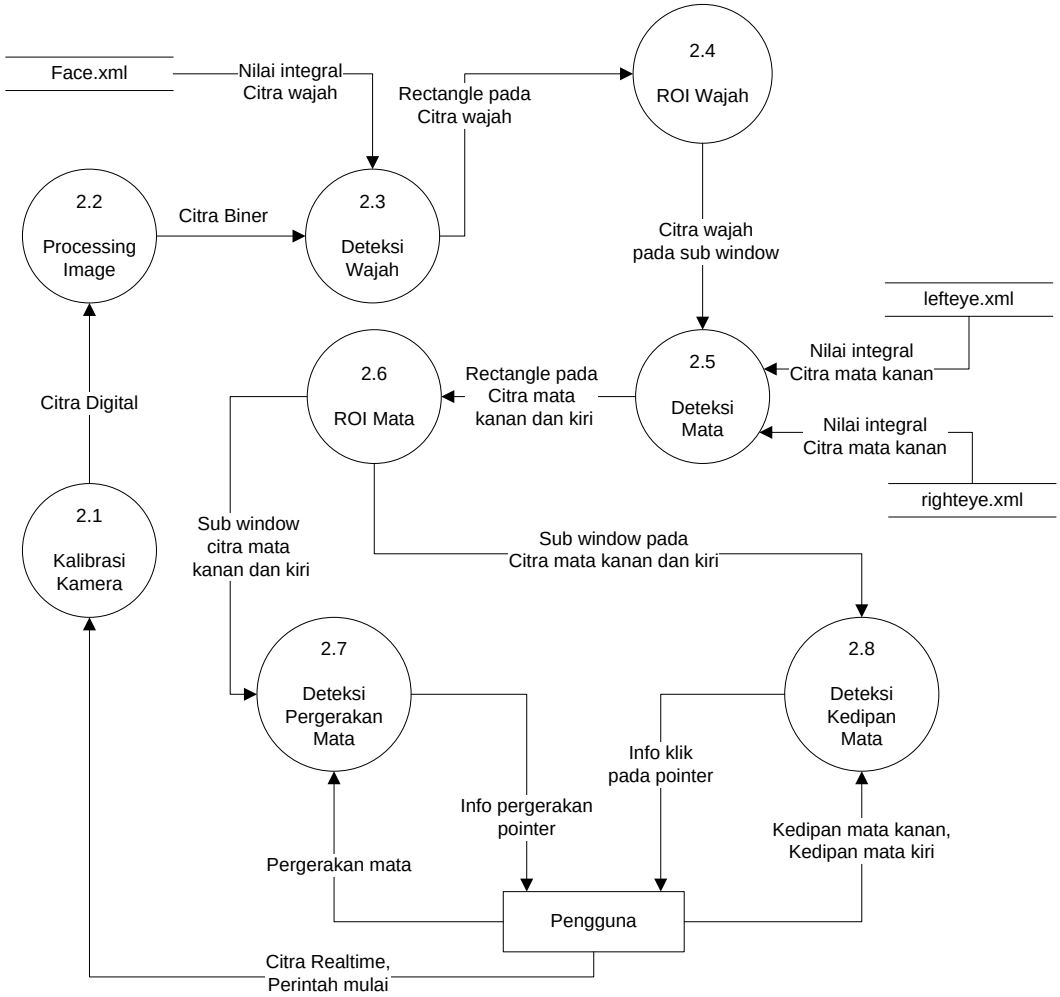
DFD Level 1 dibuat jika pada diagram konteks masih terdapat proses yang harus dijelaskan lebih rinci. Pada DFD Level 1 terdapat 5 proses yaitu buka aplikasi, deteksi objek, pengaturan, petunuk penggunaan dan tentang. DFD level 1 pengendali pointer digambarkan sebagai berikut :



Gambar 3.25 DFD Level 1 Pengendali Pointer

2. DFD Level 2 Proses Deteksi Objek

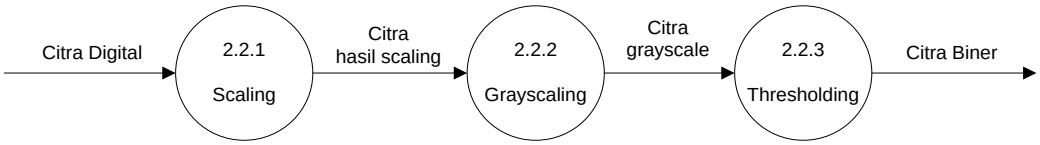
Pada DFD Level 2 proses deteksi objek terdapat 7 proses yaitu kalibrsi kamera, *processing image*, deteksi wajah, deteksi mata, *region of interest*, deteksi pergerakan mata dan deteksi kedipan mata. DFD Level 2 proses deteksi objek digambarkan sebagai berikut :



Gambar 3.26 DFD Level 2 Proses Deteksi Objek

3. DFD Level 3 Proses *Processing Image*

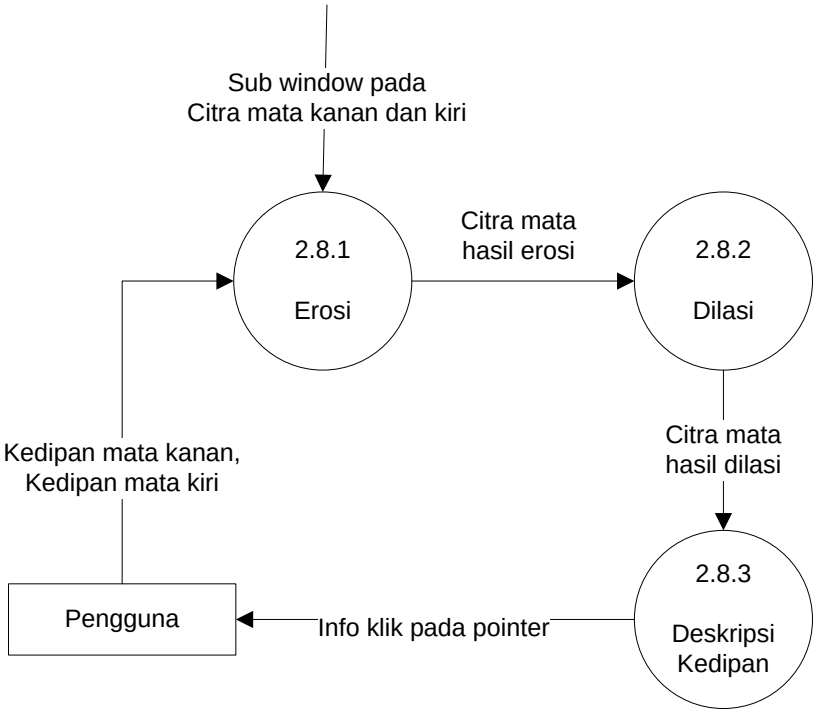
Pada DFD Level 2 proses *processing image* terdapat 3 proses yaitu *scaling*, *grayscaling* dan *thresholding*. DFD Level 2 proses *processing image* digambarkan sebagai berikut :



Gambar 3.27 DFD Level 2 Proses *Processing Image*

4. DFD Level 3 Proses Deteksi Kedipan Mata

Pada DFD Level 3 proses deteksi kedipan mata terdapat 3 proses yaitu erosi, dilasi, dan deskripsi citra. DFD Level 3 proses deteksi kedipan mata digambarkan sebagai berikut :



Gambar 3.28 DFD Level 3 Proses Deteksi Kedipan

3.3.2.3 Spesifikasi Proses

Tabel 3-2 Spesifikasi Proses

No	Proses	Keterangan
1	Nomor Proses	1
	Nama Proses	Buka Aplikasi
	Deskripsi	Memeriksa koneksi kamera dengan sistem
	Input	Kamera
	Output	Tampilan awal sistem
	Proses	Periksa ketersediaan kamera
	Logika Proses	1. Proses menerima request buka aplikasi dari pengguna 2. Proses menerima status kamera 3. Jika kamera aktif, proses akan menampilkan tampilan utama perangkat lunak. 4. Jika kamera tidak aktif, request menjalankan aplikasi ditolak, tampilan utama sistem tidak ditampilkan.
2	Nomor Proses	2
	Nama Proses	Deteksi Objek
	Deskripsi	Mendeteksi objek yang dikalibrasi oleh kamera
	Input	Citra realtime
	Output	Pergerakan pointer, klik kiri dan klik kanan pada pointer
	Proses	User melakukan pergerakan mata dan kedipan mata.
	Logika Proses	1. Proses menerima perintah mulai dan citra realtime dari pengguna. 2. Proses akan mendeteksi citra realtime tersebut dan mencocokkan nilai fitur pada citra tersebut dengan nilai fitur yang ada di data store 3. Jika terdeteksi wajah, proses akan membuat kotak persegi di sekitar area citra wajah, kemudian mendeteksi mata pada kotak persegi tersebut 4. Jika terdeteksi mata, proses akan membuat kotak persegi di area mata kanan dan mata kiri, kemudian mendeteksi pergerakan dan kedipan mata. 5. Jika pengguna melakukan pergerakan mata, maka proses akan melakukan pergerakan pada pointer sesuai dengan arah pandangan mata pengguna.

No	Proses	Keterangan
		6. Jika pengguna melakukan kedipan mata kiri maka proses akan melakukan klik kiri pada pointer 7. Jika pengguna melakukan klik kanan, maka proses akan melakukan klik kanan pada pointer
3	Nomor Proses	3
	Nama Proses	Pengaturan
	Deskripsi	Mengatur nilai brightness & contrast pada kamera
	Input	Nilai brightness & contrast
	Output	Nilai brightness & contrast baru
	Proses	Nilai brightness & contrast diubah sesuai dengan keinginan pengguna
	Logika Proses	1. Proses menerima nilai brightness & contrast pada kamera 2. Pengguna meng-input nilai brightness & contrast 3. Jika pengguna mengklik tombol simpan, nilai brightness & contrast yang di-input oleh pengguna akan tersimpan dalam sistem. 4. Jika pengguna mengklik tombol batal, nilai brightness & contrast kembali menjadi nilai default.
4	Nomor Proses	4
	Nama Proses	Petunjuk penggunaan
	Deskripsi	Menampilkan informasi petunjuk penggunaan aplikasi
	Input	Perintah petunjuk penggunaan
	Output	Informasi petunjuk penggunaan aplikasi
	Proses	Sistem menampilkan informasi petunjuk penggunaan aplikasi.
	Logika Proses	1. Proses menerima perintah petunjuk penggunaan dari pengguna 2. Proses akan menampilkan informasi petunjuk penggunaan aplikasi
5	Nomor Proses	5
	Nama Proses	Tentang
	Deskripsi	Menampilkan informasi tentang pembuat aplikasi
	Input	Perintah tentang
	Output	Informasi tentang pembuat aplikasi
	Proses	Sistem menampilkan informasi tentang pembuat aplikasi

No	Proses	Keterangan
	Logika Proses	1. Proses menerima perintah tentang dari pengguna 2. Proses akan menampilkan informasi tentang
6	Nomor Proses	2.1
	Nama Proses	Kalibrasi Kamera
	Deskripsi	Mengkalibrasi objek yang berada di depan kamera
	Input	Citra Realtime
	Output	Citra Digital
	Proses	Mengubah citra realtime menjadi citra digital
	Logika Proses	1. Proses menerima citra realtime 2. Proses akan mentransformasikan objek tersebut ke dalam dunia maya
7	Nomor Proses	2.2
	Nama Proses	Processing Image
	Deskripsi	Memproses citra yang telah dikalibrasi oleh kamera
	Input	Citra digital
	Output	Citra biner
	Proses	Melakukan scaling, grayscaling dan thresholding
	Logika Proses	1. Citra telah diubah menjadi citra digital 2. Mengubah penskalaan (*scaling*) untuk memperkecil ukuran citra 3. Mengubah citra menjadi citra grayscale 4. Mengubah citra grayscale menjadi citra biner
8	Nomor Proses	2.3
	Nama Proses	Deteksi Wajah
	Deskripsi	Mendeteksi wajah pada objek yang dikalibrasi
	Input	Citra biner
	Output	Citra wajah pada sub window
	Proses	Menampilkan kotak persegi panjang di sekitar citra wajah yang terdeteksi
	Logika Proses	1. Citra digital telah diubah menjadi citra biner 2. Proses menghitung nilai fitur pada citra biner. 3. Proses mencocokkan nilai fitur pada citra dengan fitur citra wajah pada file face.xml 4. Jika terdapat citra wajah, maka sistem menampilkan kotak persegi panjang disekitar area wajah
9	Nomor Proses	2.4
	Nama Proses	ROI wajah
	Deskripsi	Menentukan sub window pada citra wajah
	Input	Citra wajah pada kotak persegi panjang
	Output	Sub window

No	Proses	Keterangan
	Proses	Menentukan sub window dari kotak persegi panjang yang tampil pada citra wajah
	Logika Proses	1. Kotak persegi panjang pada citra wajah telah ditampilkan 2. Proses menentukan citra sub window, yaitu citra wajah yang ada di dalam kotak persegi panjang.
10	Nomor Proses	2.5
	Nama Proses	Deteksi Mata
	Deskripsi	Mendeteksi mata pada objek yang dikalibrasi
	Input	Citra wajah pada sub window
	Output	Citra mata yang terdeteksi pada sub window
	Proses	Menampilkan kotak persegi panjang di sekitar citra mata yang terdeteksi, untuk menentukan sub window pada citra mata
	Logika Proses	1. Proses menerima citra wajah pada sub window 2. Proses mengubah citra wajah yang terdeteksi menjadi citra integral 3. Proses memperkirakan posisi mata pada citra 4. Membuat fitur ke-1 sampai fitur ke-n menggunakan cascade classifier 5. Nilai tiap fitur dihitung menggunakan integral image. 6. Proses akan mencocokkan nilai fitur pada mata kanan maupun mata kiri pada file lefteye.xml dan righteye.xml 7. Jika terdapat citra mata yang sesuai dengan nilai fitur pada sample, maka sistem menampilkan kotak persegi panjang disekitar area mata
11	Nomor Proses	2.6
	Nama Proses	ROI Mata
	Deskripsi	Menentukan sub window pada citra mata
	Input	Citra mata pada kotak persegi panjang
	Output	Sub window pada citra mata
	Proses	Menentukan sub window dari kotak persegi panjang yang tampil pada citra mata
	Logika Proses	1. Kotak persegi panjang pada citra mata telah ditampilkan 2. Proses akan menentukan citra sub window, yaitu citra mata yang ada di dalam kotak persegi panjang.

No	Proses	Keterangan
12	Nomor Proses	2.7
	Nama Proses	Deteksi Pergerakan Mata
	Deskripsi	Melakukan pergerakan pointer
	Input	Citra mata pada sub window, pergerakan mata
	Output	Pergerakan pointer
	Proses	Mendeteksi pergerakan mata
	Logika Proses	1. Proses menerima citra mata pada sub window 2. Pengguna melakukan pergerakan mata 3. proses mendeteksi pergerakan mata, kemudian melakukan pergerakan pointer sesuai dengan arah pandangan mata pengguna.
12	Nomor Proses	2.8
	Nama Proses	Deteksi Kedipan Mata
	Deskripsi	Melakukan klik kiri dan klik kanan pada pointer
	Input	Citra mata pada sub window
	Output	Klik pada pointer
	Proses	Mendeteksi kedipan mata untuk proses klik kiri dan klik kanan pada pointer
	Logika Proses	1. Proses menerima citra mata pada sub window 2. Pengguna melakukan kedipan mata 3. Proses mendeteksi kedipan mata, kemudian melakukan klik pada pointer. 4. Jika pengguna mengedipkan mata kiri, sistem melakukan klik kiri pada pointer. 5. Jika pengguna mengedipkan mata kanan, sistem melakukan klik kanan pada pointer.
13	Nomor Proses	2.2.1
	Nama Proses	Scaling
	Deskripsi	Memperkecil ukuran citra yang masuk
	Input	Citra berukuran 1024 x 768
	Output	Citra berukuran 600 x 400
	Proses	Memperkecil pixel-pixel pada citra masukan
	Logika Proses	1. Proses menerima citra berukuran 1024 x 768 2. Proses memperkecil ukuran citra menjadi 600 x 400
14	Nomor Proses	2.2.2
	Nama Proses	Grayscale
	Deskripsi	Mengubah citra RGB menjadi citra grayscale
	Input	Citra RGB
	Output	Citra Grayscale
	Proses	Menghitung nilai grayscale dari citra RGB
	Logika Proses	1. Proses menerima citra RGB 2. Mengubah citra RGB menjadi citra grayscale

No	Proses	Keterangan
15	Nomor Proses	2.2.3
	Nama Proses	Thresholding
	Deskripsi	Mengubah citra grayscale menjadi citra biner
	Input	Citra grayscale
	Output	Citra biner
	Proses	Menghitung nilai biner dari citra grayscale
	Logika Proses	1. Citra RGB telah diubah menjadi citra grayscale 2. Proses mengubah citra grayscale menjadi citra biner
16	Nomor Proses	2.8.1
	Nama Proses	Erosi
	Deskripsi	Melakukan proses erosi pada citra mata
	Input	Citra mata pada sub window, kedipan mata kiri atau kanan
	Output	Citra hasil erosi
	Proses	Melakukan proses erosi untuk mengetahui adanya perubahan nilai piksel pada citra mata
	Logika Proses	1. Proses menerima perintah kedipan mata kanan atau kiri 2. Proses menerima citra mata pada sub window 3. Citra mata pada sub window diubah menjadi citra biner 4. Melakukan proses erosi pada citra mata
17	Nomor Proses	2.8.2
	Nama Proses	Dilasi
	Deskripsi	Melakukan proses dilasi pada citra mata hasil erosi
	Input	Citra mata hasil erosi
	Output	Citra mata hasil dilasi
	Proses	Melakukan proses dilasi untuk mengembalikan nilai piksel pada citra mata hasil erosi
	Logika Proses	1. Proses menerima citra hasil erosi, kemudian melakukan proses dilasi pada citra hasil erosi 2. Proses akan mengembalikan nilai piksel pada citra mata hasil erosi
18	Nomor Proses	2.8.3
	Nama Proses	Deskripsi Citra
	Deskripsi	Mendeskripsikan citra mata hasil proses dilasi
	Input	Citra mata hasil dilasi
	Output	Klik pada pointer
	Proses	Mendeskripsikan citra hasil proses dilasi

No	Proses	Keterangan
	Logika Proses	1. Proses menerima citra hasil dilasi kemudian memeriksa apakah terjadi perubahan piksel yang besar pada citra mata 2. Jika terjadi perubahan piksel pada citra mata kiri, proses akan melakukan proses klik kiri pada pointer. 3. Jika terjadi perubahan piksel pada citra mata kanan, proses akan melakukan proses klik kanan pada pointer.

3.3.3 Deskripsi kebutuhan *non*-fungsional

Analisis kebutuhan non-fungsional akan diuraikan secara rinci untuk keperluan perancangan parangkat lunak. Kebutuhan antarmuka *eksternal* tersebut meliputi antarmuka pemakai, antarmuka perangkat keras, dan antarmuka perangkat lunak.

3.3.3.1 Antarmuka Pemakai

Pengguna Perangkat lunak pengendali *pointer* yang akan dibangun ini adalah pengguna yang mengalami cacat fisik pada bagian motorik.

3.3.3.2 Antarmuka Perangkat Keras

Pembangunan perangkat lunak pengendali *pointer* ini memerlukan beberapa perangkat keras seperti :

a. **Komputer** dengan spesifikasi sebagai berikut :

1. Prosessor AMD Athlon 3200+ 2.01 GHz.
2. RAM 1 GB.

b. ***Webcam*** yang digunakan untuk proses deteksi wajah dan mata untuk pengenalan pola, *tracking* wajah dan mata untuk menggerakkan *pointer*, dan deteksi kedipan (*blink detection*) untuk proses klik pada pointer.

3.3.3.1 Antarmuka Perangkat Lunak

Sebelum menggunakan perangkat lunak pengendali *pointer* ini *user* harus terlebih dahulu menginstalasi aplikasi pendukung untuk mengaktifkan *webcam*.

Adapun perangkat lunak pendukung pembangunan perangkat lunak pengendali pointer ini antara lain :

1. Sistem Operasi Windows XP Profesional.
2. Microsoft Visual Visual Studio 2010, digunakan untuk pengkodean sistem.
3. Microsift Visio 2007, digunakan untuk memodelkan sistem
4. OpenCV 2.2, digunakan sebagai *library* tambahan untuk pengkodean sistem.

Adapun file extensi yang digunakan sebagai *library* dalam pembangunan perangkat lunak pengendali pointer ini antara lain :

1. **OpenCvSharp.dll,** digunakan untuk koneksi ke *library* OpenCV.
2. **cv210.dll,** digunakan untuk pengolahan gambar secara *real-time* dan menyediakan fungsi-fungsi untuk pengolahan data dan aplikasi multimedia. *Library* ini juga mencakup tipe data dan struktur data dalam aturan penamaan setiap operasi.

3. **cvaux210.dll,** digunakan untuk menjalankan program pada sistem operasi windows.
4. **cxcore210.dll,** digunakan untuk koneksi ke kamera atau perangkat lunak lain.
5. **highgui210.dll,** digunakan untuk menampilkan fungsi-fungsi pada aplikasi berbasis *form*.

3.3.4 Deskripsi Kebutuhan Perancangan Sistem

Tabel 3-3 Deskripsi Kebutuhan Perancangan Sistem

Kriteria	Tuntutan
Performansi	Harus dapat mendeteksi objek dengan jarak maksimal 40 cm dari kamera
	Perangkat lunak yang dibuat dapat dioperasikan pada komputer berspesifikasi minimal Intel Pentium 3 atau yang setara dengan jumlah RAM minimal 256 MB
Batasan memory	Maksimal jumlah memori yang digunakan oleh perangkat lunak tidak boleh melebihi 50 MB
Antar muka	Tulisan pesan dan menu perintah yang ditampilkan harus cukup jelas terbaca oleh pengguna dalam keadaan terang maupun gelap dengan menggunakan warna tulisan dan latar belakang yang tingkat kontrasnya tinggi dengan jenis huruf Arial berukuran minimal 12 poin.
	Modus grafis yang digunakan adalah VGA dengan resolusi minimal 640*480 dengan kedalaman warna 8 bit atau 256 warna
	Terdapat berbagai operasi dalam satu tampilan

3.3.5 Atribut Kualitas Perangkat Lunak

Tabel 3-4 Atribut Kualitas Perangkat Lunak

Kriteria Kualitas	Tuntutan Kualitas
Keandalan	Perangkat lunak dapat dijalankan pada komputer dengan spesifikasi rendah
	Perangkat lunak dapat digunakan untuk mengendalikan *pointer* pada semua aplikasi.
Ketersediaan	Bahasa pemrograman yang digunakan adalah bahasa yang kecil dan memungkinkan untuk dikembangkan.
	Menggunakan antarmuka perangkat keras yang sudah standar dan tersedia banyak dipasaran yaitu *webcam*.
Kepemindahan	Perangkat lunak dibuat dengan bahasa pemrograman yang dapat bekerja di berbagai arsitektur komputer.

3.3.6 Batasan Perancangan Sistem

Batasan-batasan pada tahap perancangan Perangkat Lunak Pengendali *Pointer* ini adalah :

1. Aplikasi ini hanya dapat mengendalikan *pointer* menggunakan mata, tidak dapat mengendalikan *pointer* dengan anggota tubuh yang lain.
2. Perancangan yang dikembangkan meliputi perancangan prosedural, kelas dan interface.

3.4 Perancangan sistem

3.4.1 Perancangan Lingkungan Implementasi

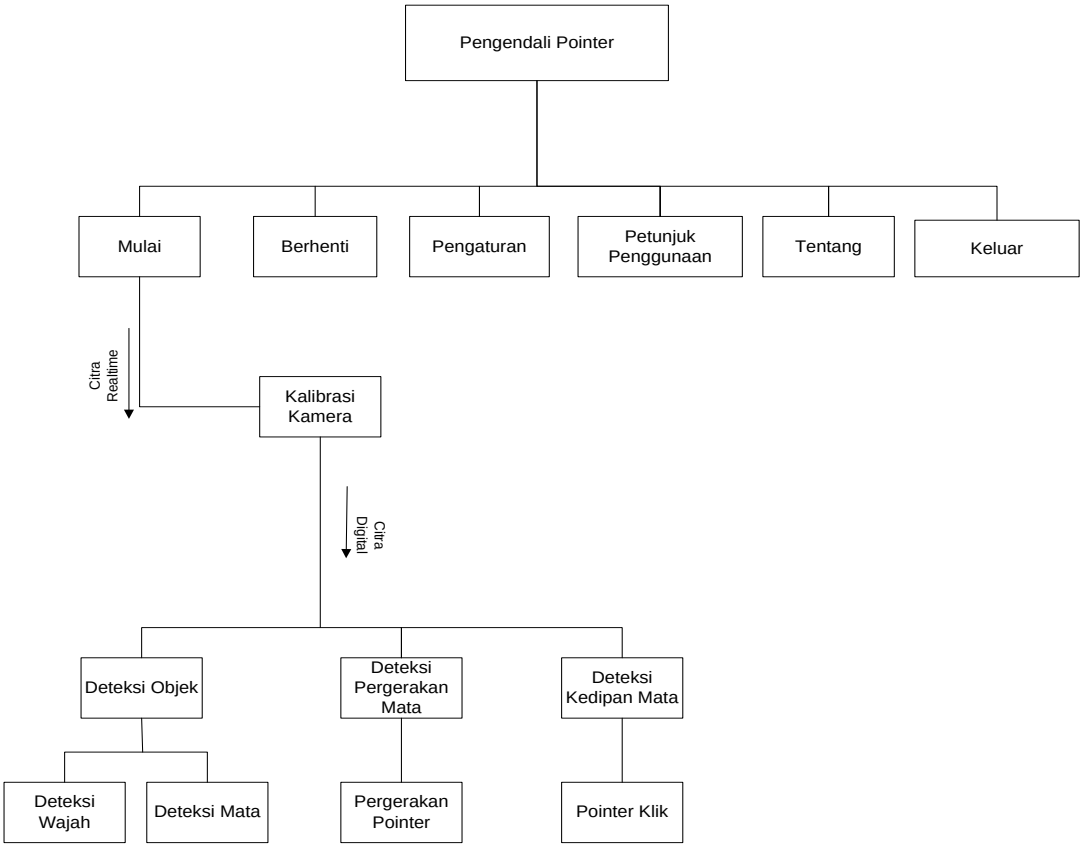
Spesifikasi lingkungan pengembangan perangkat lunak pengendali pointer dengan mata akan diuraikan pada tabel **3-5** dibawah ini. Spesifikasi ini meliputi Sistem Operasi, Development Tools dan Bahasa pemrograman yang dipakai.

Tabel 3-5 Perancangan Lingkungan Implementasi

Pengendali pointer dengan mata	Spesifikasi
Sistem Operasi	Windows XP
Development Tools	Microsoft Visual Studio 2010
Bahasa Pemrograman	C#
Library	OpenCV 2.2

3.4.2 Perancangan Struktur Program Pengendali Pointer

Struktur program merepresentasikan organisasi komponen program atau modul secara hirarki. Notasi yang digunakan merepresentasikan hirarki tersebut menggunakan diagram pohon. Fungsi pada struktur direpresentasikan dengan simbol persegi, input dan output digambarkan dengan anak panah. Adapun struktur program untuk perangkat lunak pengendali pointer dengan mata dapat dilhat pada gambar di bawah ini :



Gambar 3.29 Struktur Program Pengendali Pointer

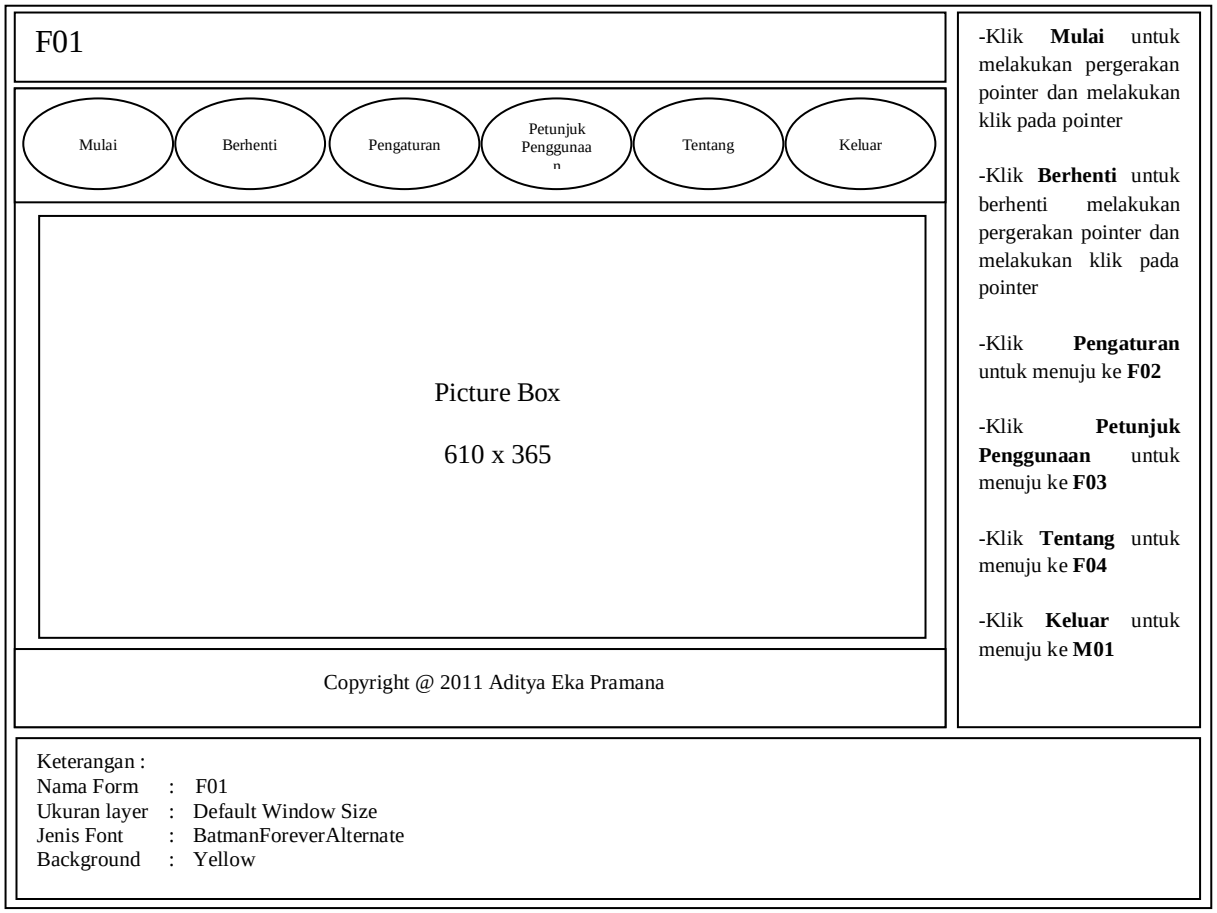
3.4.3 Perancangan Antarmuka

Perancangan antarmuka merupakan sebuah penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi. Adapun perancangan antarmuka perangkat lunak pengendali pointer adalah sebagai berikut :

3.4.3.1 Perancangan Antarmuka Form MenuUtama

1. Desain *Form* MenuUtama

Form menu utama merupakan *form* yang digunakan sebagai tampilan pada saat pengguna membuka aplikasi. Berikut ini adalah desain tampilan form aplikasi :



Gambar 3.30 Perancangan Antarmuka MenuUtama

2. Deskripsi Objek *Form* Menu Utama

Berikut ini adalah deskripsi objek *form* MenuUtama:

Tabel 3-6 Deskripsi objek form MenuUtama

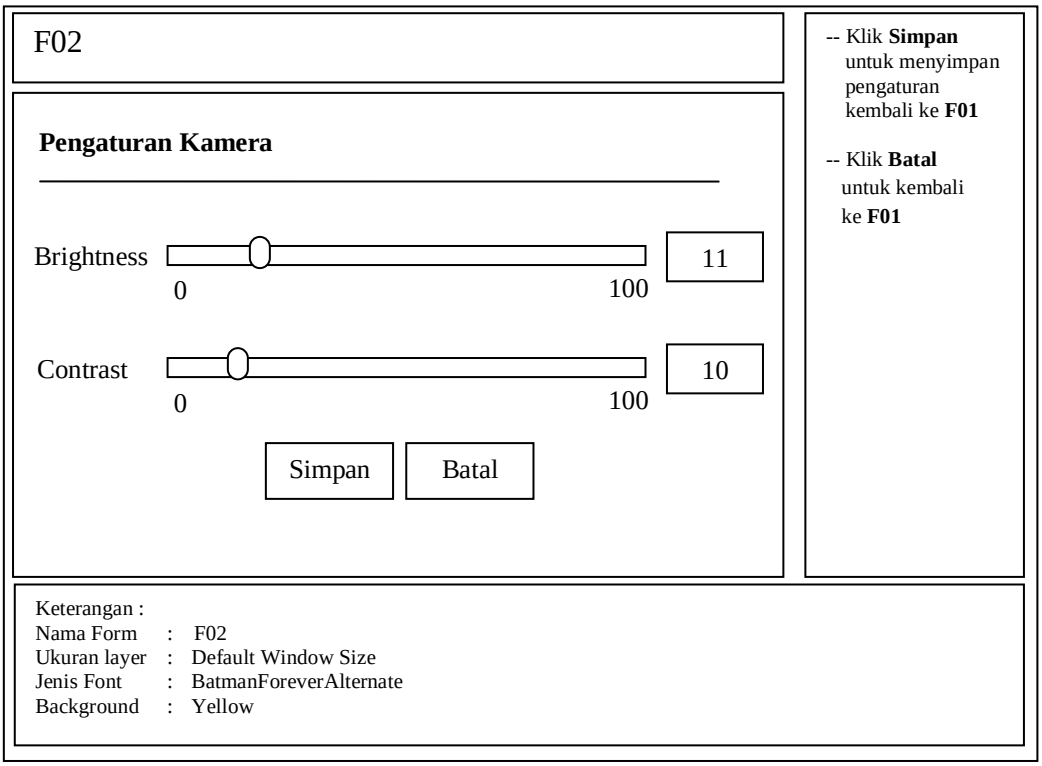
Objek	Jenis	Keterangan
Mulai	Button	Memulai mendeteksi
Berhenti	Button	Berhenti mendeteksi

Objek	Jenis	Keterangan
Pengaturan	Button	Konfigurasi sistem (F02)
Petunjuk penggunaan	Button	Petunjuk Penggunaan (F03)
Tentang	Button	Link ke form Tentang (F04)
Keluar	Button	Keluar Aplikasi (F05)
Picture Box	PicturBox	Tampilan objek yang dideteksi

3.4.3.1 Perancangan Antarmuka *Form* Pengaturan

1. Desain *Form* Pengaturan

Form pengaturan merupakan *form* yang digunakan untuk mengkonfigurasi pengaturan suara dan pengaturan kamera. Berikut ini adalah desain tampilan form pengaturan :



Gambar 3.31 Perancangan Antarmuka Pengaturan

2. Deskripsi objek *Form* Pengaturan

Berikut ini adalah deskripsi objek *form* pengaturan :

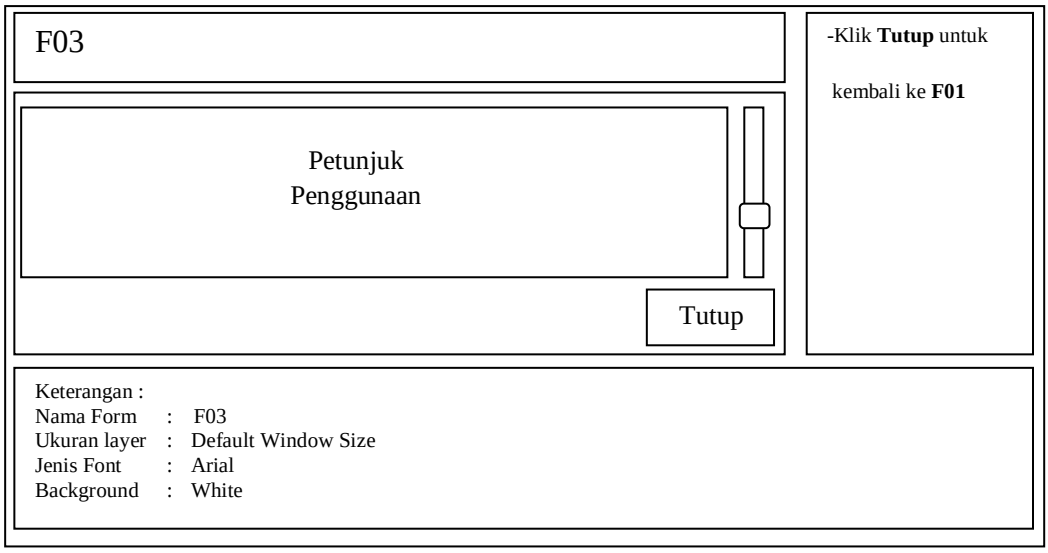
Tabel 3-7 Deskripsi objek form pengaturan

Objek	Jenis	Keterangan
Brightness	Control Box	Mengatur Brightness
Contrast	Control Box	Mengatur Contrast
Simpan	Button	Menyimpan pengaturan dan kembali ke (F01)
Batal	Button	Kembali ke (F01)

3.4.3.1 Perancangan Antarmuka Form Petunjuk

1. Desain *Form* Petunjuk

Form petunjuk penggunaan merupakan *form* yang digunakan untuk menampilkan petunjuk penggunaan aplikasi. Berikut ini adalah desain tampilan form petunjuk penggunaan :



Gambar 3.32 Perancangan Antarmuka Petunjuk Penggunaan

2. Deskripsi objek *Form* Petunjuk

Berikut ini adalah deskripsi objek *form* petunjuk penggunaan :

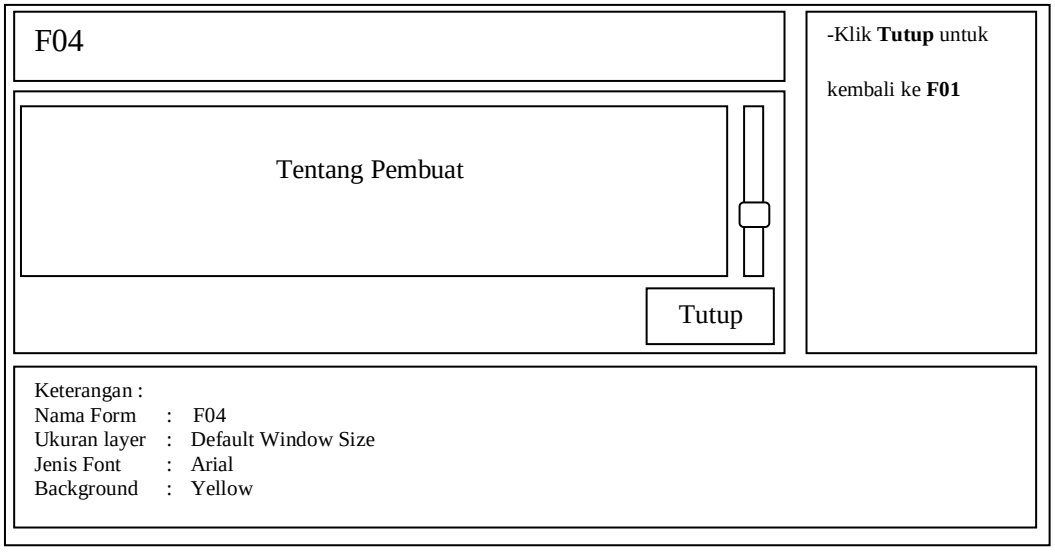
Tabel 3-8 Deskripsi objek form petunjuk

Objek	Jenis	Keterangan
Petunjuk Penggunaan	Pdf	Deskripsi petunjuk penggunaan sistem
Tutup	Button	Kembali ke (F01)

3.4.3.1 Perancangan Antarmuka Form Tentang

1. Desain *Form* Tentang

Form tentang merupakan *form* yang digunakan untuk menampilkan tentang pembuat aplikasi. Berikut ini adalah desain tampilan form tentang :



Gambar 3.33 Perancangan Antarmuka Tentang

2. Deskripsi objek *Form* Tentang

Berikut ini adalah deskripsi objek *form* tentang :

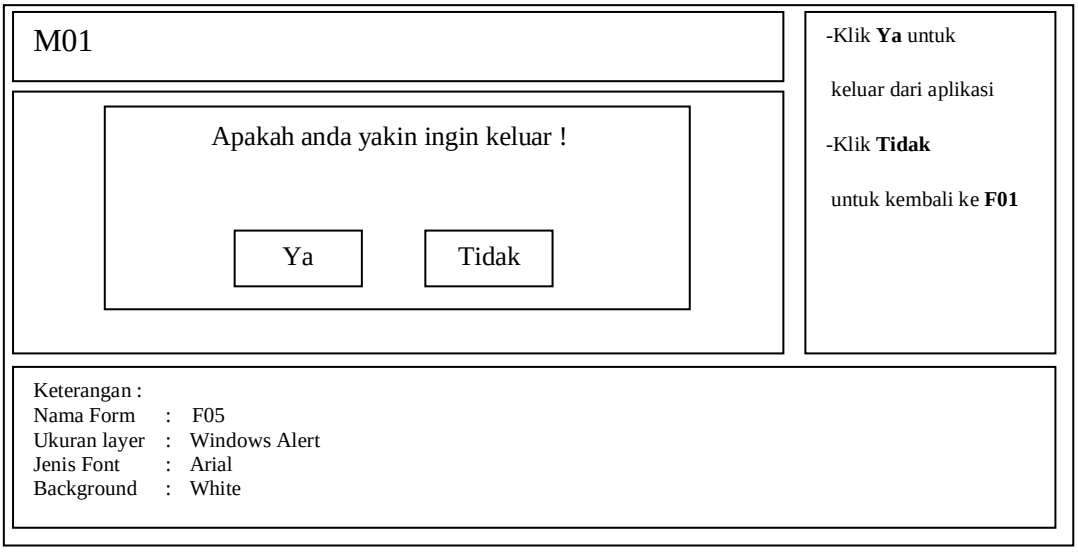
Tabel 3-9 Deskripsi objek form tentang

Objek	Jenis	Keterangan
Tentang Pembuat	Memo	Deskripsi mengenai pembuat aplikasi
Tutup	Button	Kembali ke (F01)

3.4.3.1 Perancangan Antarmuka Pesan Keluar Aplikasi

1. Desain Pesan Keluar Aplikasi

Pesan keluar aplikasi digunakan untuk konfirmasi keluar dari aplikasi. Berikut ini adalah desain tampilan pesan keluar aplikasi :



Gambar 3.34 Perancangan Antarmuka Pesan Keluar Aplikasi

2. Deskripsi objek Pesan Keluar Aplikasi

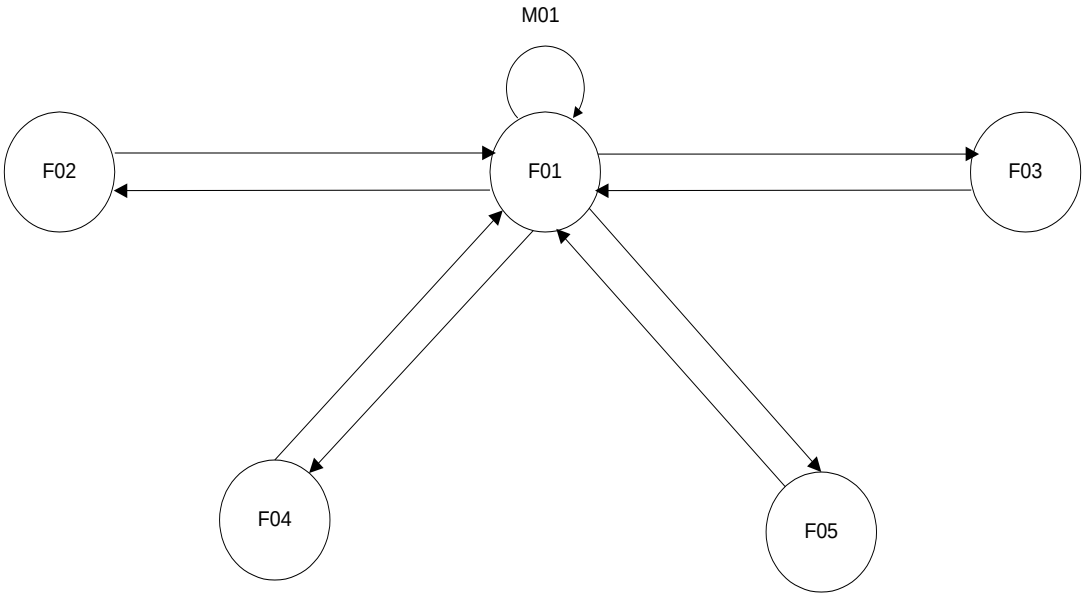
Berikut ini adalah deskripsi objek pesan keluar aplikasi :

Tabel 3-10 Deskripsi objek Pesan Keluar Aplikasi

Objek	Jenis	Keterangan
Ya	Button	Keluar dari aplikasi
Tidak	Button	Kembali ke (F01)

3.5 Jaringan Semantik

Berikut ini adalah gambar jaringan semantik yang menggambarkan hubungan antar modul perangkat lunak pengendali pointer.



Gambar 3.35 Jaringan Semantik