BAB III

ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis

III.1.1 Analisis Masalah

Permasalahan pada penelitian ini adalah sulitnya membangun sistem pengontrol *mouse pointer* komputer dengan pergerakan tangan manusia menggunakan *web-cam*. Hal ini dikarenakan sulitnya mendeteksi letak kontur tangan, *optical-flow* dan juga proses komputasi matriks dan histogram yang ada dalam sistem dikarenakan harus dibuat dengan memori yang terbatas sedangkan angka yang berada dalam matriks dan histogram sangat banyak dan terjadi perubahan nilai yang cukup besar dan cepat dalam prosesnya.

III.1.2 Analisis Kebutuhan Data

Sistem yang dibangun adalah sistem pengontrol *mouse pointer* komputer dengan pergerakan tangan manusia menggunakan *web-cam*. Suatu sistem yang melukakan kontrol atau kendali terhadap mouse pointer dengan pergerakan tangan manusia menggunakan *web-cam* dengan mengimplementasikan algoritma *lucas-kanade*, sehingga membutuhkan input berupa citra yang diambil secara *real-time* menggunakan alat input *web-cam* dan citra tersebut merupakan kumpulan matriks yang nantinya akan diolah lebih lanjut dalam sistem untuk proses identifikasi objek tangan manusia guna mengontrol *mouse pointer* komputer.

III.1.3 Analisis Algoritma

Dalam penelitian ini digunakan dua metode, akan tetapi metode yang paling utama adalah algoritma *pyramidal lucas kanade*. Algoritma *pyramidal lucas kanade* yang digunakan dalam penelitian ini adalah untuk mengetahui pergerakan *vector* yang terjadi dalam setiap perubahan frame pada video *real-time* yang didapatkan dari *web-cam*. Sedangkan metode kedua adalah *convexity defects* dimana metode ini berfungsi sebagai pengenal dari pola tangan yang diambil berdasarkan warna kulit dan menghitung jumlah jari yang terdeteksi oleh *convexity defects*.

Bukan hanya dalam *openCV* tetapi hampir disemua *tools* untuk pengembangan teknologi *computer vision* dibutuhkan proses komputasi sebelum diolah lebih lanjut dalam algoritma *pyramidal lucas kanade*. Adapun beberapa proses komputasi diperlukan dalam implementasi algoritma *pyramidal lucas kanade* adalah konversi citra menjadi *grayscale,* membuat citra *threshold,* metode *"good feature to track", find corner,* barulah implementasi algoritma *pyramidal lucas kanade.*

Dalam penggunaana *library* OpenCV dengan menggunakan bahasa pemrogramman C++, diperlukan beberapa *library* yang di include untuk menghubungkan antara sistem yang dibangun dengan *library* openCV. *Library* yang perlu di include ke dalam sistem adalah sebagai berikut :

1. opencv_imgproc
2. opencv_calib3d

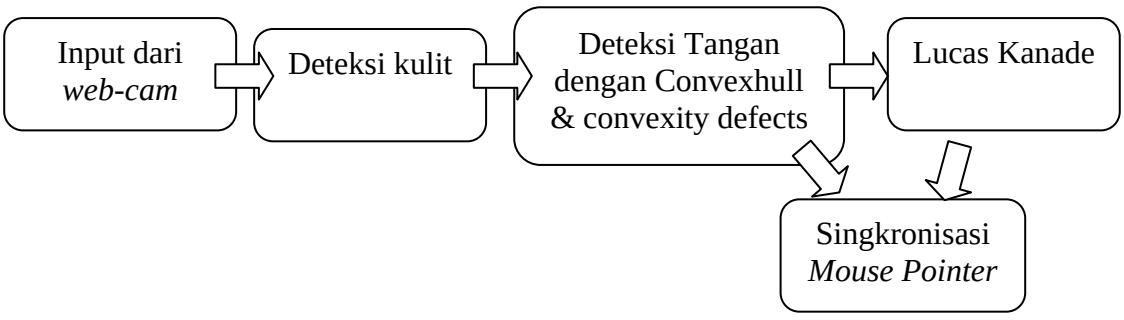
3. opencv_video
4. opencv_features2d
5. opencv_ml
6. opencv_highgui
7. opencv_objdetect
8. opencv_contrib
9. opencv_legacy

Sedangkan Header di perlukan untuk menggunakan openCV dalam sistem adalah :

1. cv.h
2. highgui.h
3. cvaux.h

Dalam implementasi teknik handtracking tentunya tidak dapat langsung diimplementasikan begitu saja, dibutuhkan *pre-processing* sebelum mengimplementasikan metoda *tracking and motion* dengan menggunakan openCV.

Berikut garis besar proses yang terjadi di dalam sistem:



Gambar III.1 *Flow* Proses *handtracking* untuk kontrol mouse pointer

Berikut adalah penjelasan langkah yang ada pada gambar III.1 yang merupakan flow proses dari sistem yang dibangun:

1. Input citra melalu *web-cam* dan menghasilkan citra 3 channel
2. Konversi warna menjadi YCrCb untuk Mendeteksi kulit
3. Melakukan deteksi tangan dan jari yang menghasilkan contour tangan
4. *Contour* tangan diolah untuk mendapatkan pergerakan atau *motion* dengan algoritma *pyramidal lucas-kanade*
5. Hasil dari deteksi tangan dengan *convexity defects* dan *motion* dengan *pyramidal lucas-kanade* disingkronisasi dengan pointer mouse menggunakan lib-X11.

III.1.3.1 Segmentasi Citra Tangan

Pada tahap deteksi tangan dengan metode *convex hull* dan *convexity defects* terdapat proses-proses tertentu dalam implementasi metode tersebut, diantaranya metode *convex hull* memerlukan citra 1 bit dan merupakan *contour* hasil *threshold*. Pembuatan citra hasil *threshold* dilakukan dengan melakukan segmentasi menggunakan metode *cvInRange* untuk melakukan segmentasi dengan nilai maximum dan minimum warna kulit dengan format YCrCb, lalu dikenakan *cvDilate* untuk melakukan segmentasi pada citra dengan hasil *cvRangeS* sebelumnya dan *cvErode* untuk memperjelas atau mempertegas hasil *threshold* yang di dapat.

Berikut adalah method *cvInrangeS* dan *cvErode* yang dipakai dalam proses segmentasi untuk mendapatkan *threshold* warna kulit :

```
void cvInRangeS(
     const CvArr* src,
     CvScalar lower,
     CvScalar upper,
     CvArr* dst
);
```

Metode CvInrage ini menggunakan *libopencv_imgproc.so*, dengan menggunakan *header cv.h* yang di dalamnya teradapat *header imgproc_c.h.* Parameter *src* merupakan *source* atau sumber citra yang ingin diproses. Parameter *lower* dan *upper* merupakan nilai minimum (*lower*) dan maksimum (*upper*) dari warna yang ingin dilihat *range* nilainya. Parameter *dst* merupakan citra hasil dari segmentasi warna yang dilakukan.

Nilai warna yang dipakai dalam YCrCb adalah :

Nilai minimum : Y = 0; Cr = 131; Cb= 80.

Nilai maksimum : Y= 245; Cr = 185; Cb= 135.

Lalu dilakukan dilatasi atau pemisahan antara *background* dan *foreground* dengan fungsi *cvDilate.* Berikut rutin yang dipakai dalam openCV:

```
void cvDilate(
     IplImage* src,
     IplImage* dst,
     IplConvKernel* B = NULL,
     int iterations = 1
     );
```

Dengan parameter yang hampir sama pada setiap fungsi openCV yaitu parameter *src* sebagai citra sumber dan *dst* sebagai variabel penyimpan citra hasil komputasi. Metode cvDilate merupakan metode yang menggunakan *library libopencv_imgproc.so* dan diletakan pada header cv.h.

Setelah itu dilakukan penegasan *threshold* yang didapat dari hasil segmentasi dengan menggunakan *cvErode*, berikut adalah rutin dari cvErode :

```
void cvErode(
     IplImage* src,
     IplImage* dst,
     IplConvKernel* B = NULL,
     int iterations = 1
     );
```

Metode cvErode terdapat pada *library libopencv_imgproc.so*, dan di *include* pada *haeder cv.h*.

Pada gambar III.2 merupakan contoh hasil segmentasi menggunakan nilai maksimum dan minimum dari warna kulit.



Gambar III.2 Contoh Hasil Segmentasi

III.1.3.2 Pencarian Countur Tangan

Setelah proses di atas barulah dapat dilakukan proses pencarian *contour* dengan menggunakan metode *Finding Contour* dan mencari kontur terbesar atau *biggest contour*. Dalam tahap ini digunakan rutin openCV yaitu *cvFindContours*. Namun sebelumnya kita harus menyiapkan tempat penampung untuk *contour* yang berbentuk *sequence* dari setiap titika *contour*, openCV menyediakan *struct* untuk *sequence* dari *contour* yaitu tipe data *CvSeq*, berikut adalah struktur dari *CvSeq*:

```
typedef struct CvSeq {
int flags;          //semacam flags
int header_size;    //size dari sequence header
CvSeq* h_prev;      //sequence sebelumnya
CvSeq* h_next;      // sequence selanjutnya
CvSeq* v_prev;      //2 sequence sebelumnya
CvSeq* v_next;      //2 sequence selanjutnya
int total;          //jumlah total dari element
int elem_size;      //ukuran dari elemen sequence
                    //dalam byte
char* block_max;    // maksima bound yang terakhir
                    // per blok
char* ptr;          // pointer yang sedang dipakai
int delta_elems;    // banyak elements yang
                    //dialokasikan
                    // ketika sequence bertambah
CvMemStorage* storage;   // tempat memory sequence
CvSeqBlock* free_blocks; // blok list yang bebas
CvSeqBlock* first; // pointer yang menunjuk ke
                    //pointer pertama dari
                    //sequence block
}
```

Struct tersebut nantinya menyimpan informasi mengenai contour dari suatu citra hasil *threshold* yang nantinya berisi *contour* tangan. *Struct* seq dalam

opencv ada pada *library libopencv_type.so* diletakan dalam header cv.h yang telah di *include header type_c.h.*

Sedangkan rutin untuk mencari contour yang terdapat pada suatu citra adalah sebagai berikut :

```
int cvFindContours(
          IplImage* img,
          CvMemStorage* storage,
          CvSeq** firstContour,
          int headerSize = sizeof(CvContour),
     CvContourRetrievalMode mode = CV_RETR_LIST,
CvChainApproxMethod method= CV_CHAIN_APPROX_SIMPLE
);
```

Metode FindContours pada opencv ada pada *library* libopencv*_imgproc.so* diletakan pada header cv.h yang telah di *include hader imgproc_c.h.* Pada gambar III.3 merupakan contoh hasil ekstraksi contour yang didapat setelah melakukan segmentasi dari warna kulit.



Gambar III.3 Contoh hasil pencarian contour

III.1.3.3 Convex Hull dan Convexity Defects dari Tangan

Setelah mendapatkan *contour* dari tangan maka selanjutnya dilakukan pendeteksian *hull* dan *defects* dari *contour* yang sudah didapat dengan metode

convex hull untuk mendeteksi bidang tangan dan *convexity defects* yang nantinya titik hasil dari *convexity defects* digunakan untuk komputasi menghitung jari.

Adapun rutin *convex hull* dalam openCV adalah sebagai berikut :

```
CvSeq* cvConvexHull2(
          const CvArr* input,
          void* hull_storage = NULL,
          int orientation = CV_CLOCKWISE,
          int return_points = 0
     );
```

Parameter *input* dalam *convex hull*merupakan contour hasil komputasi sebelumnya. Parameter *hull_storage* merupakan tampat alokasi sequence hasil komputasi *convex hull* sedangkan *orientation* bernilai default dengan konstanta *CV_CLOCKWISE* yang bernilai 1 yang artinya proses komputasi searah jarum jam. Dan tipe data dari *return value convex hull* merupakan sequence hasil dari komputasi artinya contour yang sudah menunjukan lekuk dari tangan manusia. Metode *ConvexHull* ada pada *library libopencv_imgproc.so* dan terdapat pada *header imgproc_c.h* yang sudah di *include* pada *header cv.h*.

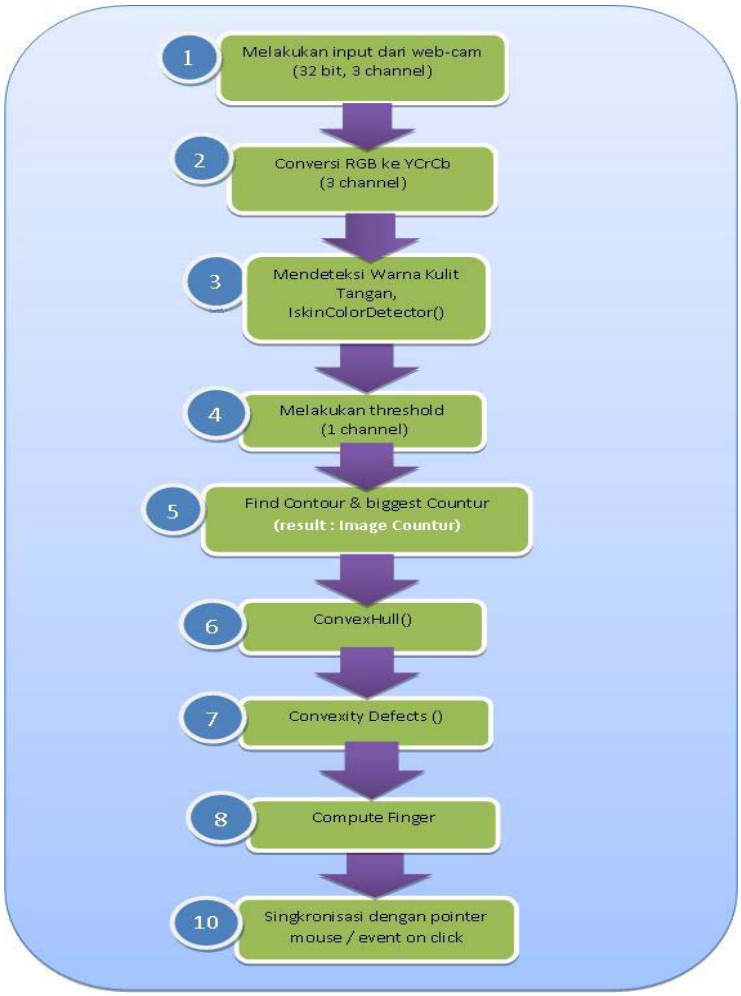
Setelah mendapatkan *hull* tangan manusia yang merupakan lekuk dari tangan manusia lalu proses dilanjutkan dengan menghitung titi *defects* dengan metoda *convexity defects*. Berbeda dengan *convex hull*, *convexity defects* memiliki tipe data tersendiri sebagai hasil komputasinya yaitu CvConvexityDefect seperti yang diperlihatkan pada bab II mengenai *ceonvexity defects*, yang nanti berfungsi menampung titik *defects* yang akan digunakan sebagai parameter untuk menghitung jumlah jari manusia. Metode *convexity defect* terdapat pada library *libopencv_imgproc.so* di *include* pada *header imgproc_c.h.*

Berikut adalah rutin yang digunakan untuk melakukan *convexity defects* dalam openCV :

```
CvSeq* cvConvexityDefects(
          const CvArr* contour,
          const CvArr* convexhull,
          CvMemStorage* storage = NULL
);
```

Parameter *contour* merupakan contour dari tangan hasil threshold pencarian warna kulit, sedangkan parameter *convexhull* merupakan nilai *hull* hasil komputasi *convex hull* sebelumnya. Yang dihasilkan berupa sequence yang nantinya dirubah dalam tipe data *cvConvexityDefects* array 2 dimensi.

Pada gambar III.2 adalah langkah yang dibutuhkan untuk menggunakan metode Convex-Hull dan Convexity Defects hingga hasil dari metode digunakan untuk melakukan penghitungan jumlah jari dan melakukan singkronisasi dengan *event click* mouse komputer.



Gambar III.4 Tahap-tahap yang dilakukan untuk metode *convex hull* dan *convexity defects*

Berikut adalah penjelasan dari gambar III.4 :

1. Input dilakukan dengan menggunakan *web-cam*.
2. Melakukan konversi untuk setiap image dari frame kedalam format warna YCrCb.
3. Melakukan pendeteksian warna kulit dengan melakukan perbandingan warna maksimum dan minimum kulit.
4. Melakukan thresholding dengan melakukan segmentasi berdasarkan warna kulit yang ditemukan.

5. Melakukan pencarian dan penggambaran *contour* juga mencari *contour* yang paling besar yang mungkin ditemukan.
6. Melakukan pencarian nilai *hull* dengan menggunakan metode *convexhull*.
7. Melakukan pencarian untuk titik-titik defect dari nilai *hull* yang telah didapatkan sebelumnya, pencarian dilakukan dengan menggunakan metode *convexity defect.*
8. Melakukan perhitungan jumlah jari yang terlihat pada kamera dengan memanfaatkan jarak dari setiap titik defect yang ditemukan.
9. Melakukan singkronisasi *event mouse* dengan hasil perhitungan jumlah jari yang terlihat, dengan nilai jari 0-1 untuk klik kiri, 2 untuk klik kanan dan 3-5 untuk keadaan mouse normal.

III.1.3.4 Deteksi Motion

Setelah pendeteksian posisi tangan dan juga menghitung jumlah jari dan melakukan singkronisasi pada pointer mouse lalu melakukan teknik motion, atau melakukan tracking pada pergerakan tangan. Disini algoritma pyramidal lucas kanade digunakan.

Sama seperti halnya metode convexhull ataupun convexity defects, penggunaan algoritma pyramidal lucas kanade yang ada dalam openCV tidak dapat begitu saja digunakan, ada proses-proses yang harus dilakukan sebelum dilakukan tracking motion pada suatu objek dalam penelitian ini adalah tangan manusia.

III.1.3.4.1 Pembuatan Citra Grayscale

Pada tahap awal dilakukan pengambila citra hasil komputasi contour pada contour paling besar/*biggest contour* untuk nantinya citra tersebut di copy kedalam dua citra baru dalam bentuk *grayscale* (walapun citra sudah dalam keadaan satu bit, ini dimaksudkan untuk menghindari kemungkinan ada perbedaan data atribut pada citra seperti *depth*). Dilakukan pengkonvesian citra dengan menggunakan rutin openCV cvCvtColor untuk merubah citra kedalam bentuk standar grayscale. Berikut adalah prototype dari cvCvtColor dala openCV:

```
void cvCvtColor(
          const CvArr* src,
          CvArr* dst,
          int code
          );
```

Metode konversi warna *cvCvtColor()* terdapat pada library *libopencv_imgproc.so* dengan *header imgproc_c.h* dan di *include* pada *header cv.h*

Citra grayscale yang dibuat adalah dua buah, satu citra (citra A) untuk posisi iterasi ke-i dan satu citra untuk posisi iterasi ke-(i-1). Karena algoritma lucas kanade menggunakan dua citra untuk melihat pergerakan suatu pixel dengan melihat citra pertama dengan kedua yaitu citra yang sudah berubah dimana pixel tersebut berpindah tempat.

III.1.3.4.2 Pencarian Titik Awal Corner dengan Good Feature ToTrack

Setelah itu dilakukan pencarian *corner* terbaik untuk dilakukan pencaria pixel. Karena sebelumnya telah dilakukan proses deteksi tangan maka prosedur

pencarian corner terbai atau *good feature to track* menjadi jauh lebih cepat dan efisien. Berikut adalah rutin untuk metode *good feature to track* dalam openCV :

```
void cvGoodFeaturesToTrack(
          const CvArr* image,
          CvArr* eigImage,
          CvArr* tempImage,
          CvPoint2D32f* corners,
          int* corner_count,
          double quality_level,
          double min_distance,
          const CvArr* mask = NULL,
          int block_size = 3,
          int use_harris = 0,
          double k = 0.4
          );
```

Metode *cvGoodFeaturesToTrack()* ada pada *library libopencv_imgproc.so* dengan *header imgproc_c.h* yang sudah di include pada header *cv.h*.

III.1.3.4.3 Pencarian Corner

Setelah itu dilakukan pencarian *corner* atau sudut-sudut yang memungkinkan terlihat dan terjadi perubahan. Pencarian didasarkan pada hasil pencari *good feature to track* sebelumnya. Adapun rutin untuk melakukan pencarian corner dalam openCV adalah sebagai berikut :

```
void cvFindCornerSubPix(
          const CvArr* image,
          CvPoint2D32f* corners,
          int count,
          CvSize win,
          CvSize zero_zone,
          CvTermCriteria criteria
          );
```

Metode *cvFindCornerSubPix()* ada pada *library libopencv_imgproc.so* dengan *header imgproc_c.h* yang sudah di include pada header *cv.h*.

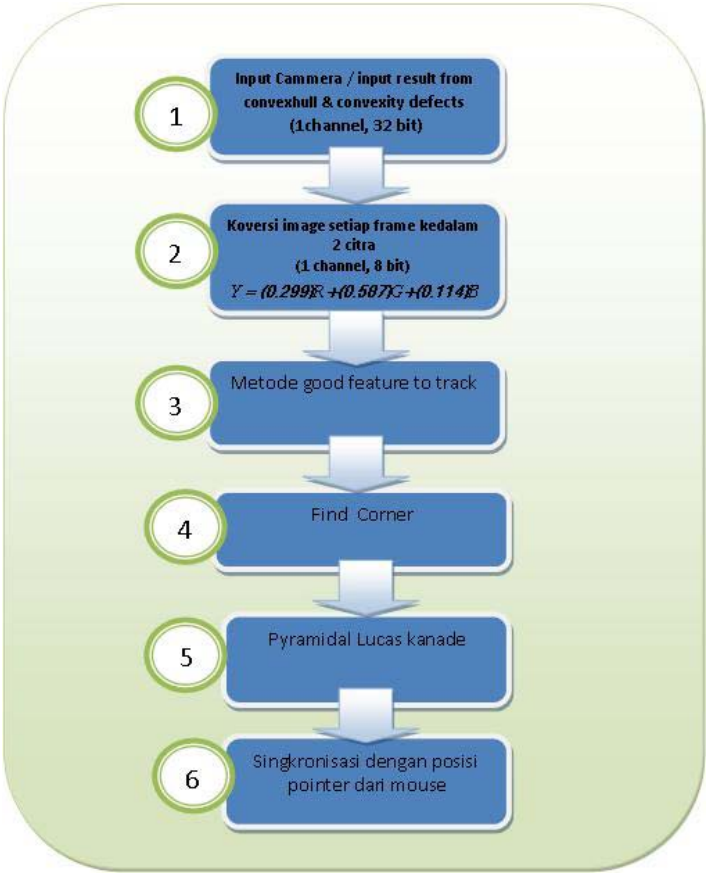
III.1.3.4.4 Tahap Pyramidal Lucas-Kanade

Setelah diketahui setiap *corner* yang terlihat lalu incormasi tersebut ditandai dan terus di monitor setiap pergerakan pixel tersebut dengan metode *pyramidal Lucas-Kanade*. *Pyramidal Lucas-Kanade* melakukan pembandingan antara satu gambar A (frame ke-i) dengan gambar B (frame ke-(i-1)) yang nantinya memberikan informas atau member result kordinat awal dan kordinat yang baru dan harus merupakan format warna *grayscale.*

Setelah melakukan komputasi dengan mengukur perpindahan pixel (biasa disebut *optical flow*) lalu algoritma pyramidal lucas-kanade akan memberikan return value berupa kordinat perpindah dari suatu pixel. Setelah itu kordinat (x,y) tersebut diberikan kepada fungsi pada libX11 untuk selanjutnya disesuaikan dengan posisi pointer mouse.

Metode *cvCalcOpticalFlowPyrLK()* terdapat pada *header tracking.hpp* dengan *library libopencv_video.so* yang sudah di *include* pada *header cv.h*.

Berikut adalah langkah-langkah yang diperlukan dalam implementasi metode Pyramidal Lucas-Kanade :



Gambar III.5 Langkah-langkah yang diperlukan dalam implementasi metode Pyramidal Lucas-Kanade

Pada gambar III.5, metode lucas kanade menggunakan citra yang didapat dari hasil perhitungan pencarian tangan sebelumnya, dengan memanfaatkan titik tengah dari tangan. Berikut adalah penjelasan dari gambar III.5 :

1. Pada gambar III.5 gambar diambil dari kamera dalam bentuk RGB.
2. Dilakukan konversi format warna gambar dari RGB menuju Grayscale.
3. Lalu menggunakan metode goodFeatureToTrack() untuk menemukan titik awal corner yang dipakai untuk melakukan pencarian corner lainnya.
4. Melakukan pencarian corner yang ada.

5. Menjalankan metode pyramidal lucas kanade dengan menggunakan gambar grayscale.
6. Melakukan singkronisasi posisi mouse dengan posisi tangan berdasarkan titik pergerakan yang dihasilkan dari metode pyramidal lucas kanade.

III.1.4 Analisis Kebutuhan Nonfungsional

Kebutuhan non fungsional yang dibutuhkan untuk membangun sistem ini mencakup dua hal yaitu; kebutuhan perangkat keras dan perangkat lunak.

III.1.4.1 Analisis Kebutuhan Perangkat Keras

Adapun kebutuhan perangkat keras yang diperlukan untuk menjalankan sistem ini adalah :

1. Processor minimal Pentium 4 3.1 GHz (lebih disesuaikan dengan sistem operasi yang dipakai).
2. RAM 384 MB atau lebih.
3. Hardisk minimal 45 GB (untuk Ubuntu 9.10) atau 50GB Ubuntu 10.10 (disesuaikan dengan sistem operasi yang dipakai).
4. *Web-cam* minimal 2 MP (disarankan 5 MP atau Play Station Eye 3).

III.1.4.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk menjalankan sistem pengontrol mouse pointer dengan *hand tracking* ini adalah :

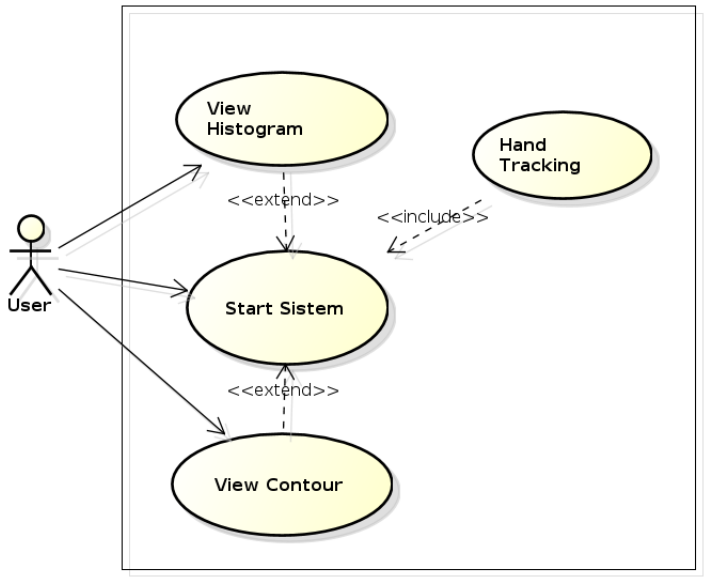
1. Sistem Operasi Ubuntu 9.10 atau Ubuntu 10.04 atau Ubuntu 10.10.

2. GCC compiler (untuk Ubuntu).
3. OpenCV 2.2.
4. QT 4 Framework

III.1.5 Analisis Kebutuhan Fungsional

III.1.5.1 Diagram Use Case

Didalam sistem hanya terdapat satu pengguna yaitu User. User ini yang nantinya menggunakan sistem yang telah terinstall dan ditanamkan dalam sistem operasinya (Ubuntu dan Windows). Tugas user sebagai aktor dapat terlihat pada diagram *Use Case* pada gambar



Gambar III.6 Use Case Sistem Pengontroll Mouse dengan Tangan

III.1.5.2 Start Sistem

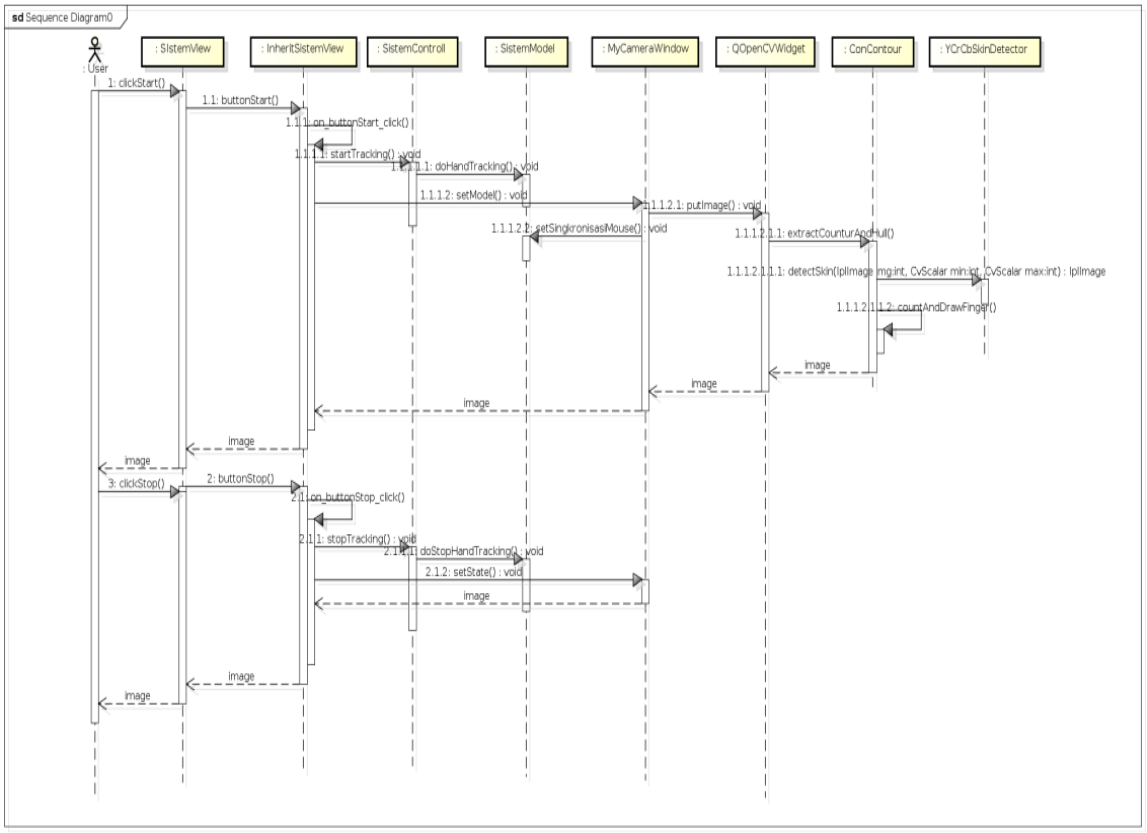
III.1.5.2.1 Start System

Start system merupakan proses yang mengawali semua sistem. Start system ini yang memberikan event untuk mengaktifkan semua metode yang perlu dilakukan. Skenario dari Start System dapat dilihat pada table .

Tabel III-1 Skenario Start Sistem

Identifikasi		
Nomor	1	
Nama	Start System	
Tujuan	Memulai sistem pengontroll mouse pointer dengan hand tracking	
Deskripsi	Sistem menerima event on click, dan mengirim sinyal untuk memulai tracking	
Aktor	User/pengguna	
Skenario Utama		
Kondisi awal	Sistem belum berjalan/belum melakukan Hand Tracking	
Aksi Aktor		Reaksi Sistem
1. Melakukan Click pada tombol ”start”		2. Menampilkan pendeteksian tangan bila tangan mulai terdeteksi
3. Menggerakan tangan dan posisi tangan		4. Melakukan singkronisasi dengan memindahkan posisi user
5. Melakukan click pada tombol “stop”		6. Menghentikan proses tracking dan kontrol
Kondisi akhir	7. Input citra yang berasal dari frame real time dari *web-cam* berhasil di tracking dan berhasil mengsingkrosnisasikan pointer.	

III.1.5.3 Dinamika Objek



Gambar III.7 Sequence Start System

Dilakukan proses utama yaitu *tracking motion* dan juga melakukan trigger untuk memulai sistem utama dan menghentikan sistem utama juga mengaktifkan atau mematikan kondisi histogram dan kontur.

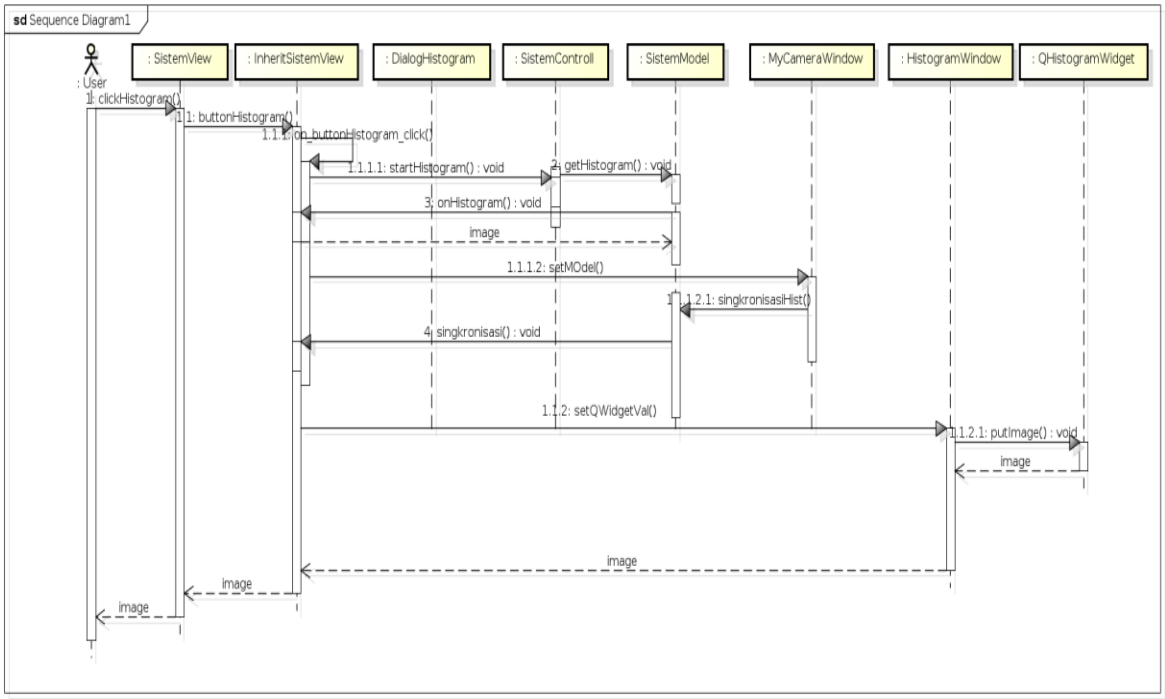
III.1.5.3.1 View Histogram

View Histogram proses untuk menampilkan kondisi atau menginterpretasikan histogram dari kondisi citra yang real-time pada setiap frame yang di ambil.

Tabel III-2 Skenario View Histogram

Identifikasi		
Nomor	2	
Nama	View Histogram	
Tujuan	Menampilkan histogram dari citra dari setiap frame yang di hitung, dan diinterpretasikan dalam bentuk citra.	
Deskripsi	Sistem menerima event on click tombol view Histogram, untuk menampilkan kondisi histogram dari citra yang terlihat.	
Aktor	User/pengguna	
Skenario Utama		
Kondisi awal	Sistem sudah berjalan histogram belum muncul	
Aksi Aktor		Reaksi Sistem
1. Melakukan Click pada tombol ”view histogram”		2. Menampilkan windows histogram dari citra yang sedang ditampilkan.
Kondisi akhir	3. Windows histogram tampil dan citra yang merupakan interpretasi dari histogram berubah secara dinamis sesuai keadaan citra dari frame yang diambil.	

III.1.5.4 Dinamika Objek



Gambar III.8 Sequence View Histogram

Setelah sistem utama di aktifkan maka dilakukan komputasi untuk memperlihatkan hasil komputasi sistem utama berupa histogram batang kondisi dari 255 warna RGB yang di wakili oleh 50 diagram batang secara dinamis.

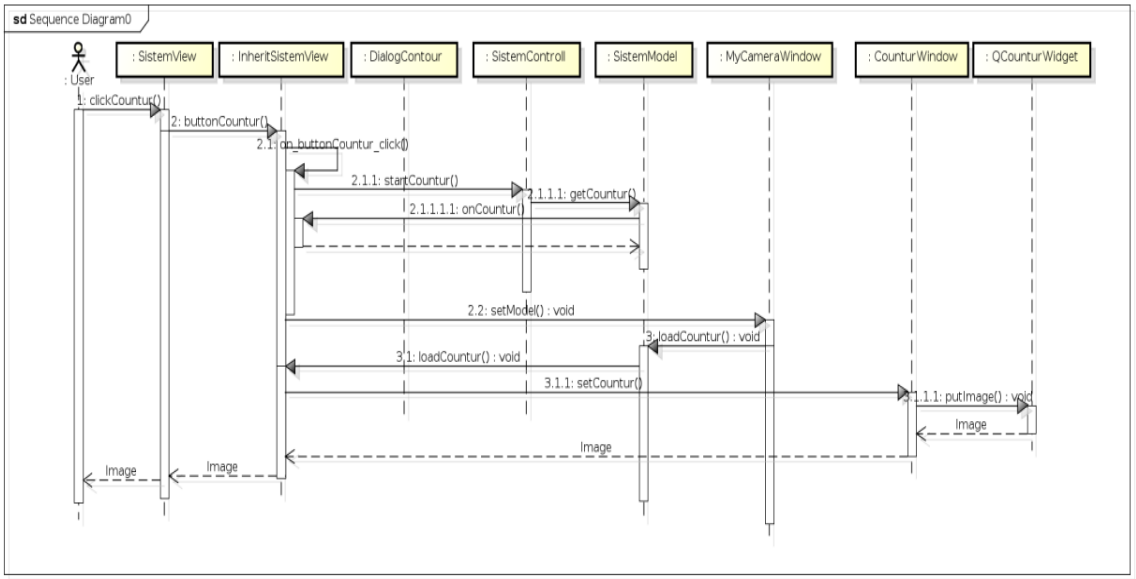
III.1.5.4.1 View Contour

View Contour proses untuk menampilkan citra yang sudah dikenakan oleh komputasi threshold dari sumber citra yang sebelumnya diambil real-time pada setiap frame yang di ambil.

Tabel III-3 Skenario View Contour

Identifikasi		
Nomor	3	
Nama	View Histogram	
Tujuan	Menampilkan hasil komputasi threshold yang ditampilkan dalam windows.	
Deskripsi	Sistem menerima event on click tombol *Contour*, untuk menampilkan hasil Threshold dari citra yang terlihat.	
Aktor	User/pengguna	
Skenario Utama		
Kondisi awal	Sistem sudah berjalan,thresholdsudah dihitung, tetapi belum ditampilkan.	
Aksi Aktor		Reaksi Sistem
1. Melakukan Click pada tombol ”Contour”		2. Menampilkan windows *threshold* dari citra yang sedang ditampilkan.
Kondisi akhir	3. Windows threshold tampil dan citra yang merupakan hasil dari komputasi threshold berubah secara dinamis sesuai keadaan citra dari frame yang diambil.	

III.1.5.5 Dinamika Objek



Gambar III.9 Sequence View Contour

Proses komputasi perubahan citra menjadi kontur yang berupa citra 1 bit yang nantinya ditampilkan untuk memperlihatkan hasil proses komputasi yang sedang berlangsung.

III.1.5.5.1 HandTracking

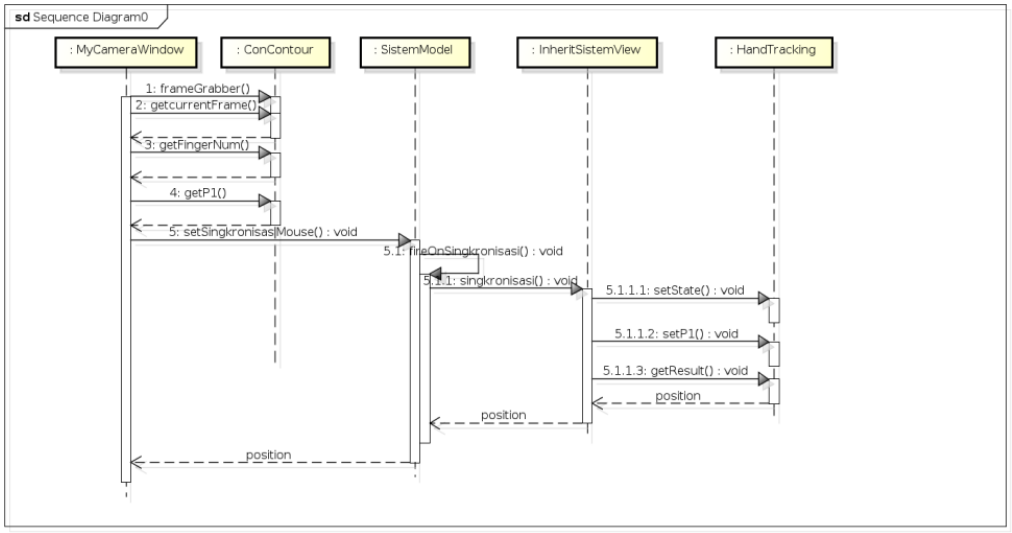
HandTracking merupakan proses untuk melakukan pencarian kordinat tangan lalu disingkronisasi dengan posisi pointer mouse pada sistem operasi komputer juga menyesuaikan setiap event click pada mouse pointer yang telah ditentukan.

Table III.1 Skenario Hand Tracking

Identifikasi	
Nomor	4
Nama	HandTracking
Tujuan	Melakukan tracking posisi tangan dan melakukan singkronisasikan posisi dan event dari mouse pointer.

Deskripsi	Sistem melakukan singkronisasi dan mencocokan posisi mouse pointer dengan kordinat posisi tangan dan juga setiap event yang telah ditentukan.	
Aktor	-	
Skenario Utama		
Kondisi awal	Sistem sudah berjalan,Contour sudah ditemukan, kordinat tangan sudah ditemukan.	
1. Aksi Aktor		2. Reaksi Sistem
-		3. Menyesuaikan posisi pointer dengan posisi tangan beserta menyesuaikan setiap event yang diberikan.
Kondisi akhir	4. Posisi mouse pointer terus berubah menyesuaikan dengan posisi tangan dan keadaan gesture tangan.	

III.1.5.6 Dinamika Objek

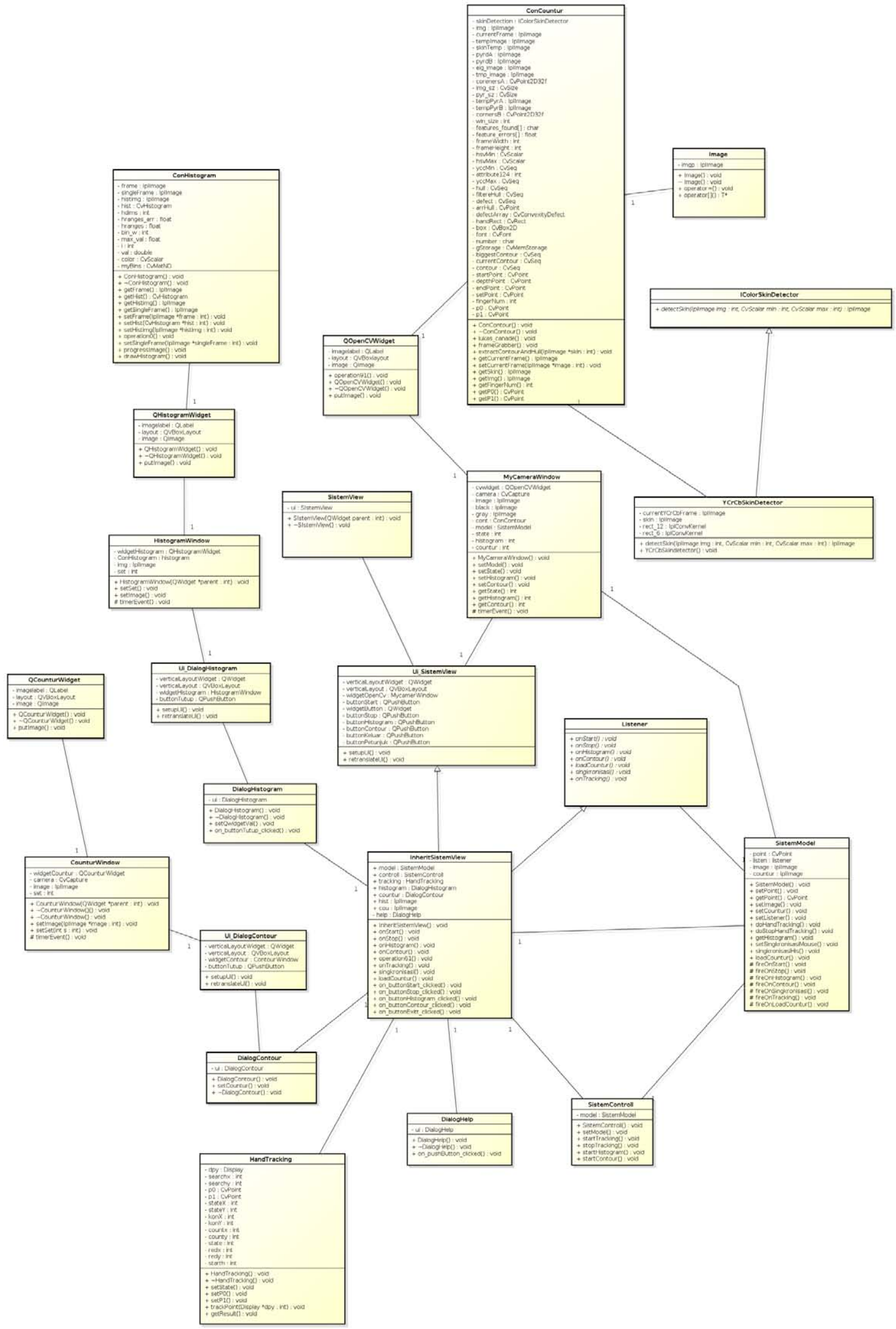


Gambar III.10 Sequence HandTracking

Setelah proses *tracking* dan *motion* di aktifkan, maka proses dilanjutkan untuk melakukan singkronisasi posisi tangan dan keadaan gesture tangan dengan posisi *mouse pointer* dan *event click* yang terdapat pada *mouse* dan sistem operasi

III.1.5.7 Diagram Class

Terdapat tiga jenis kelas dalam sistem, yaitu kelas *Model,* kelas *View,*dan kelas c*ontroller*. Kelas *Model* merupakan kelas yang digunakan sebagai penampung data atau representasi data itu sendiri dan juga merupakan tempat terjadinya proses *logic business*. Kelas *Controller* merupakan kelas yang digunakan sebagai tempat terjadinya seluruh proses yang ada dalam sistem dan juga validator yang memeriksa validasi data yang masuk, dan kelas *view*merupakan kelas yang digunakan untuk membangun tampilan antarmuka sistem. Seluruh kelas-kelas yang ada dalam sistem pencarian dapat dilihat pada gambar III.11.



Gambar III.11 Class Diagram System

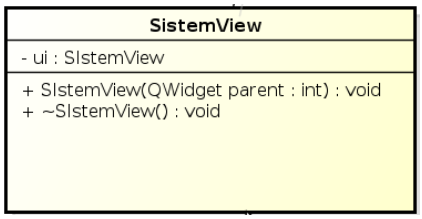
Setiap kelas-kelas pada diagram Class gambar III.9 memiliki jenis kelasnya masing-masing seperti yang terlihat pada tabel III.4.

Tabel III-4 Jenis-jenis Kelas

Nama Kelas	Jenis Kelas
SistemView	View
SistemControll	Controller
SistemModel	Model
HandTracking	Controller
ConHistogram	Controller
ConContour	Controller
Listener	Controller
InheritSistemView	View
CounturWindow	View
DialogHistogram	View
DialogCountur	View
HistogramWindow	View
IcolorSkinDetector	Controller
Image	Data
MyCameraWindow	View
QOpenCVWidget	View
YCrCbDetector	Controller

III.1.5.7.1 Kelas SistemView

SistemView adalah kelas yang merupakan Graphic User Interface (GUI) utama dari sistem, yang merupakan tempat interaksi user dan sistem baik untuk memulai sistem ataupun untuk menghentikan sistem serta menampilkan beberapa hasil komputasi.



Gambar III.12 Class SistemView

III.1.5.7.1.1 Deskripsi Atribut

Tabel III-5 Deskripsi Atribut Kelas SistemView

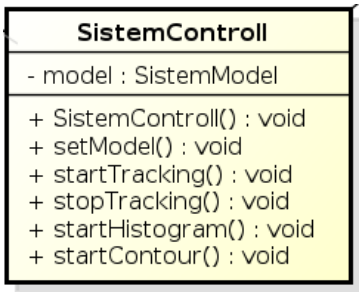
Nama	Tipe	Data	Keterangan
ui	QPushButton	Object	Objek button dari QT Framework untuk memulai sistem.

III.1.5.7.1.2 Deskripsi Operasi

1. *SistemView()*, merupakan Contructor Kelas SistemView.
2. *~SistemView(),*merupakan destructor Kelas SistemView.

III.1.5.7.2 Kelas SistemControll

Pada Gambar III.13 merupakan class SistemControll yang berfungsi untuk mengaktifkan semua proses, dan dalam sistem ini digunakan sebagai perantara antara kelas model dengan kelas MyCammeraOpenCV sehingga tidak terjadi tabrakan.



Gambar III.13 Kelas SistemControll

III.1.5.7.2.1 Deskripsi Atribut

Tabel III-6 Deskripsi Atribut Kelas SisteControll

Nama	Tipe	Data	Keterangan
model	SistemModel	Object	Objek yang nantinya akan mengirimkan nilai logic dan mengeksekusinya di kelas controll.

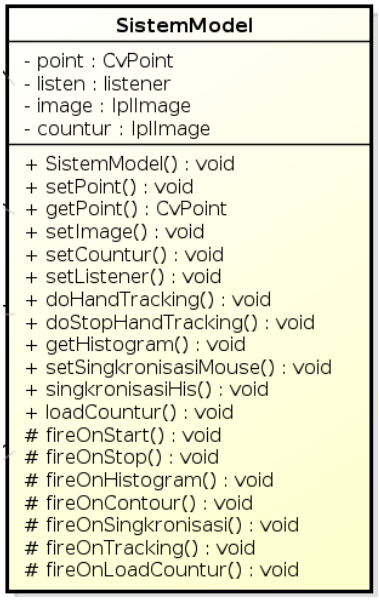
III.1.5.7.2.2 Deskripsi Operasi

1. *SistemControll()*, constructor kelas SistemControll;
2. *setModel()*, memasukan nilai untuk objek sistemModel.
3. *startTracking()*, menjalakan metode tracking tangan.
4. *stopTracking()*, menghentikan metode tracking tangan.

5. *startHistogram()*, menjalankan event untuk mengaktifkan tampilan histogram.
6. *startContour()*, menjalankan event untuk mengaktifkan tampilan hasil komputasi contour.

III.1.5.7.3 Kelas SistemModel

Kelas SistemModel merupakan kelas model yang berisi metode yang menuju proses *logic business* dari sistem sehingga memudahkan untuk mengakses dan menekan kemungkinan terjadi duplikasi karena pendeklarasian ulang header.



Gambar III.14 Class SistemModel

III.1.5.7.3.1 Deskripsi Atribut

Tabel III-7 Deskripsi Atribut Kelas SistemModel

Nama	Tipe	Data	Keterangan
point	CvPoint	Object	Nilai posisi kordinat dari tangan.
listen	SistemListener	Object	Objek dari kelas

			SistemListener yang nantinya menghubungkan model dengan view.
image	IplImage	Object	Objek citra yang menampung hasil komputasi.
contour	IplImage		Objek citra contour yang menampung hasil komputasi contour.

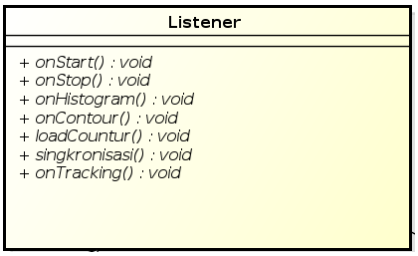
III.1.5.7.3.2 Deskripsi Operasi

1. SistemModel(), merupakan constructor dari kelas SisteModel.
2. setPoint(), metode accessor untuk mengisi nilai point.
3. getPoint(), metode accessor untuk mengambil nilai point.
4. setImage(), metode accessor untuk mengisi nilai citra image.
5. setContour(), metode accessor untuk mengisi nilai citra contour.
6. setListener(), metode accessor untuk mengisi nilai listener.
7. doHandTracking(), metode untuk memulai proses *hand tracking*.
8. doStopHandTracking(), metode untuk menghentikan proses *hand tracking*.
9. getHistogram(), mengaktifkan visualisasi *histogram*.
10. setSingkronisasiMouse(), memulai singkronisasi dengan *mouse pointer*
11. singkronisasi(), melakukan singkronisasi secara terus menerus.
12. loadContour(), mengambil nilai contour secara teruse menerus.
13. fireOnStart(), event untuk memulai hand tracking.
14. fireOnStop(), event untuk menghentikan hand tracking.

15. fireOnHistogram(), event untuk menampilkan histogram hasil komputasi.
16. fireOnContour(), even untuk menampilkan contour hasil komputasi.
17. fireOnSingkronisasi(), event melakukan singkronisasi dengan mouse pointer.
18. fireOnTracking, event untuk melakukan tracking untuk setiap frame yang baru.
19. fireOnLoadContour, event untuk mengambil contour baru dari setiap frame yang baru.

III.1.5.7.4 Kelas SistemListener

Kelas Listener adalah kelas pembantu untuk menghubungkan kelas model dan kelas view sehingga setiap perubahan data citra pada model akan langsung ditampilkan pada kelas view. Gambar III.15 merupakan gambar citra SistemListener.



Gambar III.15 Abstract Class SistemListener

III.1.5.7.4.1 Deskripsi Atribut

Kelas abstract SistemListener tidak memiliki atribut.

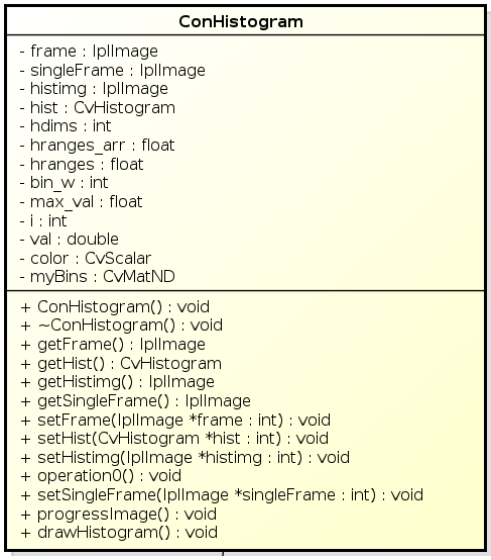
III.1.5.7.4.2 Deskripsi Operasi

1. OnStart(), event untuk memulai hand tracking.

2. OnStop(), event untuk menghentikan hand tracking.
3. OnHistogram(), event untuk menampilkan histogram hasil komputasi.
4. OnContour(), even untuk menampilkan contour hasil komputasi.
5. OnSingkronisasi(), event melakukan singkronisasi dengan mouse pointer.
6. OnTracking, event untuk melakukan tracking untuk setiap frame yang baru.
7. OnLoadContour, event untuk mengambil contour baru dari setiap frame yang baru.

III.1.5.7.5 Kelas ConHistogram

Kelas ConHistogram merupakan kelas yang melakukan perhitungan untuk mendapatkan visualisasi histogram yang terjadi pada frame yang sedang dilihat. Gambar III.16 merupakan gambar kelas ConHistogram.



Gambar III.16 Class ConHistogram

III.1.5.7.5.1 Deskripsi Atribut

Tabel III-8 Deskripsi Atribut Kelas ConHistogram

Nama	Tipe	Data	Keterangan
frame	IplImage	Object	Tipe data citra, frame yang didapat dari kamera.
singleFrame	IplImage	Object	Tipe data citra, objek penyimpan citra frame sementara.
histImg	IplImage	Object	Tipe data citra dari histogram yang terbentuk.
hist	CvHistogram	Object	Tipe data histogram, menyimpan data histogram citra.
hdims	Integer	Numeric	Nilai hasil bagi dari ukuran histogram.
hranges_arr	float	real	Nilai range histogram berbentuk array.
hranges	float	real	Nilai range histogram.
bin_w	int	Numeric	Nilai bin dari histogram.
Max_val	float	Real	Nilai maximum.
i	integer	numeric	Iterator
val	double	real	-
color	CvScalar	Object	Nilai warna dari citra.
myBins	CvMatND	Object	-

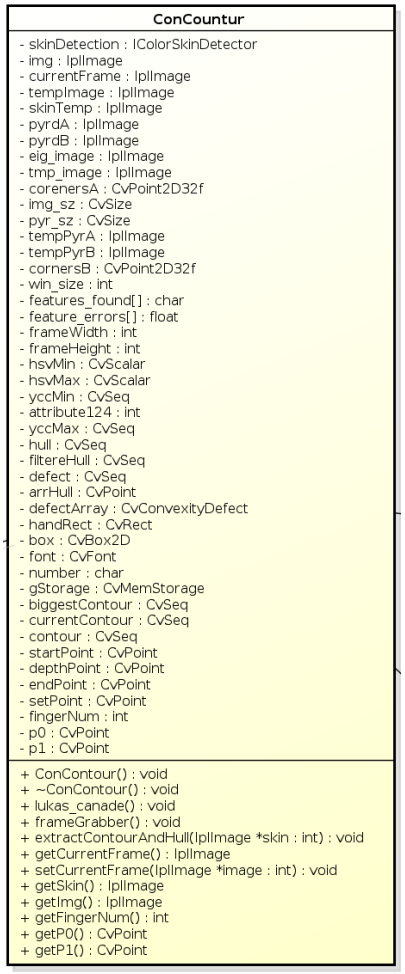
III.1.5.7.5.2 Deskripsi Operasi

1. *ConHistogram(),* constructor dari kelas ConHistogram.
2. *~ConHistogram(),* destructor dari kelas ConHistogram.
3. *getFrame(),* mengambil nilai dari objek frame.
4. *setHist(),* mengambil nilai dari objek hist
5. *getHistImg(),*mengambil nilai dari objek citra histogram.

6. *getSingleFrame()*,mengambil nilai dari objek singleFrame.
7. *setFrame*(), memasukan/menginisialisasi nilai untuk objek frame.
8. *setHist()*,memasukan/menginisialisasi nilai untuk objek hist.
9. *setHistImg()*,memasukan/menginisialisasi nilai untuk objek histImg.
10. *operation()*, operasi komputasi histogram.
11. *setSingleFrame()*,memasukan/menginisialisasi nilai untuk objek singleFrame.
12. *progressImage()*, memproses image sebelum dilakukan komputasi histogram.
13. *drawHistogram()*, menggambar histogram hasil komputasi.

III.1.5.7.6 Kelas ConContour

Kelas ConContour merupakan kelas yang bertanggung jawab terhadap proses utama untuk melakukan segmentasi, pendeteksian tangan hingga implementasi lucas-kanade. Pada gambar III.17 merupakan gambar dari kelas ConContour.



Gambar III.17 Class ConContour

III.1.5.7.6.1 Deskripsi Atribut

Tabel III-9 Deskripsi Atribut Kelas ConContour

Nama	Tipe	Data	Keterangan
skinDetection	IColorSkinDetector	Object	Objek pendeteksi warna kulit
img	IplImage	Object	Objek Citra untuk lucas kanade
currentFrame	IplImage	Object	Objek citra yang sedang di proses
tempImage	IplImage	Object	Objek citra yang dipakai sebagai *variable temporary*
skinTemp	IplIamge	Object	Objek citra sementara hasil segmentasi kulit
pyrdA	IplImage	Object	Objek citra pixel awal

			untuk lucas kanade
pyrdB	IplImage	Object	Objek citra pixel akhir untuk lucas kanade
eig_Image	IplImage	Object	Objek citra temporary untuk proses lucas kanade
tmp_image	IplImage	Object	Objek citra temporary untuk proses lucas kanade
coremersA	CvPoint2D32f	Object	Objek citra temporary untuk proses lucas kanade
img_sz	CvSize	Object	Objek citra temporary untuk proses lucas kanade
pyr_sz	CvSize	Object	Objek citra temporary untuk proses lucas kanade
tempPyrA	IplImage	Object	Objek citra temporary untuk proses lucas kanade
tempPyrB	IplImage	Object	Objek citra temporary untuk proses lucas kanade
corners_B	CvPoint32D3F	Object	Objek citra temporary untuk proses lucas kanade
win_size	Int	Object	Objek yang berisi ukuran window
features_found	Char	Object	Objek mark untuk lucas kanade
Feature_errors	float	Object	Objek temporary untuk proses lucas kanade
frameWidth	Int	Object	Objek temporary untuk proses lucas kanade, ukuran citra.
frameHeight	Int	Object	Objek temporary untuk proses lucas kanade, ukuran citra.
yccMin	CvScalar	Object	Objek nilai minimum warna kulit.
yccMax	CvScalar	Object	Objek nilai maksimum warna kulit.
Attribute124	Int	Object	Objek temporary untuk proses lucas kanade.
hull	CvSeq	Object	Objek untuk menyimpan titik point hull.
filtereHull	CvSeq	Object	Objek untuk menyimpan titik point hull yang sudah di filter.
defect	CvSeq	Object	Objek untuk menyimpan titik-titik poin defect.

arrHull	CvPoint	Object	Objek untuk menyimpan titik-titik poin hull dalam bentuk array.
defectArray	CvConvexityDefect	Object	Objek untuk menyimpan titik-titik poin *defects* dalam bentuk array.
handRext	CvRect	Object	Objek poin kotak pembatas tangan.
box	CvBox2D	Object	Objek box untuk titik tengah area tangan.
font	CvFont	Object	Objek tulisan pada citra.
number	Char	Object	Objek temporary berupa angka.
gStorage	CvMemStorage	Object	Objek memori yang di alokasikan.
biggestContour	CvSeq	Object	Objek sequence contour terbesar.
currentContour	CvSeq	Object	Objek sequence contour yang sedang diproses.
contour	CvSeq	Object	Objek sequence *temporary.*
startPoint	CvPoint	Object	Objek point untuk poin awal dari defect.
endPoint	CvPoint	Object	Objek point untuk poin akhir dari defect.
depthPoint	CvPoint	Object	Objek point untuk poin jarak dari defect.
setPoint	CvPoint	Object	Objek untuk menginisialisasi Point.
fingerNum	Int	Object	Objek jumlah jari.
p0	CvPoint	Object	Objek posisi awal dari lucas kanade.
p1	CvPoint	Object	Objek posisi akhir dari lucas kanade.

III.1.5.7.6.2 Deskripsi Operasi

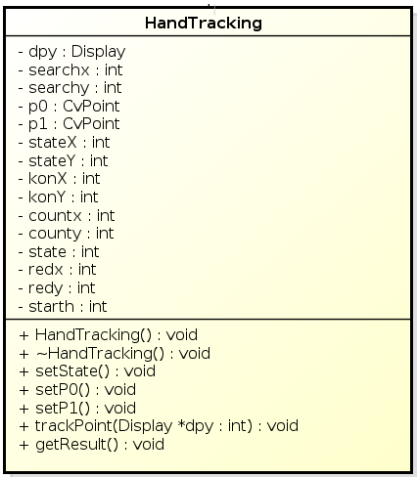
7. *ConContour()*, constructor dari kelas ConContour.

8. ~ConContour(), destructor dari kelas ConContour.

9. *lukas_canade()*, memanggil metode lucas kanade.
10. *frameGrabber()*, melakukan proses utama.
11. *extractContourAndHull()*, melakukan pencarian terhadap contour dan hull.
12. *getCurrentFrame()*, mengambil nilai dari objek currentFrame.
13. *setCurrentFrame()*, mengisi atau menginisialisasi objek currentFrame.
14. *getSkin()*, mengambil nilai dari objek currentFrame.
15. *getImg()*, mengambil nilai dari objek img.
16. *getFingerNum()*, mengambil nilai dari objek fingerNum.
17. *getP0()*, mengambil nilai dari objek p0.
18. *getp1()*, mengambil nilai dari objek p1.

III.1.5.7.7 Kelas HandTracking

Kelas HandTracking merupakan kelas yang melakukan singkronisasi pointer mouse baik posisi dari mouse hingga event klik kiri atau kanan dari mouse. Gambar III.18 merupakan gambar kelas HandTracking.



Gambar III.18 Class HandTracking

III.1.5.7.7.1 Deskripsi Atribut

Tabel III-10 Deskripsi Atribut Kelas HandTracking

Nama	Tipe	Data	Keterangan
dpy	Display	Object	Objek dari server pointer mouse X11
searchX	Int	Object	Objek nilai posisi x yang di cari
searchY	Int	Object	Objek nilai posisi y yang di cari
p0	CvPoint	Object	Objek kordinat awal citra dari lucas kanade.
p1	CvPoint	Object	Objek kordinat akhir citra dari lucas kanade.
stateX	int	Object	Objek nilai x pada posisi kursor
stateY	Int	Object	Objek nilai y pada posisi kursor
konX	Int	Object	Objek kondisi kordinat yang sedang berlangsung.
konY	Int	Object	Objek kondisi kordinat yang sedang berlangsung.
countx	Int	Object	Objek hasil perhitungan nilai x.
county	Int	Object	Objek hasil perhitungan nilai y.

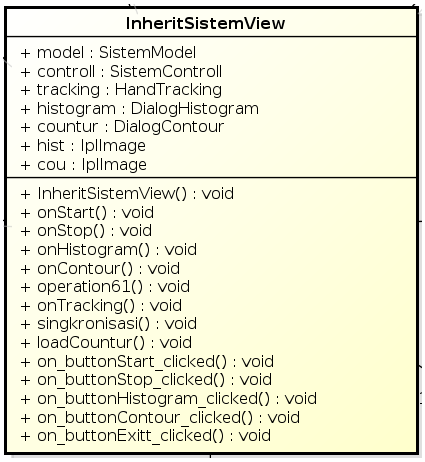
state	Int	Object	Objek status pada pointer mouse.
redx	Int	Object	-
redy	int	Object	-
stath	Int	Object	-

III.1.5.7.7.2 Deskripsi Operasi

1. HandTracking(), constructor dari kelas HandTracking.
2. ~HandTracking(), destructor dari kelas HandTracking.
3. setState(), mengisi/menginisialisasi objek state.
4. setP0(), mengisi/menginisialisasi objek p0.
5. setP1(), mengisi/menginisialisasi objek p1.
6. on_mouse(), melakukan proses singkronisasi pointer mouse.
7. trackPoint(), mengambil informasi pointer mouse.
8. play(), melaukukan inisialisasi driver pointer mouse.
9. getResult(), mendapatkan hasil dari singkronisasi pointer mouse.

III.1.5.7.8 Kelas InheritSistemView

Kelas InheritSistemView merupakan kelas turunan dari kelas SistemView yang merupakan interface utama yang berfungsi untuk mempermudah akses pada kelas SistemView. Gambar III.19 merupakan kelas InheritSistemView.



Gambar III.19 Class InheritSistemView

III.1.5.7.8.1 Deskripsi Atribut

Tabel III-11 Deskripsi Atribut Kelas MouseControll

Nama	Tipe	Data	Keterangan
model	SistemModel	Object	Objek dari kelas SistemModel
Control	SistemControll	Object	Objek dari kelas SistemControll.
tracking	HandTracking	Object	Objek dari kelas HandTracking.
histogram	DialogHistogram	Object	Objek dari kelas DialogHistogram.
contour	DialogContour	Object	Objek dari kelas DialogContour
hist	IplImage	Object	Objek temporary untuk image histogram.
cou	IplImage	Object	Objek temporary untuk image contour.

III.1.5.7.8.2 Deskripsi Operasi

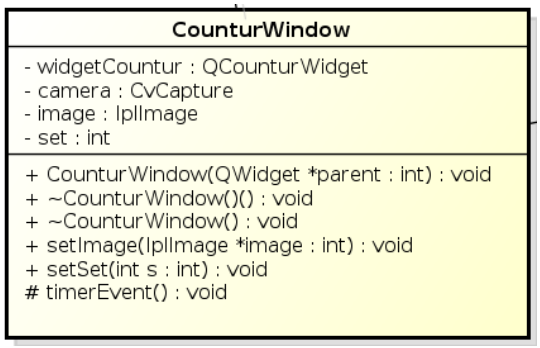
1. *onStart()*, method event setelah event start.

2. *onStop()*, method event setelah event stop.

3. *onHistogram(),* method event setelah event Histogram.
4. *onContour(),*method event setelah event Contour.
5. *onTracking(),*method event selama tracking.
6. *sinkronisasi(),* method event selama singkronisasi.
7. *loadCountur(),*method event selama pencarian contour.
8. *on_buttonStart_clicked(),* method event start.
9. *on_buttonStop_clicked(),*method event stop.
10. *on_buttonHistogram_clicked(),* method event histogram.
11. *on_buttonContour_clicked(),* method event contour.
12. *on_buttonExit_clicked(),*method event exit.

III.1.5.7.9 Kelas ContourWindow

Kelas CounturWindow merupakan kelas yang melakukan penerjemahan citra yang dihasilkan openCV ke dalam standar framework QT. Gambar III.20 merupakan gambar kelas CounturWindow.



Gambar III.20 Kelas ContourWindow

III.1.5.7.9.1 Deskripsi Atribut

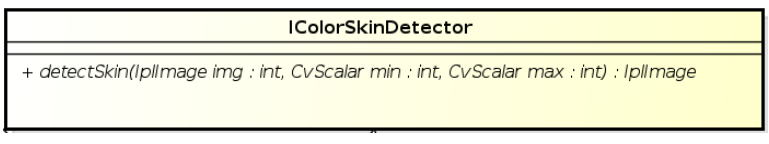
Nama	Tipe	Data	Keterangan
widgetCountur	QCounturWidget	Object	Objek dari kelas QCounturWidget.
camera	CvCapture	Object	Objek yang berfungsi untuk menangkap frame dari kamera.
image	Ipllmage	Object	Objek citra yang menampung citra dari frame.
set	int	Object	-

III.1.5.7.9.2 Deskripsi Operasi

1. CounturWindow(), constructor untuk kelas CounturWindow.
2. ~CounturWindow(), destructor untuk kelas CounturWindow.
3. setImage(IplImage *image), memasukan/menginisialisasi objek image.
4. setQWidget(), memasukan/menginisialisasi objek parent qwidget.
5. setSet(), memasukan/menginisialisasi objek set.
6. timerEvent(), melakukan proses secara berkala.

III.1.5.7.10 Kelas IColorSkinDetector

Kelas IColorSkinDetector merupakan kelas abstract atau kelas virtual untuk membantu dalam proses pendeteksian warna kulit. Gambar III.21 merupakan gambar kelas IColorSkinDetector.



Gambar III.21 Kelas Abstrak IColorSkinDetector

III.1.5.7.10.1 Deskripsi Atribut

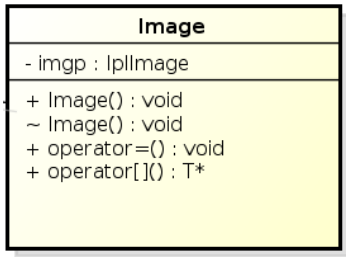
Kelas abstract SistemListener tidak memiliki atribut.

III.1.5.7.10.2 Deskripsi Operasi

1. detectSkin(), metode pendeteksi warna kulit.

III.1.5.7.11 Kelas Image

Kelas Image merupakan kelas yang membantu dalam tahap komputasi atau manipulasi citra untuk mendeteksi keberadaan objek tangan pada suatu image. Gamabar III.22 merupakan kelas Image.



Gambar III.22 Kelas Image

III.1.5.7.11.1 Deskripsi Atribut

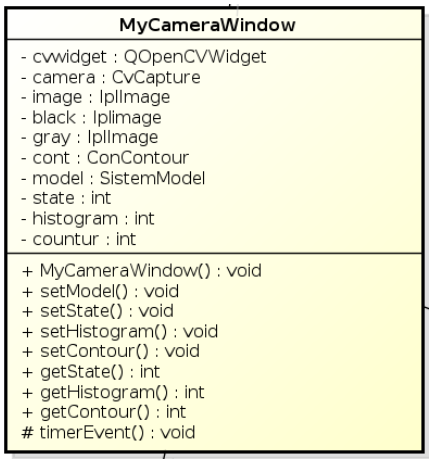
Nama	Tipe	Data	Keterangan
Imgp	IplImage	Object	Objek data citra.

III.1.5.7.11.2 Deskripsi Operasi

1. *Image()*, constructor kelas Image.
2. *~Image()*, destructor kelas Image.
3. *Operator=()*, override operator **=.**
4. *Operator[]()*, override operator [].

III.1.5.7.12 Kelas MyCameraWindow

Keals MyCameraWindow merupakan kelas untuk membantu dalam menampilkan *output* berupa gambar bergerak atau *video real time* kondisi yang diterima kamera. Gambar III.23 merupakan gambar kelas MyCameraWindow.



Gambar III.23 Kelas MyCameraWindow

III.1.5.7.12.1 Deskripsi Atribut

Nama	Tipe	Data	Keterangan
cvwidget	QOpenCVWidget	Object	Objek *grafik user interface* citra.
camera	CvCapture	Object	Objek penghubung device kamera.
image	Ipllmage	Object	Objek citra 3 channel RGB.
black	Ipllmage	Object	Objek citra dalam format 1 channel *black and white*.
gray	Ipllmage	Object	Objek citra dalam format 1 channel *grayscale.*
cont	ConContour	Object	Objek dari kelas ConContour.
model	SistemModel	Object	Objek dari kelas SistemModel.
state	int	Object	Objek status dari proses.

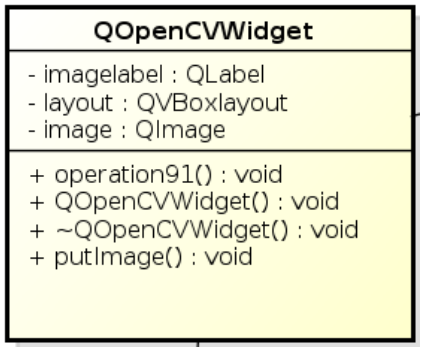
histogram	int	Object	Objek dari satus proses histogram.
countur	int	Object	Objek dari satus proses contour.

III.1.5.7.12.2 Deskripsi Operasi

1. *MyCameraWindow(),*consrtuctor untuk kelas MyCameraWindow.
2. *setModel(),* mengisi/menginisialisasi objek model.
3. *setState(),* mengisi/menginisialisasi objek state.
4. *setHistogram(),* mengisi/menginisialisasi objek histogram.
5. *setContour(),* mengisi/menginisialisasi objek contour.
6. *getState(),* mengambil nilai dari objek state.
7. *getHistogram(),* mengambil nilai dari objek histogram.
8. *getContour(),* mengambil nilai dari objek contour.
9. *timerEvent(),* melakukan proses secara berkala.

III.1.5.7.13 Kelas QOpenCvWidget

Kelas QOpenCVWidget merupakan widget yang dibuat untuk menampilkan frame yang didapat dari kamera dan hasil komputasi. Gambar III.24 merupakan gambar kelas QOpenCVWidget.



Gambar III.24 Kelas QOpenCVWidget

III.1.5.7.13.1 Deskripsi Atribut

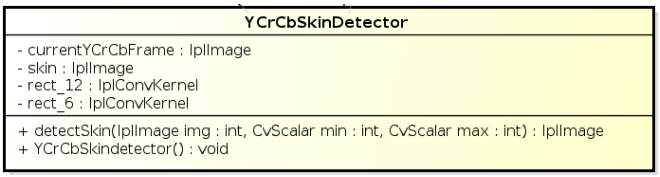
Nama	Tipe	Data	Keterangan
imagelabel	QLabel	Object	Objek text label untuk citra.
layout	QvBoxlayout	Object	Objek komponen qt.
image	Qlmage	Object	Objek citra komponen qt.

III.1.5.7.13.2 Deskripsi Operasi

1. *QOpenCVWidget(),* constructor dari kelas QOpenCVWidget.
2. *~QOpenCVWidget(),* destructor dari kelas QOpenCVWidget.
3. *putImage(),* mengisi atau menginisialisasi objek image.

III.1.5.7.14 Kelas YCrCbSkinDetector

Kelas YCrCbSkinDetector merupakan kelas turunan IColorSkinDetector untuk mengimplementasikan metode untuk mendeteksi warna kulit tangan manusia. Gambar III.25 merupakan gambar kelas YCrCbSkinDetector.



Gambar III.25 Kelas YCrCbSkinDetector

III.1.5.7.14.1 Deskripsi Atribut

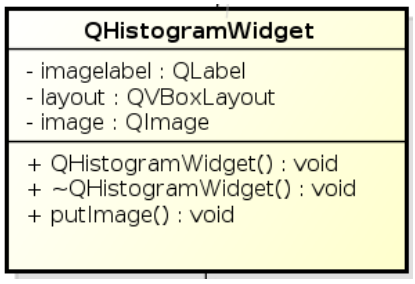
Nama	Tipe	Data	Keterangan
currentYCrCbFrame	IplImage	Object	Objek citra dalam format warna YCrCb.
Skin	IplImage	Object	Objek citra dari warna kulit.
Rect_12	IplConvKernel	Object	Object dari struct IplConvKernel.
Rect_6	IplConvKernel	Object	Object dari struct IplConvKernel.

III.1.5.7.14.2 Deskripsi Operasi

1. *YCrCbSkinDetector,* constructor dari kelas YCrCbSkinDetector
2. *detectSkin,* segmentasi warna kulit.

III.1.5.7.15 Kelas QHistogramWidget

Kelas QHistogramWidget merupakan widget yang dibuat untuk menampilkan frame yang didapat dari kamera dan hasil komputasi untuk membentuk histogram. Gambar III.24 merupakan gambar kelas QHistogramWidget.



Gambar III.26 Kelas QHistogramWidget

III.1.5.7.15.1 Deskripsi Atribut

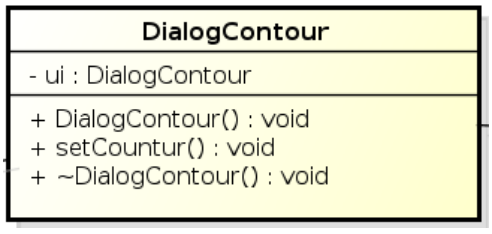
Nama	Tipe	Data	Keterangan
imagelabel	Qlabel	Object	Objek text label qt untuk menampung citra sementara.
layout	QVBoxLayout	Object	Object layout dari qt.
image	Qimage	Object	Object untuk image dari qt.

III.1.5.7.15.2 Deskripsi Operasi

1. *QHistogramWidget(),* constructor dari kelas QHistogramWidget.
2. *~QHistogramWidget(),* destructor dari kelas QHistogramWidget.
3. *putImage(),* mengisi/menginisialisasi objek image.

III.1.5.7.16 Kelas DialogContour

Kelas DialogContour merupakan kelas view yang menampilkan atau membangun interface untuk menampilkan citra contour dari frame yang terlihat. Gambar III.27 merupakan kelas DialogContour.



Gambar III.27 Kelas DialogContour

III.1.5.7.16.1 Deskripsi Atribut

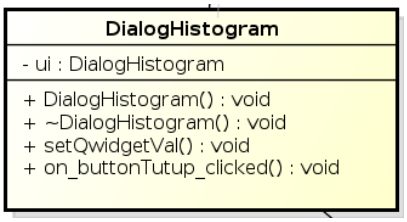
Nama	Tipe	Data	Keterangan
ui	DialogContou	Object	Objek dari kelas DialogContour.

III.1.5.7.16.2 Deskripsi Operasi

1. *DialogContour()*, constructor dari kelas DialogContour.
2. *setContour()*, mengisi/menginisialisasi nilai contour.
3. *~DialogContour()*, destructor dari kelas DialogContour.

III.1.5.7.17 Kelas DialogHistogram

Kelas DialogHisotgram merupakan kelas view yang menampilkan atau membangun interface untuk menampilkan citra histogram dari frame yang terlihat. Gambar III.27 merupakan kelas DialogContour.



Gambar III.28 Kelas HistogramWidnow

III.1.5.7.17.1 Deskripsi Atribut

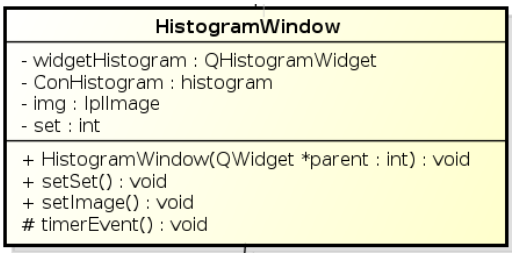
Nama	Tipe	Data	Keterangan
ui	HistogramContou	Object	Objek dari kelas DialogHistogram.

III.1.5.7.17.2 Deskripsi Operasi

1. *DialogHistogram()*, constructor dari kelas DialogHistogram.
2. *setQWidget()*, mengisi/menginisialisasi nilai QWidget.
3. *~DialogHistogram()*, destructor dari kelas Dialoghistogram.

III.1.5.7.18 Kelas HistogramWindow

Kelas HistogramWindow merupakan kelas widget yang disisipkan pada dialog untuk menampilkan hasil komputasi histogram yang menghasilkan visualisasi dari histogram. Gambar III.28 merupakan gambar dari kelas HistogramWindow.



Gambar III.29 Kelas HistogramWindow

III.1.5.7.18.1 Deskripsi Atribut

Nama	Tipe	Data	Keterangan
widgetHistogram	QHistogramWidget	Object	Objek dari kelas QHistogramWidget.
histogram	ConHistogram	Object	Objek dari kelas ConHistogram.
img	IplImage	Object	Objek citra.
set	int	Object	Objek status dari kelas HistogramWindow.

III.1.5.7.18.2 Deskripsi Operasi

1. *HistogramWindow(),* Constructor untuk kelas HistogramWindow.
2. *setQWidget(),* mengisi/menginisialisasi objek QWidget.
3. *setSet(),* mengisi/menginisialisasi objek set.
4. *setImage(),* mengisi/menginisialisasi objek image.
7. *timerEvent(),* melakukan proses secara berkala.

III.1.5.8 Diagram Activity

III.1.5.8.1 Aktifitas SistemView

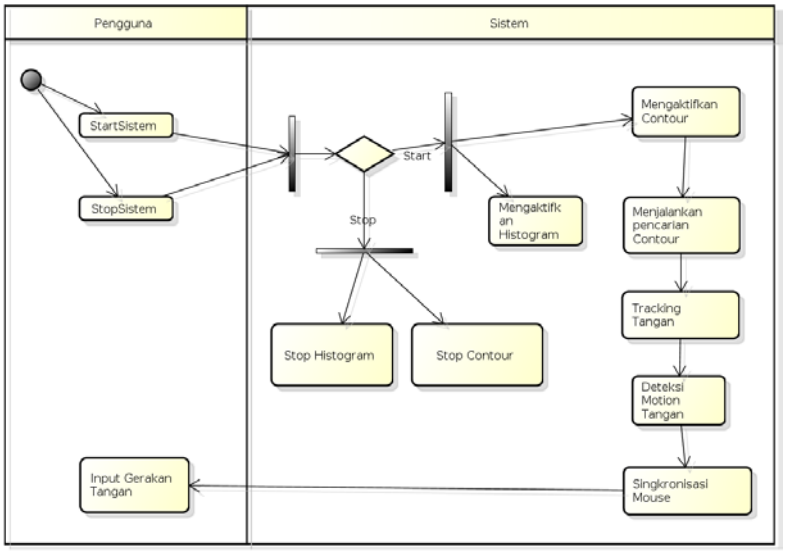
Aktifitas yang terdapat dalam sistem adalah melakukan tracking pada setiap frame yang di dapat dari input *web-cam*. Dengan melakukan komputasi yang dibutuhkan agar komputer dapat membedakan setiap gerakan dan bentuk dari tangan manusia. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut:

1. Mengaktifkan Sistem, dalam aktifitas ini user melakukan event untuk mengaktifkan aktifitas utama guna memulai proses tracking. Input yang masuk langsung dirubah kedalam bentuk grayscale didalam aktifitas ini.
2. Menghentikan Sistem, dalam aktifitas ini user melakukan event untuk menghentikan aktifitas utama. Input yang masuk langsung tidak akan diproses aktifitas ini.
3. Jika statusnya start, maka akan mengaktifkan komputasi histogram. Aktifitas ini melakukan komputasi yang menghasilkan histogram 2 dimensi dari keadaan citra frame yang terbaru.
4. Jika statusnya start, maka akan mengaktifkan pencarian kontur, aktifitas ini melakukan pencarian terhadap kontur yang terdapat pada citra yang membandingkan nilai warna dengan warna kulit. Setelah ditemukan maka kontur ini akan disimpan dalam bentuk linkedlist sequential. Lalu dilakukan penggambaran kontur dalam linkedlist tersebut.

5. Menjalankan pencarian kontur, aktifitas ini melakukan memisahkan antara warna kulit dan warna background dengan melakukan segmentasi dan menghasilkan kontur dan dilanjutkan dengan melakukan pencarian kontur terbesar.
6. Melakukan tracking tangan, aktifitas ini menggunakan metode *convex hull* dan *convexity defects* melakukan komputasi dari jumlah jari yang terlihat oleh komputer. Selain itu aktifitas ini melakukan pendeteksian vektor dari arah pergerakan tangan. aktifitas ini menampilkan kondisi kontur yang terjadi pada citra utama yang masih memiliki 3 channel dan 32 bit. Dengan menggunakan petunjuk tipe data llinkedlist sequential yang dikirim dari aktifitas pencarian kontur.
7. Deteksi *motion* tangan, aktifitas ini menjalankan metode lucas kanade untuk mencari *optical flow* atau pergerakan pixel dari tangan, sehingga dapat mengambil nilai perubahan pixel tersebut dalam bentuk kordinta 2 dimensi.
8. Singkronisasi, aktifitas yang melakukan singkronisasi kordinat tangan atau vektor posisi tangan dengan kordinat posisi pointer mouse. Aktifitas ini pun mensingkronisasikan event dari klik kiri atau pun klik kanan dari mouse yang disesuaikan dengan hasil komputasi kontur jari tangan yang terlihat kamera.
9. Input gerak tangan, dalam aktifitas ini user/pengguna melakukan pergerakan tangan dan perubahan kontur tangan untuk mengendalikan pointer mouse dengan segala komputasi yang terjadi di dalam sistem.
10. Jika status stop, maka proses komputasi histogram atau contour akan dihentikan.

Seluruh aktifitas hand tracking dapat dilihat tahapannya pada diagram *activity* pada gambar.

III.1.5.8.2



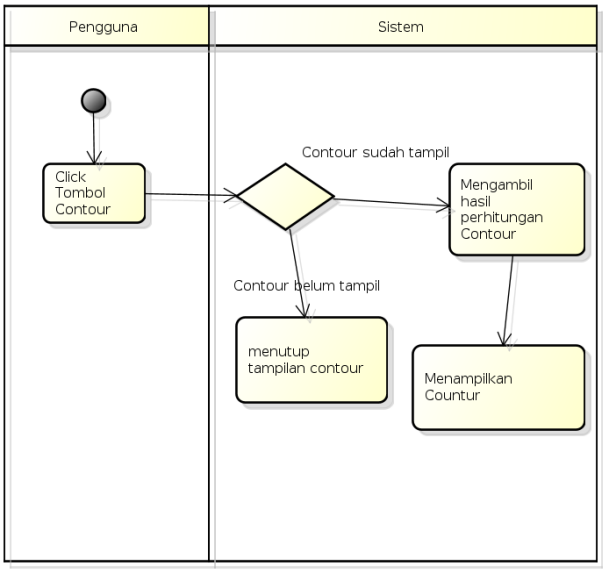
Gambar III.30Acitivty Diagram dari System

III.1.5.8.3 Aktifitas ViewContour

Aktifitas yang terdapat dalam *activity* ini adalah melakukan komputasi pada setiap frame yang di dapat dari input *web-cam* untuk menghasilkan contour dari citra dan menampilkannya. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut:

1. Click tombol contour, aktifitas ini pengguna melakukan click pada tombol contour, dan akan melakukan *check* status apakah contour sudah tampil atau belum.
2. Jika status contour belum tampil, maka statusnya adalah *start* sehingga selanjutnya adalah aktifitas Mengambil hasil perhitungan contour yaitu hasil perhitungan kontur dari proses utama yang sudah dalam berbentuk citra.

3. Menampilkan contour, aktifitas ini menampilkan citra hasil komputasi yang sudah di ambil dan di konversi agar dapat ditampilkan secara *real time*.
4. Jika status contour sudah tampil, maka statusnya adalah *stop* sehingga akan menuju aktifitas menutup tampilan contour sehingga akan menutup *window* tampilan citra hasil komputasi contour seblumnya.



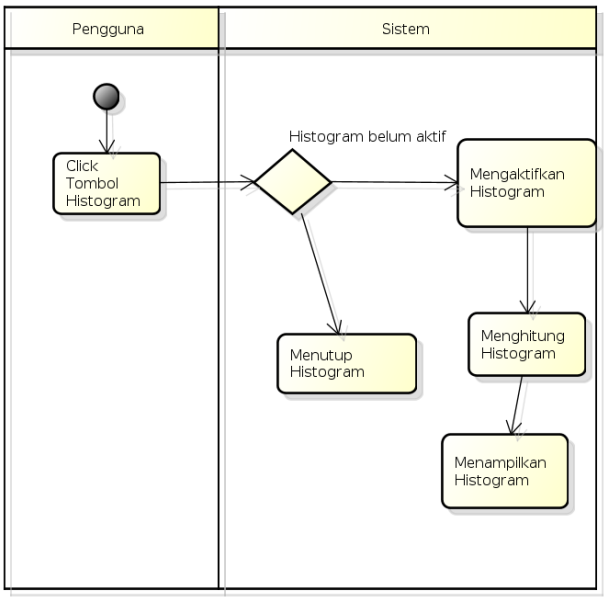
Gambar III.31 Diagram activiry dari ViewContour

III.1.5.8.4 Aktifitas ViewHistogram

Aktifitas yang terdapat dalam *activity* ini adalah melakukan komputasi pada setiap frame yang di dapat dari input *web-cam* untuk menghasilkan histogram dari citra dan menampilkannya. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut:

1. Click tombol histogram, aktifitas ini pengguna melakukan click pada tombol histogram, dan akan melakukan *check* status apakah histogram sudah tampil atau belum.

2. Jika status histogram belum tampil, maka statusnya adalah *start* sehingga selanjutnya adalah aktifitas mengaktifkan komputasi histogram, mempersiapkan untuk melakukan komputasi yang menghasilkan interpretasi histogram dari citra.
3. Menghitung histogram, aktifitas ini melakukan komputasi secara *real-time* untuk mendapatkan nilai-nilai citra untuk membangun histogram yang sesuai.
4. Menampilkan histogram, aktifitas ini menampilkan citra hasil komputasi histogram yang berbentuk diagram batang dan ditampilkan secara *real time*.
5. Jika status histogram sudah tampil, maka statusnya adalah *stop* sehingga akan menuju aktifitas menutup tampilan histogram sehingga akan menutup *window* tampilan citra hasil komputasi histogram seblumnya.



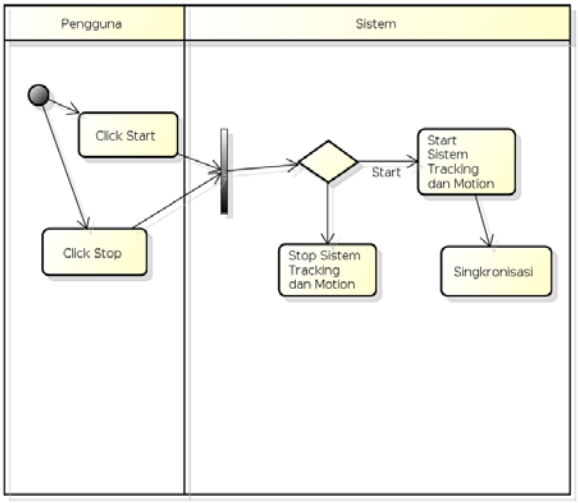
Gambar III.32 Diagram activity ViewHistogram

III.1.5.8.5 Aktifitas HandTracking

Aktifitas yang terdapat dalam *activity* ini adalah melakukan singkronisasi terhadap pointer mouse dengan posisi tangan dan gesture dari tangan. Tahapan-tahapan yang terjadi dalam sistem adalah sebagai berikut:

1. Mengaktifkan Sistem, dalam aktifitas ini user melakukan *event* untuk mengaktifkan aktifitas utama guna memulai proses *tracking*. Proses utama dari ssitem akan langsung berjalan, proses *tracking* tangan langsung dilakukan.
2. Menghentikan Sistem, dalam aktifitas ini user melakukan event untuk menghentikan aktifitas utama. Input yang masuk langsung tidak akan diproses aktifitas ini.
3. Jika Start, maka aktifitas Start Sistem *Trakcing* dan *Motion* akan dilakukan, aktifitas ini merupakan aktifitas utama dari sistem dana akan menghasilkan kordinat dari posisi tangan dan juga jumlah jari hasil komputasi dari metode *convexity defect.*
4. Proses singkronisasi, aktifitas ini melakukan singkronisasi terhada pointer mouse yang di set dengan menggunakan parameter yang nilainya didapat dari hasil komputasi pencarian posisi tangan sebelumnya untuk posisi kursor dan jumlah jari untuk melakukan *event* click kanan dan klik kiri.
5. Jika Stop, maka aktifitas Stop *tracking* dan *motion* dilakukan, sehingga tidak ada lagi komputasi *tracking* dan *motion*, sehingga tidak ada input bagi

parameter untuk singkronisasi pointer mouses dan sistem singkronisasi pun dihentikan.



Gambar III.33 Diagram activity HandTracking

III.1.5.9 Diagram State

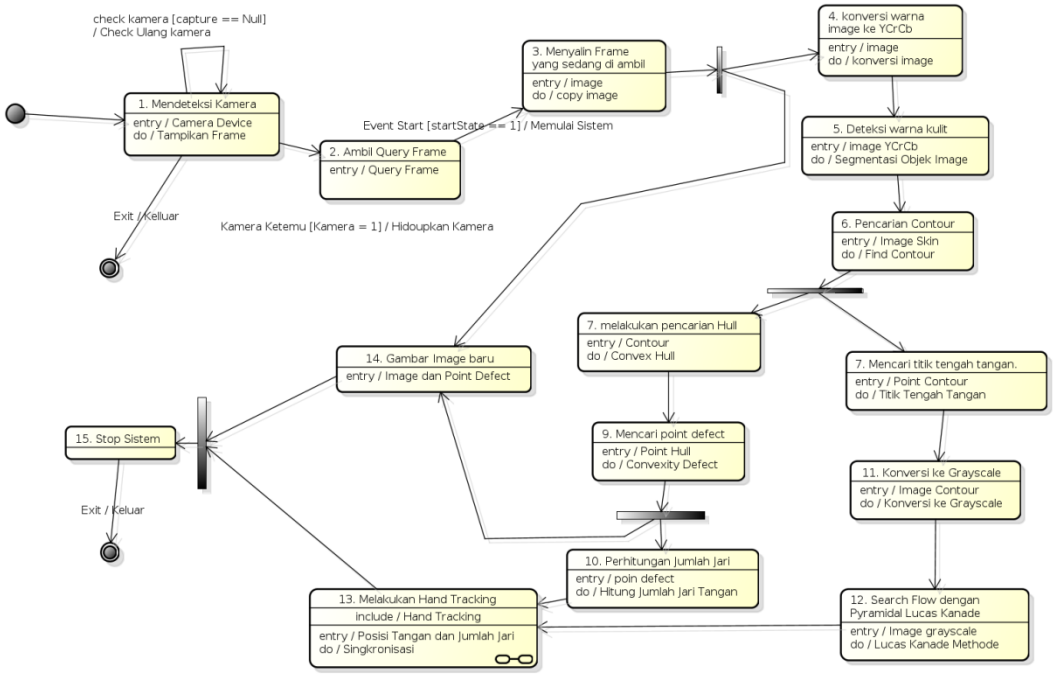
III.1.5.9.1 State Sistem

Interaksi yang terjadi dalam sistem yang menerima input melalui kamera *web*, digambarkan dalam diagram state berikut :

1. Pada tahap pertama sistem melakukan cek kondisi kamera, apabila tidak ada akan mengeluarkan *error* dan mengecek ulang kemungkinan keberadaan kamera, atau keluar dari sistem.
2. Jika terdeteksi maka dilakukan pengambilan *query frame* dari video yang didapat secara *real-time*, yang isinya merupakan kumpulan frame yang terlihat kamera.

3. Melakukan penyalinan citra dari setiap frame yang terlihat guna memisahkan citra untuk ditampilkan sebagai output yang terlihat dan citra untuk diolah dikenakan komputasi.
4. Untuk citra yang diperuntukan untuk dikenakan komputasi atau pengolahan citra, dilakukan konversi format warna dari RGB menuju YCrCb.
5. Dilakukan pencarian warna kulit atau mendeteksi warna kulit pada citra yang sudah dirubah ke dalam YCrCb dengan menggunakan range minimum dan maximum dari warna kulit, lalu dilakukan segmentasi untuk memisahkan objek kulit dan objek lainnya.
6. Dilakukan pencarian kontur dari kulit dan tangan dengan menggunakan kontur terbesar untuk mendeteksi objek tangan.
7. Dilakukan pencarian nilai *hull* pada kontur tangan yang ditemukan dengan menggunakan metode *convexhull*.
8. Citra hasil *convexhull* di lakukan komputasi untuk menemukan titik *defect* dengan menggunakan *convexity defects*.
9. Dari hasi kontur dilakukan perhitungan titik tengah dari objek tangan untuk nantinya dipakai untuk implementasi *pyramidal lucas kanade.*
10. Konversi citra titik tengah dari objek tangan ke dalam citra berformat *grayscale*.
11. Melakukan pergerakan atau pergeseran objek tangan dengan mengimplementasikan metode pyramidal lucas kanade.

12. Dilakukan komputasi menghitung jari yang terdeteksi pada citra kontur dengan memanfaatkan titik-titik *defect* yang telah ditemukan sebelumnya.
13. Melakukan singkronisasi pada posisi *mouse pointer* dengan posisi dari tangan hasil metode pyramidal lucas kanade dan singkronisasi event klik kanan dan klik kiri dengan menggunakan hasil perhitungan jumlah jari yang terbaca oleh sistem.
14. Dilakukan penambahan objek pada citra awal dengan menambahkan petunjuk atau navigator yang berupa gambar garis atau kontur dari hasil komputasi agar pengguna dapat melihat kondisi citra yang terlihat.
15. Menghentikan sistem yang sedang berjalan yang nantinya bisa dilanjutkan untuk menghentikan sistem.

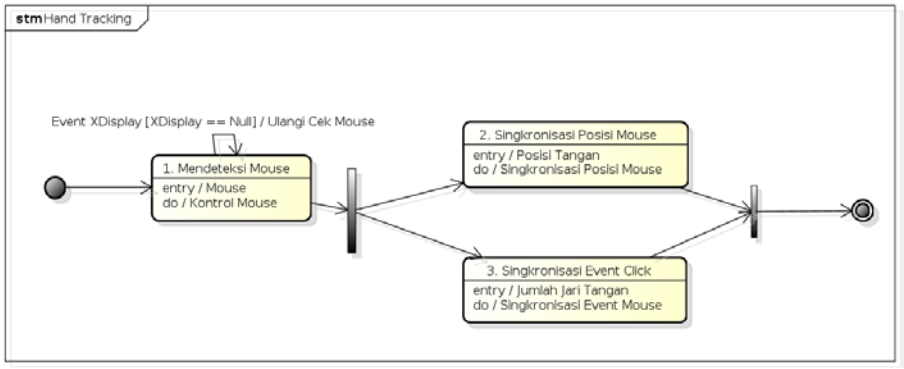


Gambar III.34 Diagram state sistem sengontrol pointer mouse dengan handtracking

III.1.5.9.2 State Hand Tracking

Proses tracking dalam sistem yang menerima input kordinat tangan dan melakukan singkronisasi dengan mouse pointer, digambarkan dalam diagram state pada gambar III.35.

1. Pada state pertama dilakukan pendeteksian mouse dengan menggunakan driver X11 dengan memanggil server mouse pada ubuntu dan menyesuaikan dengan ukuran *display* desktop.
2. Dilakukan singkronisasi posisi mouse dengan posisi tangan berdasarkan input kordinat posisi pada citra tangan yang didapat.
3. Dialakukan singkronisasi *event* klik kanan dan klik kiri disesuaikan dengan input jumlah jari yang dideteksi dalam citra yang memiliki objek tangan.

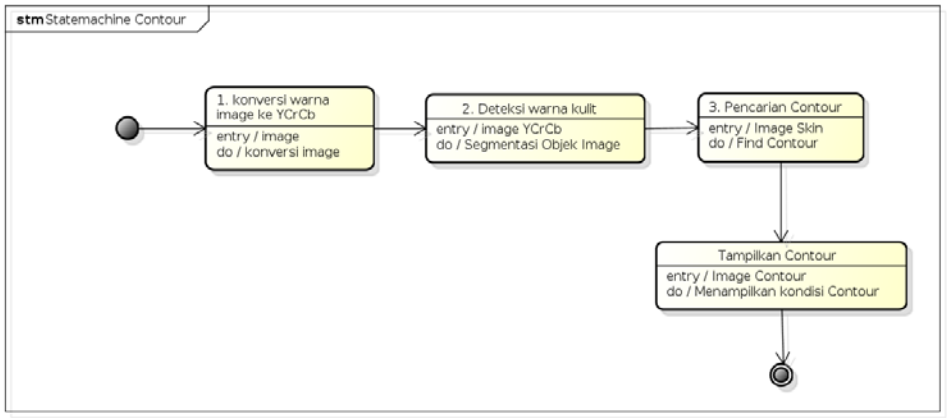


Gambar III.35 State Diagram Hand Tracking

III.1.5.9.3 State Kontur

Proses penampilan kontur dari citra yang ditangkap oleh kamera, digambarkan dalam diagram state pada gambar III.36.

1. Pengkonversian citra dari format warna RGB ke YCrCb sehingga dapat mempermudah dalam proses pendeteksian warna kulit.
2. Mendeteksi warna kulit dan melakukan segmentasi pada citra sehingga dihasilkan citra dengan warna 2 bit.
3. Pencarian kontur pada citra dengan warna 2 bit lalu melakukan penggambaran ulang pada citra tersebut dengan memunculkan kontur yang ditemukan.
4. Menampilkan kontur pada form yang dibuat dan telah ditempatkan widget untuk menampilkan kondisi citra yang sudah dikenakan komputasi untuk menemukan kontur dari tangan.



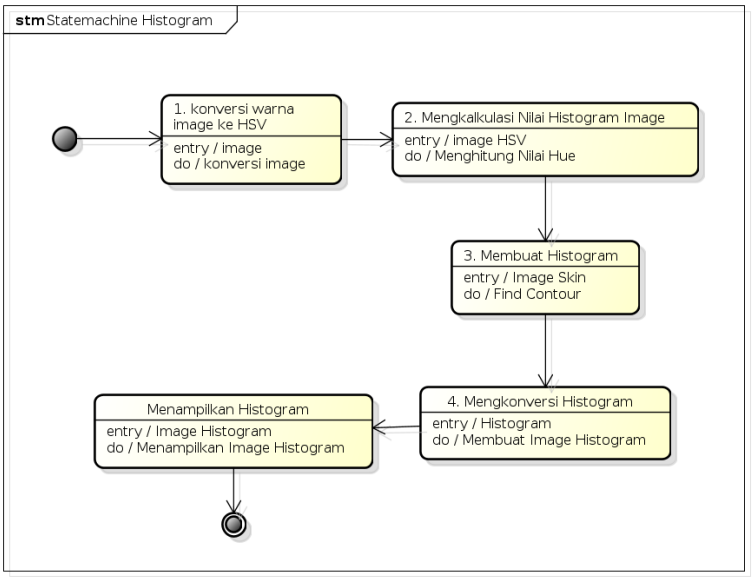
Gambar III.36 Diagram state Contour

III.1.5.9.4 State Histogram

Proses penampilan kontur dari citra yang ditangkap oleh kamera, digambarkan dalam diagram state pada gambar III.37.

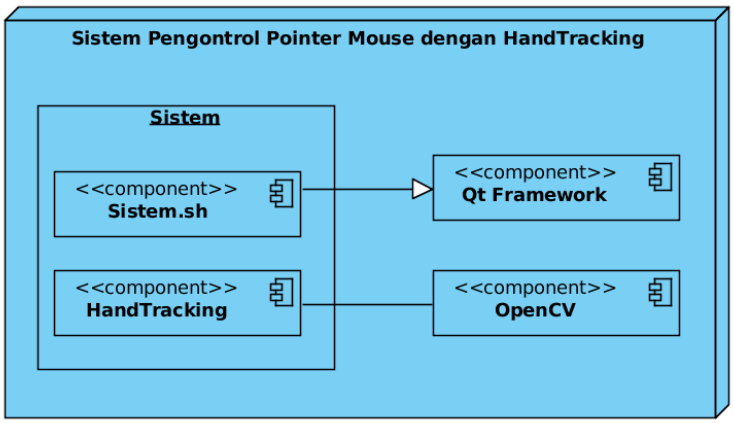
1. Dilakukan konversi warna pada citra yang didapat dari kamera, dari format warna RGB ke HSV.

2. Melakukan perhitungan atau mengkalkulasi nilai H pada citra denganb format warna HSV sehingga mendapatkan nilai warna RGB dengan rangae nilai 0-255 dengan diwakili 50 batang diagram ($\lfloor 255/50 \rfloor = 5$) sehingga 1 batang pada diagram batang nanti sama dengan 5 bit pada nilai *pixel* H.
3. Melakukan pembuatan histogram dengan mengaktifkan metode untuk melakukan komputasi untuk membuat histogram.
4. Mengkonversi histogram yang didapat kedalam diagram batang dan mekonversinya menjadi suatu citra.
5. Menampilkan histogram yang dibuat dalam form yang sudah ditambahkan widget untuk menampilkan citra yang sedang terlihat kamera dalam bentuk histogram.



III.1.5.10 Diagram Deployment

Berikut adalah diagram deploy yang dibuat dari sistem pengontrol pointer mouse dengan handtracking pada gambar III.18.



Gambar III.37 Diagram deployment

III.2 Perancangan

III.2.1 Perancangan Antar Muka

III.2.1.1 Deskripsi Layar

Secara umum tampilan utama dari sistem yang akan dibangun adalah sebagai berikut.

Gambar III.38 Tampilan Utama System

Pada gambar di atas, akan berisi layar daerah tempat *output* citra setiap *frame* yang diambil secara *real-time* dari *web-cam* dengan besar layar berukuran 320x240.Dan terdapat tombol Start dan Stop untuk memulai atau mengaktifasi sistem agar menjalankan semua fungsionalnya, dan terdapat tombol *view Hisotgram* untuk menampilkan interpretasi dari histogram setiap citra dan juga *view Threshold* untuk memperlihatkan hasil komputasi dari threshold yang dikenakan pada setiap frame.

III.2.1.1.1 Deskripsi Objek

Tabel III-12 Deskripsi Objek Antarmuka Tampilan Utama

Nama	Jenis	Keterangan
Gambar video	Image	Tampilan video real-time dan hasil pendeteksian tangan
Start	Button	Untuk memulai sistematau menjalankan proses tracking tangan
Stop	Button	Untuk menghentikan sistem atau menghentikan proses tracking tangan
Histogram	Button	Menampilkan kondisi histogram setiap frame
Contour	Button	Menampilkan kondisi threshold setiap frame
Petunjuk	Button	Menampilkan petunjuk cara pemakaian
Keluar	Button	Menghentika dan Menutup Sistem

Ketika tombol Start ditekan maka sistem akan memulai membaca *web-cam* dan mengambil gambar yang terlihat dari kamera lalu ditampilkan pada layar video lalu memulai proses hand tracking dengan melakukan identifikasi tangan dan mensingkronisasikan dengan mouse pointer yang ada. Ketika tombol *View*

Histogram ditekan maka akan memunculkan tampilan histogram dari citra yang diambil dari setiap frame yang diambil dari webcam secara *real-time*. Begitu juga dengan tombol *View Threshold*, jika tombol ini ditekan maka akan menampilkan windows citra hasil threshold berdasarkan source yang didapat dari citra hasil frame dari video real-time webcam. Jika tidak ada webcam yang terpasang dengan komputer maka akan mengirimkan pesan kesalahan.

III.2.1.1.2 Algoritma

Jika tombol Start ditekan maka sistem akan melakukan hand tracking dengan menjalankan proses algoritma sebagai berikut:

```
Deklarasi
     menu : Integer
     tampilContour : boolean
     tampilHistogram : boolean
     citra : Image
     tracking : boolean
     histogram : showHistogram
     contour : showContour
Deskripsi
     If menu >=1 then
          begin
          Tracking = true
          endif
     If menu =2 then
          begin
          tampilContour = true
          showContour
          endif
     if menu =3 then
          begin
          tampilHistogram = true
           showHistogram
          endif
```

III.2.1.2 Antarmuka View Histogram

Adapun antarmuka untuk view histogram yang dibangun adalah sebagai berikut :

Gambar III.39 Deskripsi Objek Antarmuka Tampilan Histogram

III.2.1.2.1 Deskripsi Onjek

Nama	Jenis	Keterangan
Gambar video	Image	Tampilan video real-time dan hasil interpretasi Histogram dari setiap frame.
Tutup	Button	Menutup window histogram

III.2.1.3 Antarmuka View Threshold

Berikut ini adalah Form untuk menampilkan hasil Threshold atau contour yang merupakan hasil komputasi sistem utama :

Gambar III.40 Deskripsi Objek Antarmuka Tampilan Threshols

III.2.1.3.1 Deskripsi Objek

Nama	Jenis	Keterangan
Gambar video	Image	Tampilan video real-time dan hasil komputasi Threshold dari setiap frame.
Tutup	Button	Menutup window contour.

III.2.1.4 Antarmuka Petunjuk

III.2.1.5

Gambar III.41 Deskripsi Objek Antarmuka Tampilan Petunjuk

III.2.1.5.1 Deskripsi Objek

Nama	Jenis	Keterangan
Gambar Contoh Keadaan Sistem	QLabel	Tampilan gambar yang disertai penjelasan cara pemakaian sistem.
Tutup	Button	Menutup window petunjuk.
Tab 1	QTab	Tampilan kondisi sistem saat berjalan.
Tab 2	QTab	Tampilan gambar penjelasan kondisi gesture tangan untuk menggunakan sistem.
Tab 3	QTab	Tampilan gambar penjelasan fungsi tombola tau fasilitas dari sistem.

III.2.1.6 Antarmuka Peringatan Kesalahan

Dalam menjalakan sistem, setiap input yang didapat harus dibuat valid, dalam sistem ini input didapat dari video *real-timeweb-cam* sehingga diperlukan validasi *device* untuk memastikan *web-cam* dapat digunakan dan citra yang didapat adalah citra yang valid.

Tabel III-13 Tabel pesan kesalahan device

Kode Pesan	Antarmuka Kesalahan
ERROR	

III.2.2 Jaringan Semantik

Alur antarmuka yang terdapat pada sistem pengontrol mouse pointer dengan handtracking, dapat digambarkan dalam jaringan pada gambar III.39.

Gambar III.42 Jaringan Semantik Sistem