BAB V
PERANCANGAN SISTEM

5.1. Metode Perancangan Sistem

Metode perancangan yang dipakai untuk membangun aplikasi pendeteksi wajah ini ini adalah metode perancangan berorientasi obyek. Proses pembuatan aplikasi dimulai dari perancangan sistem menggunakan UML yang akan menghasilkan Use Case diagram, Class diagram , dan Sequence diagram.

Metode ini digunakan karena *framework* openFrameworks dan OpenCV yang digunakan menggunakan konsep berorientasi obyek, dan juga karena metode ini mempunyai banyak kelebihan daripada metode prosedural. Kelebihan dari metode analisis berorientasi obyek adalah :

1. Kode-kode pemrograman dibagi-bagi menjadi unsur yang disebut objek, dan objek ini menangani tugas masing-masing sehingga pengaturan dan perubahan kode serta fungsionalitasnya menjadi lebih mudah.
2. Penggunaan kembali kode yang telah dibuat, karena berupa objek.
3. Pemeliharaan dan perluasan software ke depannya menjadi lebih mudah, dengan menggunakan metode ini.

Diharapkan dengan analisis yang digunakan ini dapat mempermudah dalam pengembangan aplikasi *computer vision* khususnya aplikasi pendeteksi wajah, dan juga pengembangannya ke depan.

5.2. Perancangan Sistem

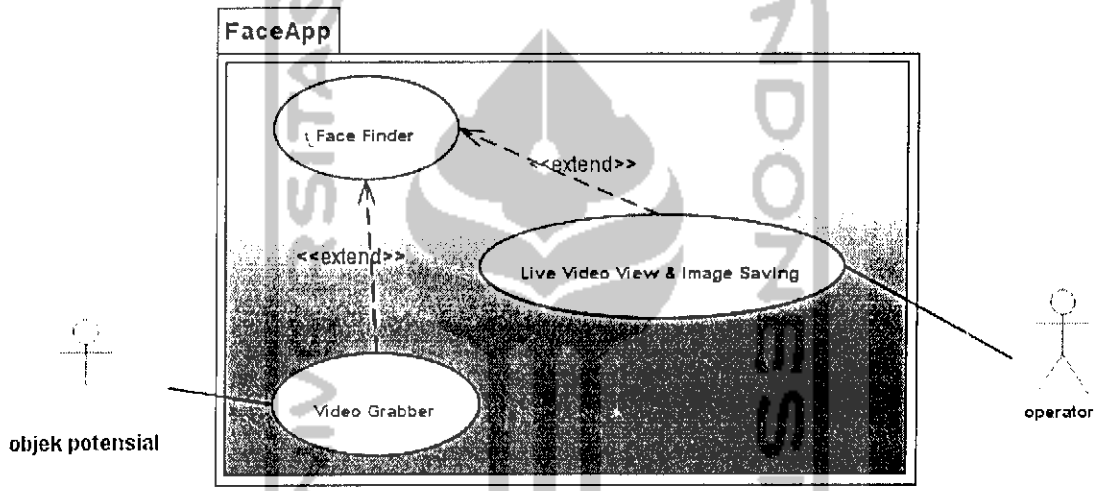
Dari hasil perancangan dengan menggunakan metode UML menghasilkan :

- Use case diagram
- Sequence diagram
- Activity diagram
- Class diagram

Pada perancangan sistem ini juga dibuat suatu *pseudo-code* untuk membantu pada implementasi sistem.

5.2.1 Use Case Diagram

Diagram ini digunakan untuk menggambarkan fungsi-fungsi yang dibutuhkan oleh sebuah sistem dan pada bagian ini diagram akan lebih digunakan untuk merepresentasikan interaksi antara pengguna dan sistem. Secara umum, *use case diagram* menunjukkan hubungan antara sistem dan objek-objek yang ada didalamnya dengan aktor-aktor atau objek di luar sistem.



Gambar 5.1 – Use Case Diagram

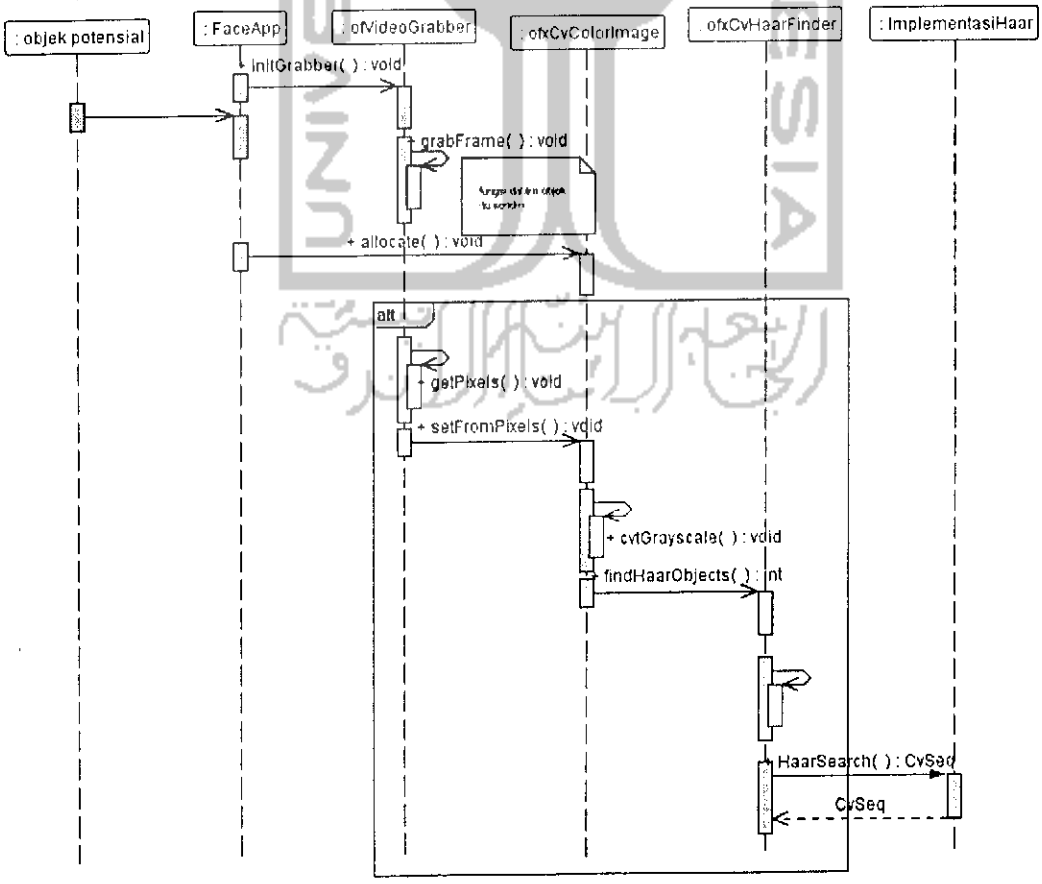
Pada kasus pengembangan aplikasi ini, *use case* diagram yang dibuat menggambarkan proses yang terjadi antara aktor objek potensial deteksi, dengan fungsi-fungsi pada aplikasi, dan juga hubungan aplikasi dengan operator yang menangani aplikasi ini.

5.2.2 Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan perilaku pada suatu skenario. Diagram ini menunjukkan sejumlah obyek dan message (pesan) yang diletakkan diantara obyek-obyek ini dalam *use-case*. Dalam perancangan sistem ini, dibuat banyak *sequence diagram* sesuai skenario yang dimaksud.

5.2.2.1 Sequence Diagram Use Case Video Grabber

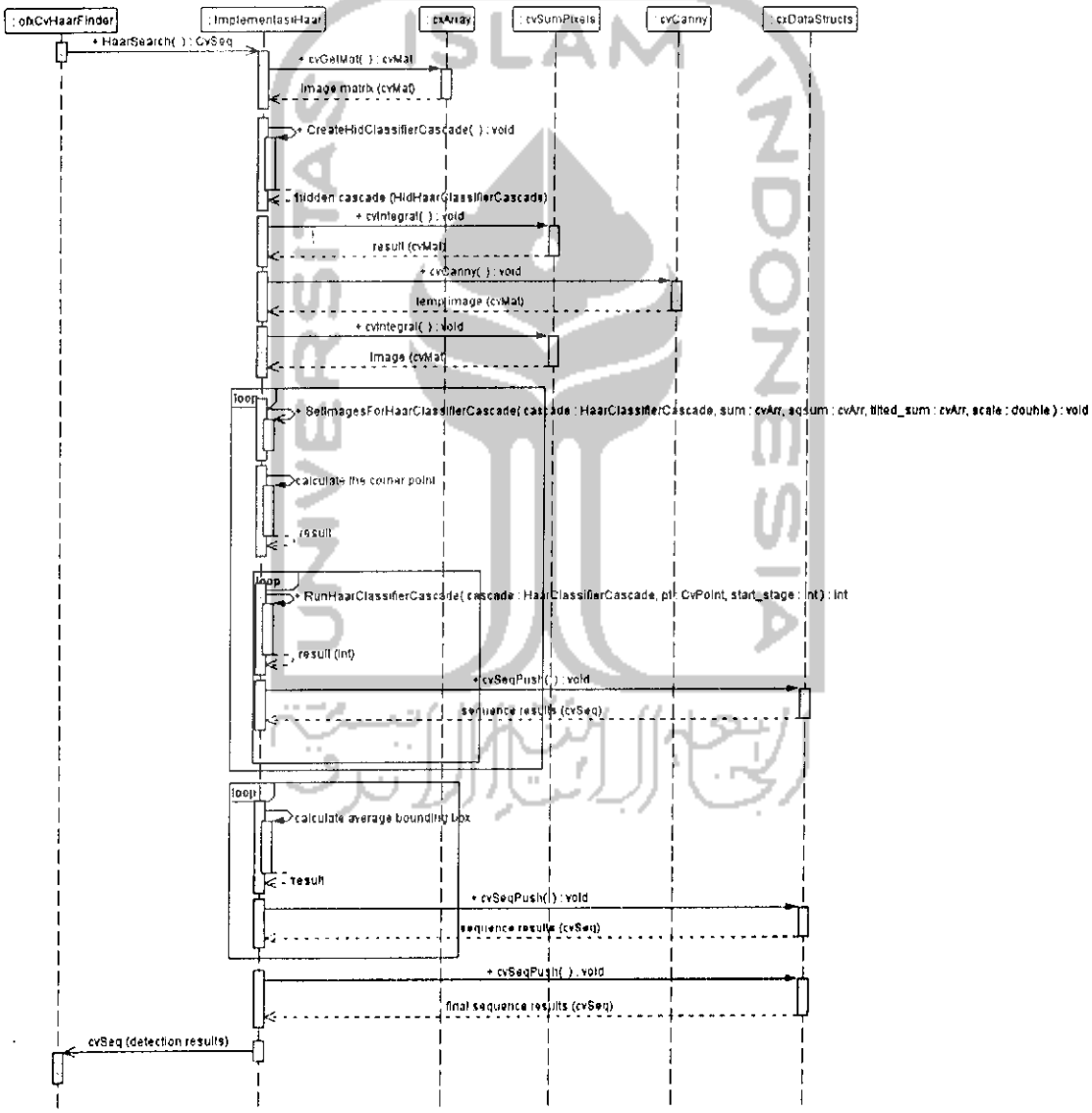
Sequence diagram pada gambar 5.2 menunjukkan proses yang terjadi pada saat pengambilan citra video sampai pemrosesan per-frame sebelum dilakukan pencarian objek wajah, sesuai dengan *use case* Video Grabber.



Gambar 5.2 - Sequence Diagrram Use Case Video Grabber

5.2.2.2 Sequence Diagram Use Case Face Finder

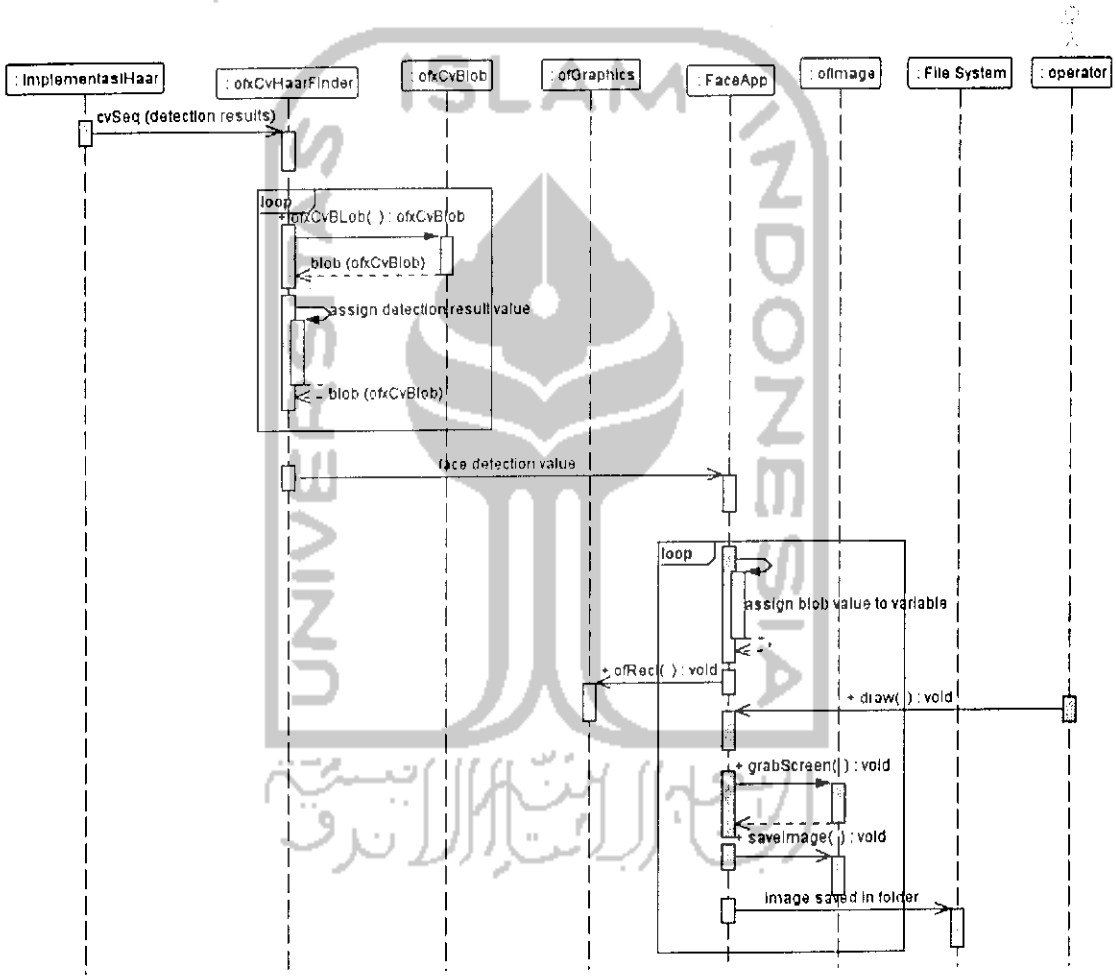
Sequence diagram pada Gambar 5.3 menunjukkan alur yang terjadi pada proses pendeteksian objek wajah, yaitu mulai dari citra per-frame diterima proses pendeteksian objek pada citra tersebut dan kemudian melempar hasilnya ke fungsi yang lain untuk kemudian diproses lebih lanjut, sesuai dengan *use case* Face Finder.



Gambar 5.3 – Sequence Diagram Use Case Face Finder

5.2.2.3 Sequence Diagram Use Case Live Video View dan Image Saving

Sequence diagram pada Gambar 5.4 menunjukkan proses akhir dari pendeteksian objek dan menandainya ke layar. Fungsi yang ada menerima nilai balikan dari kelas yang bertugas mendeteksi objek dari citra per-frame, dan kemudian menggambar persegi panjang pada lokasi ditemukannya objek wajah pada layar *live video*, sesuai dengan *use case* Live Video View dan Image Saving.

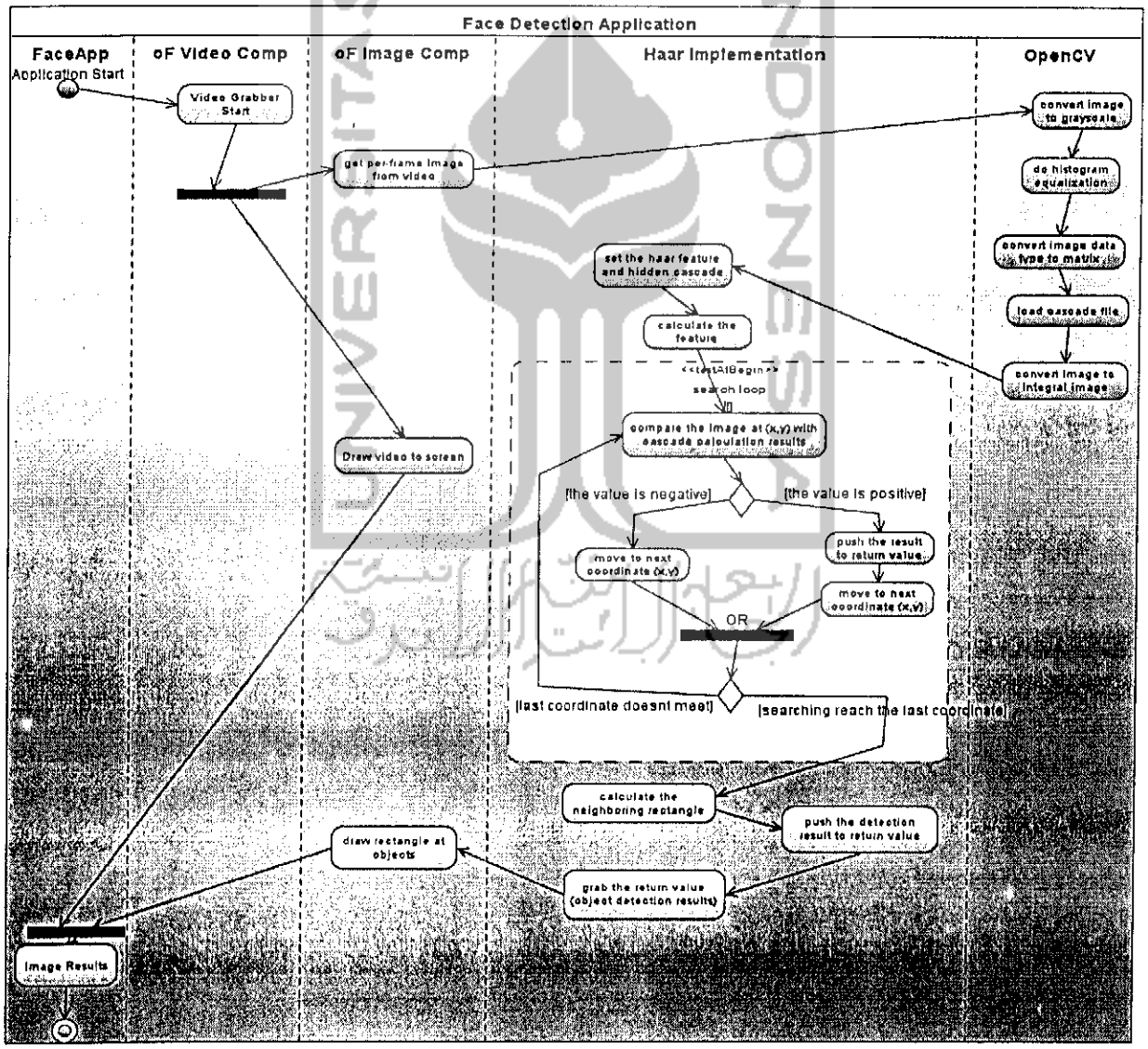


Gambar 5.4 – Sequence Diagram Use Case Live Video View & Image Saving

5.2.3 Activity Diagram

Activity Diagram digunakan ketika perancang ingin menggambarkan alur dari kelas-kelas yang terlibat secara non-teknis dan logika proses. *Activity Diagram* ini mirip dengan *data flow diagram* tradisional, tetapi dapat menggambarkan proses yang terjadi paralel. *Activity diagram* dapat digunakan untuk menunjukkan perilaku

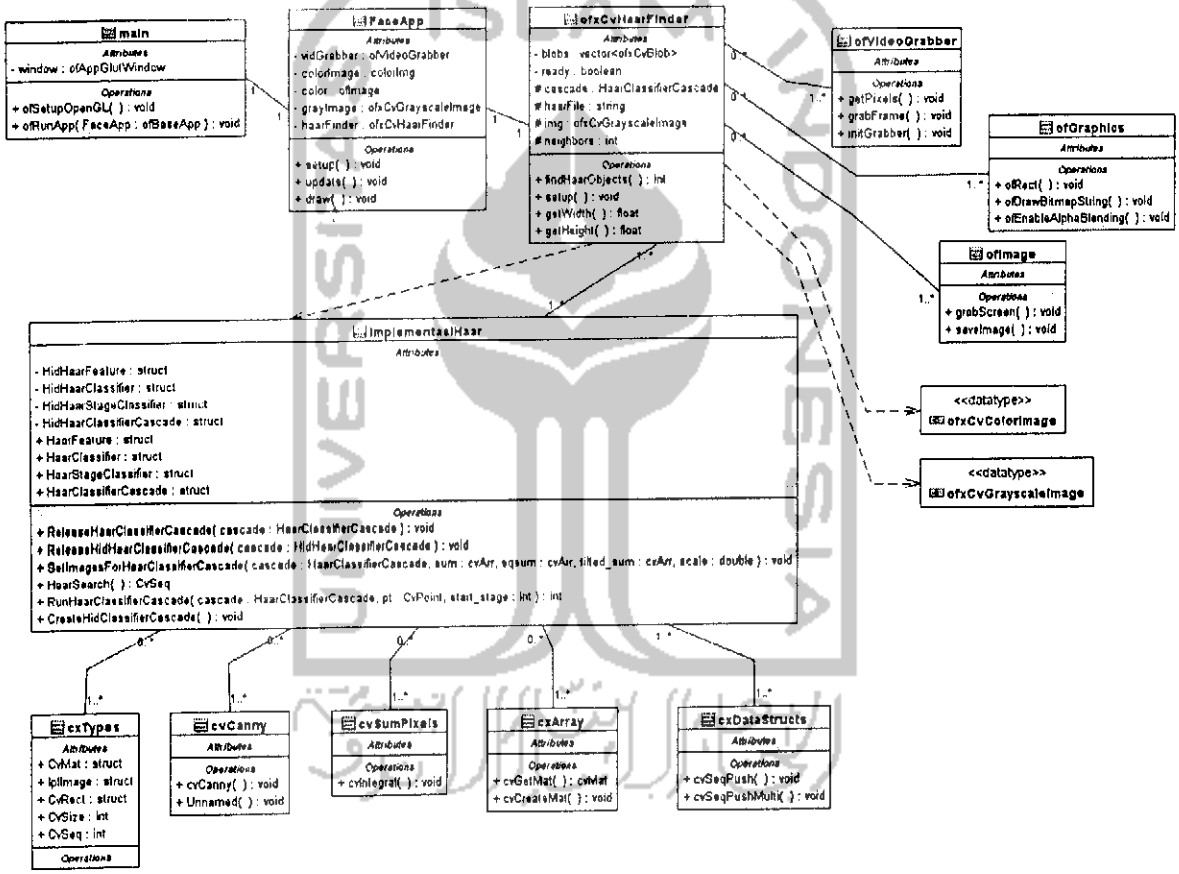
dari banyak kelas dan alir dari objek, data, atau alir kendali dari banyak kelas yang berbeda. Gambar 5.5 dibawah menunjukkan alur yang terjadi pada proses aplikasi deteksi wajah dan objek-objek yang berperan didalamnya.



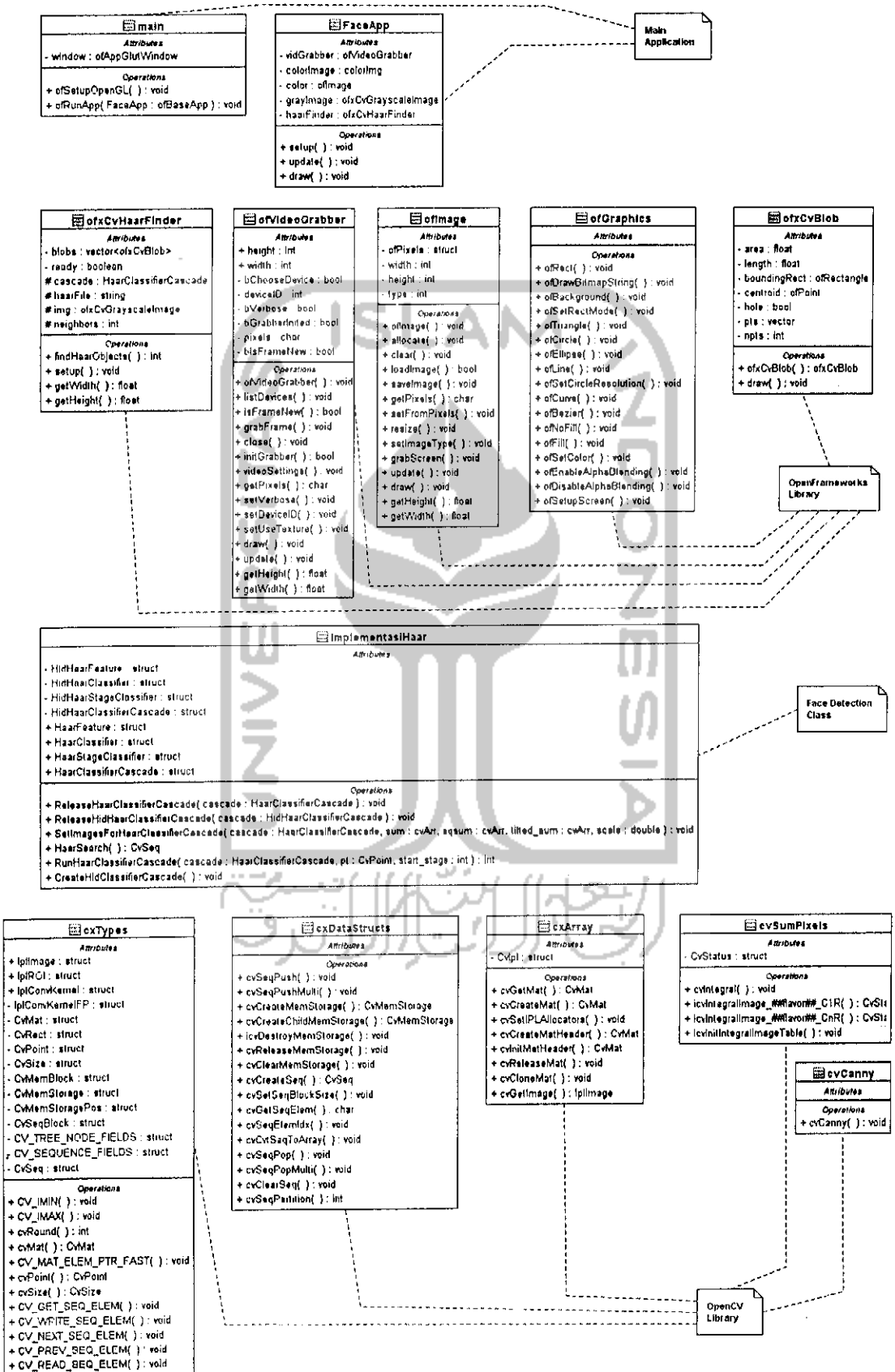
Gambar 5.5 – Activity Diagram

5.2.4 Class Diagram

Class diagram menunjukkan kelas-kelas yang terlibat pada keseluruhan proses secara teknis dan juga atribut-atribut serta operasi yang ada dalam kelas masing-masing. Diagram ini juga menggambarkan *multiplicity* yang ada pada kelas-kelas yang saling berhubungan, seperti tertera pada gambar 5.6. Kemudian gambar 5.7 menunjukkan atribut dan operasi yang ada pada diagram itu secara detail.



Gambar 5.6 – Class Diagram



Gambar 5.7 – Class Diagram Detail

5.2.5 Pseudo Code Fungsi

Pada perancangan ini juga dibuat suatu *pseudo-code* dari fungsi-fungsi yang terdapat pada file kelas `ImplementasiHaar.cpp`, dimana kelas ini adalah kelas yang menangani pendeteksian objek.

5.2.5.1 Fungsi `SetImageForHaarClassifierCascade`

Pseudo-code fungsi :

```
1. Mengubah citra inputan menjadi suatu array matrix dengan variabel cvMat.
2.membuat hidden cascade dengan memanggil fungsi CreateHidClassifierCascade() jika
hidden cascade pada cascade aslinya kosong.
3.mengisikan variabel 'sum' yang merupakan citra input ke dalam cascade asli,di var
'sum' juga.
4.membuat suatu persegi panjang bertipe cvRect. Mengatur ukuran real_window_size dari
cascade dengan disesuaikan skala yang telah didefinisikan di parameter fungsi.
5.mengisi variabel p0,p1,p2,p3 dari hidden cascadenya dan juga pq0,pq1,pq2,pq3.dgn
penghitungan :
        5.1.pastikan nilai penskalaan original window size dari real cascade lebih
kecil dari nilai integral imagenya.
        5.2.kemudian isi nilai dengan jumlah total panjang memory dan data pointer
dari sum dikali ukuran original window size,seperti ini:
                5.2.1.(mat).data.ptr + (size_t)(mat).step*(row) + (pix_size)*(col)
6.isikan elemen p0,p1,p2,p3 dari nilai persegi panjang node fitur hidden cascade
dengan penghitungan :
        6.3.hampir sama dengan langkah 5,parameternya tr(poin x & y rectangle dan
ukuran width & heightnya).
        6.4.Melakukan penghitungan poin x & y persegi panjang region :
                6.4.5.tr.x = r[k].x*scale
                6.4.6.tr.width = r[k].width*scale
                6.4.7.tr.y = r[k].y*scale
                6.4.8.tr.height = r[k].height*scale
7.isikan elemen pq0,pq1,pq2,pq3 dari nilai persegi panjang node fitur hidden cascade.
8.isi weight ke-[k] dari feature hidden node classifier dengan weight sebelumnya
dikalikan ((1/(equ_rect.width*equ_rect.height)) * 1).
9.isi weight ke-[0] dengan -sum0/area0
        9.1.isikan sum0 dengan weight ke-[k] sebelumnya * tr.width * tr.height
        9.2.isikan area0 dengan tr.width * tr.height
        9.3.isi tr.height & tr.width dengan rectangle ke-[k] dari fitur asli * scale.
10.hasil akhirnya adalah poin-poin elemen rectangle hidden cascade.
```

5.2.5.2 Fungsi `RunHaarClassifierCascade`

Pseudo-code fungsi :

```
1.mengecek apakah poin mulai (x,y) dimana fungsi ini berjalan tidak kurang dari 0
```

2.mengecek apakah real window size dari cascade tidak lebih besar dari matrix 'sum' dari hidden cascade.
3.menghitung rerata dengan penghitungan :
(p0(dari hidden cascade)[offset]-p1[offset]-p2[offset]+p3[offset]) * (1/(_cascade->orig_window_size.width-2)*scale)*(_cascade->orig_window_size.height-2)*scale)).
Offset adalah hasil penskalaan.
3.jika cascade adalah tree :

3.1.tentukan nilai sum dari penghitungan empat poin rectangle hidden node dan tentukan apakah melewati node kiri atau kanan.

3.2.kemudian jika nilainya lebih besar atau mencapai threshold,maka lanjutkan ke classifier child.

3.3.Jika tidak mencapai threshold, periksa jika penunjuk ke next-nya null,

3.4.jika null maka isi dengan nilai penunjuk ke stage classifier parent.kemudian isi penunjuk dengan penunjuk ke stage classifier next.

4.jika cascade adalah stump (ujung) :

4.1.isi sum dengan penghitungan poin-poin pojok rectangle hidden node [0] dan [1] dikalikan bobot masing-masing.

4.2.jika ada nilai rectangle ke[2],isikan juga ke sum.isi sum dari hidden stage classifier dengan sum hidden stage classifier sebelumnya ditambah : (nilai alpha ke-[0] hidden classifier jika 'sum' lebih kecil dari threshold,dan alpha ke-[1] jika sebaliknya).

4.3.kemudian jika penghitungan stage sum tersebut lebih kecil daripada threshold hidden stage classifier,maka nilai balikan fungsi ini adalah - (minus).

5.jika bukan tree atau stump : isi stage sum dengan penghitungan seperti langkah 3 dan jika lebih kecil dari threshold hidden stage classifier, maka berikan nilai balikan - (minus) dan keluar fungsi.
6.hasil akhirnya adalah nilai integer positif atau negatif.

5.2.5.3 Fungsi HaarSearch

Pseudo-code fungsi :

1.melakukan konversi ke integral image
2.melakukan canny pruning jika diperlukan.
3.melakukan penskalaan dari citra objek wajah yang telah ditentukan dalam file setting xml,(20x20),
4.sebelum melakukan pencarian, memanggil fungsi SetImagesForHaarClassifierCascade untuk menyiapkan cascade dan hidden cascade yang nantinya digunakan pada pencarian.
5.menentukan titik koordinat awal dimana harus dilakukan deteksi dan juga titik koordinat dimana harus menghentikan pencarian.
6. melakukan penghitungan pada 4 titik poin dari persegi panjang original window dari objek,yang tertera di file xml.
7.menentukan koordinat awal dan koordinat akhir,serta nilai 4 poin referensi array yang akan digunakan untuk penghitungan 'sum' integral image untuk memperoleh penjumlahan area dalam region tersebut.
8.fungsi mulai melakukan perulangan untuk mendeteksi objek :

8.1.memanggil fungsi RunHaarClassifierCascade dengan parameternya adalah cascade yang telah tersedia dan koordinat posisi dimana fungsi ini berjalan.

8.2.jika fungsi RunHaarClassifierCascade memberikan hasil positif, maka region yang ditandai akan disimpan ke suatu variabel sequence, dan akan pencarian akan dilanjutkan lagi dengan perulangan.

9.beberapa region yang lolos klasifikasi ditandai koordinat posisi dan ukurannya.

10.region persegi panjang yang saling berdekatan akan disatukan menjadi satu persegi panjang dan dianggap sebagai satu kesatuan objek.

11.nilai-nilai penghitungan langkah 10 akan dimasukkan ke suatu variabel nilai balikan berupa sequence.