BAB 3
ANALISIS DAN PERANCANGAN

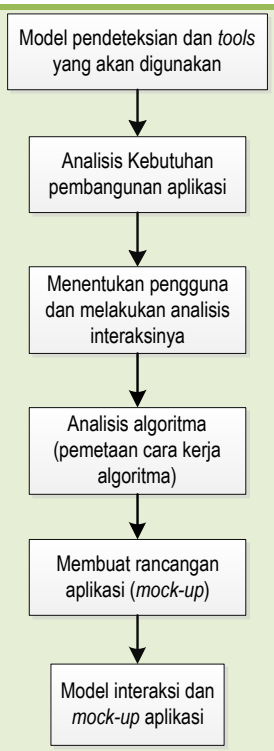
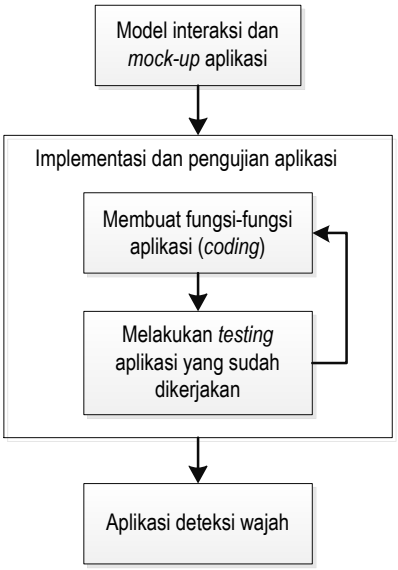
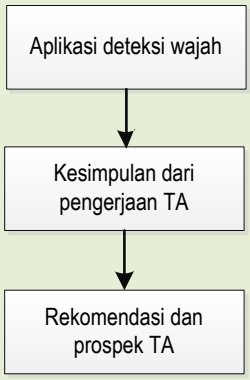
Pada bab ini berisi tentang analisis dan perancangan terhadap permasalahan yang sedang diteliti seperti analisis kebutuhan data dan informasi serta teknik dan peralatan yang digunakan dalam perancangan aplikasi.

3.1 Kerangka Tugas Akhir

Tabel 3.1 Kerangka Tugas Akhir

Tahap & Hasil	Langkah Penelitian	Literatur & Referensi
Tahap 1 : Mencari informasi tentang deteksi wajah. (Memahami konsep deteksi wajah, memahami metode pengembangan aplikasi, memahami *tools* pengenmbangan aplikasi deteksi wajah). Hasil : Menentukan algoritma pendeteksian. Kontribusi : Berguna untuk perancangan dan implementasi pembangunan aplikasi.	Memahami konsep deteksi wajah ↓ Analisis kebutuhan yang akan digunakan dalam pembangunan aplikasi ↓ Menentukan algoritma pendeteksian	1. Face Detector [NUR12]. 2. Sistem Pendeteksi Wajah Manusia Pada Citra Digiral [NUG04]. 3. Metode Pengembangan Sistem Waterfall [KAD03]. 4. Bagaimana Image Processing Mendapatkan Wajah dari Sebuah Foto [LIS12]. 5. Robust Real-TIme face Detection [VIO01].
Tahap 2 : Pengumpulan data. (Mengumpulkan data yang diperlukan berdasarkan studi literatur dan referensi). Hasil : Model pendeteksian dan *tools* untuk pembangunan aplikasi. Kontribusi : Berguna untuk perancangan dan implementasi pembangunan aplikasi.	Menentukan algoritma pendeteksian ↓ Memahami model deteksi wajah *Viola-Jones* ↓ Mencari *tools* yang akan digunakan ↓ Analisis hubungan antara algoritma *Viola-Jones* dengan *tools* pembangunan aplikasi ↓ Model pendeteksian dan *tools* yang akan digunakan	1. Sistem Deteksi Wajah dengan Menggunakan Metode *Viola-Jones* [DWI12]. 2. *Face Detection* Dengan Metoda *Haar-Cascade* [LIE12]. 3. Pengertian dan Sejarah Matlab [ROH14]. 4. Membuat aplikasi pengolahan citra menggunakan *openCV* [ARI13]. 5. Deteksi Wajah Metode *Viola Jones* Pada *Opencv* Menggunakan Pemrograman *Python* [ARY12].

Tabel 3.1 Kerangka Tugas Akhir

Tahap & Hasil	Langkah Penelitian	Literatur & Referensi
Tahap 3 : Melakukan analisis dan membuat rancangan aplikasi. (Menentukan pengguna, menentukan model interaksi antara pengguna dan aplikasi, membuat *mock-up*) Hasil : Model interaksi dan *mock-up* aplikasi. Kontribusi : Untuk mengetahui interaksi antara *user* dengan aplikasi, untuk implementasi pembangunan aplikasi.		1. Adaboost [HEN12]. 2. Haar-Cascade dan Adaboost [HAD13]. 3. Membuat *GUI* di Matlab [PUT12]. 4. Membuat *GUI* Pada Matlab 7 [HAR13].
Tahap 4 : Melakukan implementasi dan pengujian. (Penulisan kode program, pembuatan fungsi-fungsi aplikasi, *testing* aplikasi yang sudah dikerjakan). Hasil : Aplikasi deteksi wajah menggunakan algoritma *Viola-Jones.* Kontribusi : Untuk menarik kesimpulan dari hasil yang sudah dikerjakan.		1. Pengolahan Citra Digital Menggunakan Matlab [WIJ07]. 2. Pengantar Komputasi Numerik dengan Matlab [SAH05]. 3. Teknik pemrograman Matlab [MAT14]
Tahap 5 :Kesimpulan TA Hasil : Kesimpulan dan hasil dari pengerjaan TA, rekomendasi dan prospek TA Kontribusi : -		-

3.2 Analisis

Analisis dilakukan untuk mengurai secara mendalam terhadap perancangan yang akan dilakukan. Pada perancangan aplikasi deteksi wajah ini juga dilakukan analisis seperti analisis kebutuhan dan analisis interaksi.

3.2.1 Analisis Kebutuhan

Sesuai dengan metodologi pengembangan perangkat lunak yang digunakan pada tugas akhir ini, maka tahapan pertama yang dilakukan adalah analisis terhadap kebutuhan yang diperlukan. Analisis kebutuhan merupakan langkah awal untuk menentukan aplikasi seperti apa yang akan dibuat, ketika akan membuat sebuah aplikasi. Dalam perancangan aplikasi deteksi wajah ini, dibutuhkan banyak informasi mengenai pendeteksian wajah seperti teknik serta peralatan yang digunakan untuk merancang sebuah aplikasi deteksi wajah.

Dalam merancang aplikasi deteksi wajah dibutuhkan sebuah algoritma yang akan diterapkan pada aplikasi yang akan dirancang. Pada perancangan aplikasi deteksi wajah ini menggunakan sebuah algoritma pendeteksian yaitu algoritma *Viola-Jones*. Aplikasi deteksi wajah secara keseluruhan akan mengikuti cara kerja dan teknik yang ada pada algoritma *Viola-Jones*. Selain sebuah algoritma, kebutuhan lain yang diperlukan dalam perancangan aplikasi deteksi wajah ini adalah sebuah perangkat lunak yang sesuai untuk merancang aplikasi deteksi wajah. Perangkat lunak yang dibutuhkan dalam merancang aplikasi deteksi wajah ini adalah perangkat lunak pengolahan citra atau yang disebut *image processing*.

Selain kebutuhan perangkat lunak, perangkat lain yang tentunya dibutuhkan dalam perancangan aplikasi deteksi wajah ini adalah sebuah perangkat keras untuk merancang aplikasi tersebut. Adapun spesifikasi dari perangkat keras yang digunakan pada perancangan aplikasi deteksi wajah ini dapat dilihat pada tabel 3.2.

Tabel 3.2 Spesifikasi Perangkat Keras

Perangkat Keras	Spesifikasi
Processor	AMD Turion II X2 M500
Memory	2Gb
Video Card	ATI Radeon HD 4200
Harddisk	250Gb
Web Camera	Acer Crystal Eye
Monitor	LED 14 Inch

3.2.2 Fungsional Aplikasi

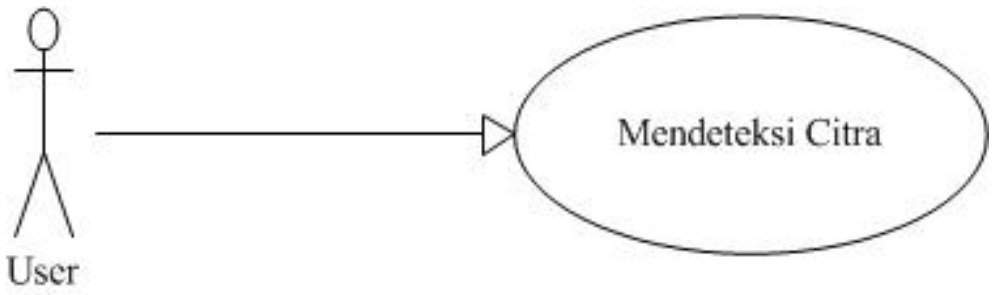
Berikut ini merupakan fungsional dari aplikasi deteksi wajah yang akan dirancang, yaitu :

1. Aplikasi deteksi wajah dapat menangkap citra yang berasal dari kamera yang terhubung dengan aplikasi.
2. Aplikasi deteksi wajah dapat memasukan *file* JPEG dari direktori komputer ke dalam aplikasi.

3. Aplikasi deteksi wajah dapat mendeteksi wajah dari citra yang ditangkap oleh kamera ataupun dari citra masukan komputer.
4. Aplikasi deteksi wajah dapat menampilkan citra dan informasi hasil pendeteksian kepada *user.*

3.2.3 Analisis Interaksi

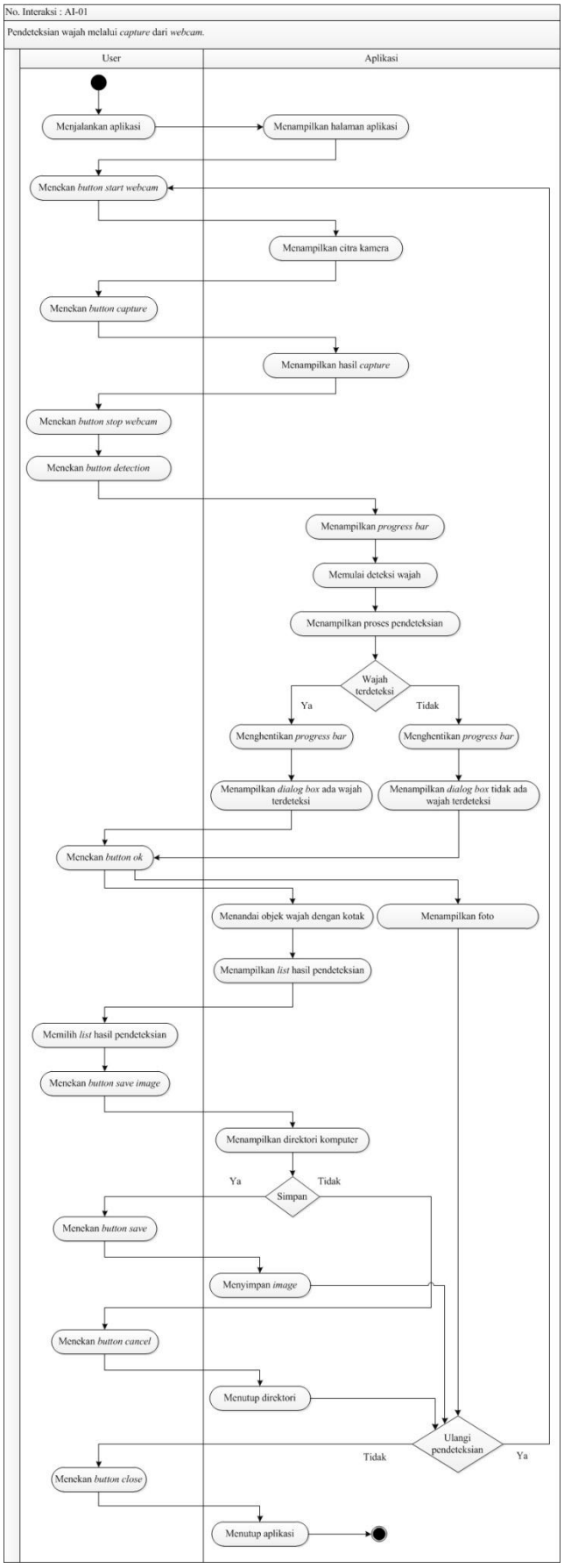
Pada perancangan aplikasi deteksi wajah ini, diidentifikasikan hanya ada satu aktor yang terlibat dalam proses pendeteksian yaitu *user*. Dalam interaksinya *user* berhubungan langsung dengan aplikasi deteksi wajah. Proses utama dalam interaksi ini adalah *user* menjalankan aplikasi deteksi wajah untuk mendeteksi citra. Citra yang akan dideteksi yaitu hanya citra wajah dari objek yang tertangkap oleh kamera ataupun citra masukan dari komputer saja. *Use-case diagram* dari interaksi deteksi wajah dapat dilihat pada gambar 3.1.



Gambar 3.1 *Use-case Diagram* Deteksi Citra

Diagram *use-case* di atas menggambarkan bahwa *user* menjalankan aplikasi deteksi wajah dan selanjutnya aplikasi akan mendeteksi citra. Sesuai dengan fungsional aplikasi, dalam proses pendeteksian pada sebuah citra maka aplikasi akan menangkap citra menggunakan kamera yang terhubung dengan aplikasi ataupun melalui citra masukan dari komputer. Apabila citra yang akan dideteksi sudah didapatkan, maka proses selanjutnya adalah aplikasi akan melakukan proses pendeteksian dari citra yang sudah didapatkan. Objek utama dalam proses pendeteksian ini adalah wajah, jika terdapat wajah yang terdeteksi maka, aplikasi akan menampilkan informasi kepada *user* bahwa ada wajah yang terdeteksi pada citra yang dideteksi begitu juga sebaliknya. *Flowmap Diagram* dari setiap interaksi antara *user* dan aplikasi dapat dilihat pada gambar 3.2 dan gambar 3.3.

1. Pendeteksian wajah melalui *capture* dari *webcam.*



Gambar 3.2 *Activity Diagram* pendeteksian wajah melalui *capture* dari *webcam*

Pada gambar 3.2 menggambarkan interaksi antara *user* dengan aplikasi, pada saat *user* melakukan deteksi wajah dengan menggunakan kamera yang terhubung dengan aplikasi. Pada saat aplikasi dijalankan oleh *user*, maka aplikasi akan menampilkan antarmuka dari aplikasi. Selanjutnya untuk melakukan deteksi wajah maka *user* harus menekan *button start webcam* untuk melakukan proses pendeteksian melalui kamera.

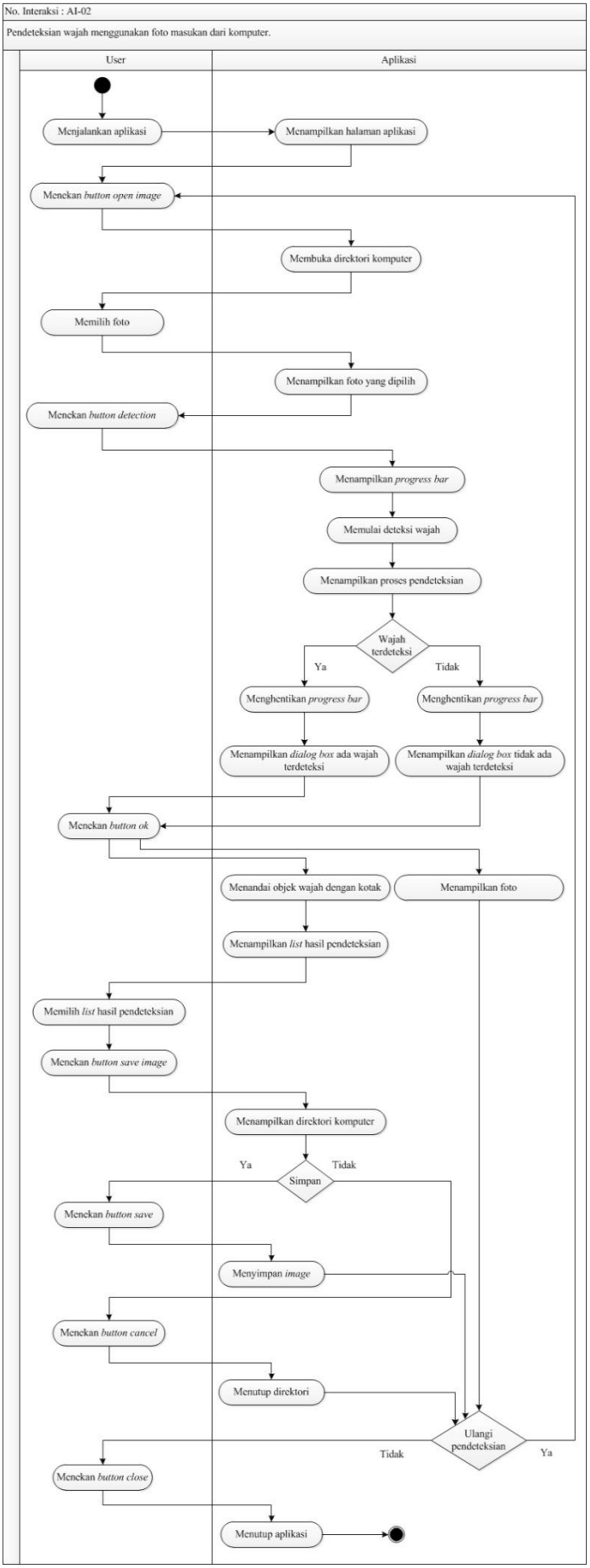
Untuk menangkap foto dari citra kamera maka *user* harus menekan *button capture,* maka aplikasi selanjutnya akan menampilkan hasil *capture* pada panel *input*. Untuk memulai proses pendeteksian wajah, *user* harus menekan *button detection*, selanjutnya aplikasi akan menampilkan sebuah *progress bar* yang menandakan bahwa proses deteksi wajah sedang berjalan. Selama proses pendeteksian berjalan, aplikasi juga akan menampilkan proses tersebut pada panel *detection process.* Apabila proses pendeteksian sudah selesai, maka aplikasi akan menghentikan *progress bar* dan aplikasi akan menampilkan informasi bahwa ada wajah yang terdeteksi atau tidak. Jika ada wajah yang terdeteksi maka aplikasi akan menampilkan informasi bahwa ada wajah yang terdeteksi oleh aplikasi dan sebaliknya jika tidak ada wajah yang terdeteksi, maka aplikasi akan menampilkan informasi bahwa tidak ada wajah yang terdeteksi oleh aplikasi dan kamera akan kembali aktif dan menampilkan citra kamera.

Apabila pada saat aplikasi berhasil mendeteksi wajah tetapi *user* ingin kembali melakukan proses deteksi dengan melakukan *capture* dengan objek lain, maka user harus menekan *button start webcam* untuk mengaktifkan kembali kamera. Untuk menutup aplikasi user harus menekan *button close.*

2. Pendeteksian wajah menggunakan foto masukan dari komputer.

Pada gambar 3.3 digambarkan interaksi antara *user* dengan aplikasi untuk deteksi wajah dengan menggunakan foto masukan dari komputer. Untuk melakukan deteksi wajah dengan foto masukan maka *user* harus menekan *button open image* pada aplikasi. Selanjutnya aplikasi akan membuka dan menampilkan direktori komputer. Apabila direktori sudah muncul maka *user* bisa memilih foto masukan untuk proses pendeteksian. Setelah foto dipilih, aplikasi akan menampilkan foto tersebut pada panel *input* pada aplikasi. Selanjutnya, *user* harus menekan *button detection* untuk melakukan proses deteksi wajah pada foto yang sudah dimasukkan. Selanjutnya aplikasi akan menampilkan *progress bar* dan proses deteksi wajah akan dilakukan oleh aplikasi. Selama proses pendeteksian berjalan, aplikasi juga akan menampilkan proses tersebut pada panel *detection process.* Apabila proses pendeteksian sudah selesai maka *progress bar* akan berhenti dan akan tampil sebuah *dialog box* yang menampilkan informasi hasil pendeteksian.

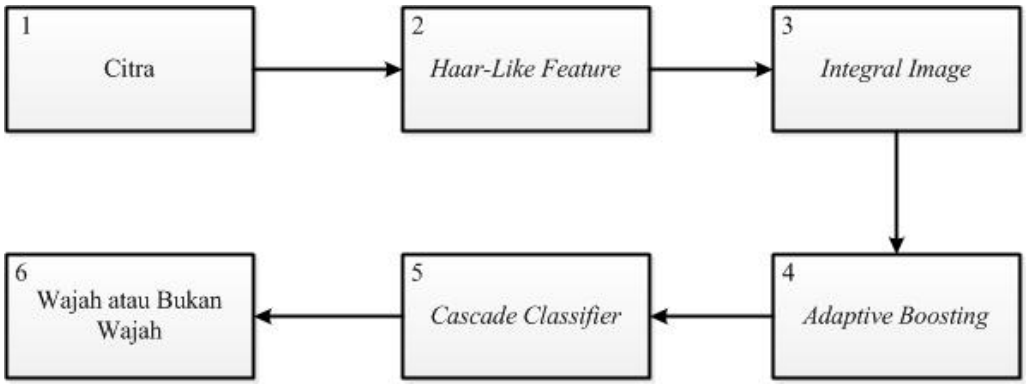
Aplikasi akan menampilkan pesan apakah ada wajah yang wajah terdeteksi atau tidak pada *dialog box* yang ditampilkan. Apabila user ingin kembali melakukan pendeteksian dengan menggunakan foto yang berbeda maka user kembali harus menekan *button open image* untuk kembali memilih *file* foto yang akan dideteksi. Untuk menutup aplikasi user bisa menekan *button close*.



Gambar 3.3 *Activity Diagram* Pendeteksian wajah menggunakan foto masukan dari komputer.

3.2.4 Algoritma

Pada proses pendeteksian wajah dengan menggunakan algoritma *Viola-Jones*, ada beberapa proses yang dilakukan sebelum akhirnya akan menghasilkan sebuah *output* wajah yang terdeteksi pada sebuah citra. Dalam deteksi wajah *Viola-Jones*, proses-proses tersebut yaitu *Haar-Like Featrure*, *Integral image*, *Adaboost* (*Adaptive Boosting*), dan *Cascade Classifier* seperti pada skema proses deteksi wajah yang dijelaskan pada bab sebelumnya. Skema proses dari tiap-tiap tahap yang dilalui oleh sebuah citra untuk memperoleh hasil pendeteksian wajah dapat dilihat pada gambar 3.4 [DWI12] .



Gambar 3.4 Skema Deteksi *Viola-Jones* [DWI12]

Untuk detail dari tiap tahap yang dilalui oleh sebuah image pada saat proses pendeteksian wajah seperti pada gambar 3.4 diatas adalah sebagai berikut :

1. Pemilihan fitur

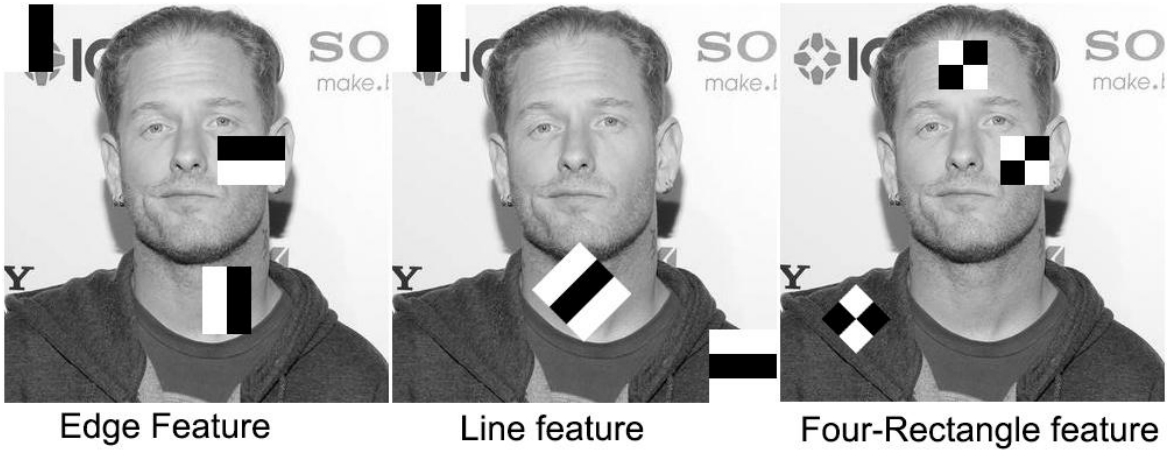
a. *Haar-like feature*

Untuk mendeteksi adanya fitur wajah pada sebuah *image*, proses pertama yang dilakukan oleh algoritma *Viola-Jones* adalah dengan merubah *image* tersebut menjadi citra *grayscale*. Contohnya *haar-like feature* dapat dilihat pada gambar 3.5.

Gambar 3.5 Contoh Perubahan Citra RGB *Image* Menjadi *Grayscale*

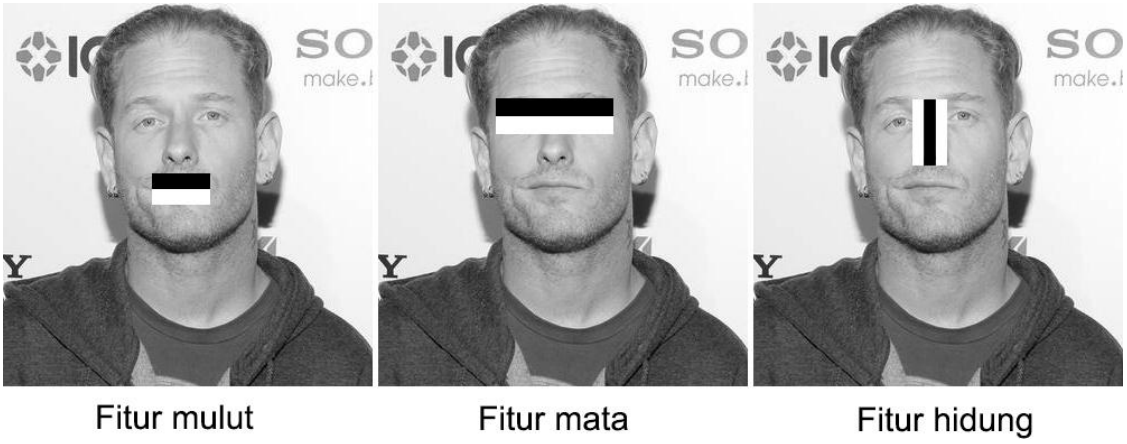
Setelah citra *image* dirubah menjadi citra *grayscale*, proses selanjutnya yaitu memilih fitur *Haar* yang ada pada *image* tersebut yang dalam algoritma *Viola-Jones* disebut dengan *Haar-Like*

feature. Teknik yang dilakukan yaitu dengan cara mengkotak-kotakkan setiap daerah pada *image* dari mulai ujung kiri atas sampai kanan bawah. Proses ini dilakukan untuk mencari apakah ada fitur wajah pada area tersebut. Dalam algoritma *Viola-Jones*, ada beberapa jenis fitur yang bisa digunakan seperti *Edge-feature, Line feature, dan Four-rectangle feature.* Pada proses pemilihan fitur Haar, fitur-fitur tersebut digunakan untuk mencari fitur wajah seperti mata, hidung, dan mulut. Pada setiap kotak-kotak fitur tersebut terdiri dari beberapa *pixel* dan akan dihitung selisih antara nilai *pixel* pada kotak terang dengan nilai *pixel* pada kotak gelap. Apabila nilai selisih antara daerah terang dengan daerah gelap di atas nilai ambang (*threshold*), maka daerah tersebut dinyatakan memiliki fitur. Proses pemilihan fitur dapat dilihat pada gambar 3.6.



Gambar 3.6 Pemilihan Fitur Wajah

Untuk memilih fitur mata, hidung, dan mulut maka digunakan kotak-kotak fitur yang bisa dilihat pada gambar 3.7.



Gambar 3.7 Pemilihan Fitur Mata, Hidung, Mulut

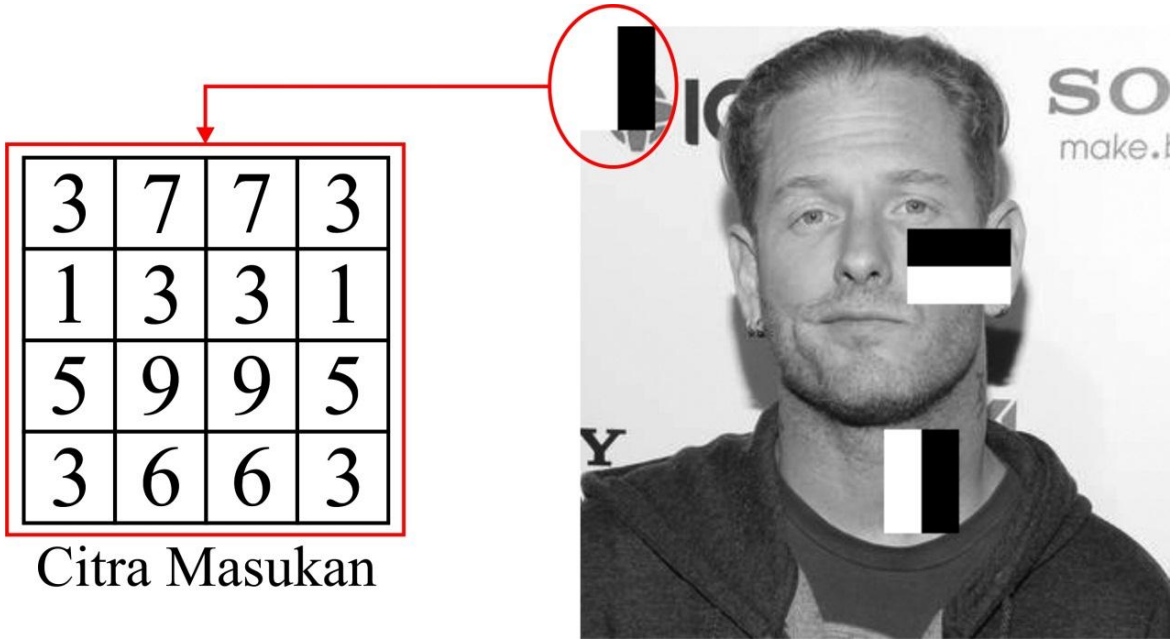
Pada umumnya citra wajah yang menghadap frontal ke depan dareah mata, tepi hidung, mulut dan dagu cenderung lebih gelap dibandingkan dengan daerah kedua pipi, dagu dan kening. Untuk

mempermudah dan mempercepat proses perhitungan nilai *Haar* pada sebuah *image*, algoritma *Viola-Jones* menggunakan sebuah perhitungan yang disebut dengan *Integral Image*.

b. *Integral Image*

Integral image sering digunakan pada algoritma untuk pendeteksian wajah. Dengan menggunakan *integral image* proses perhitungan bisa dilakukan hanya dengan satu kali scan dan memakan waktu yang cepat dan akurat. *Integral image* digunakan untuk menghitung hasil penjumlahan nilai *pixel* pada daerah yang dideteksi oleh fitur *haar*.

Nilai-nilai *pixel* yang akan dihitung merupakan nilai-nilai *pixel* dari sebuah citra masukan yang dilalui oleh fitur *haar* pada saat pencarian fitur wajah. Pada setiap jenis fitur yang digunakan, pada setiap kotak-kotaknya terdiri dari beberapa *pixel*. Apabila ada sebuah citra masukan yang dilalui oleh fitur *haar* dapat dilihat pada gambar 3.8.



Gambar 3.8 Nilai *Pixel-Pixel* Pada Sebuah Fitur

Dari nilai-nilai *pixel* yang didapatkan pada fitur tersebut, maka akan dihitung nilai integral image pada fitur tersebut dengan rumus 1.3.

$$s(x,y) = i(x,y) + s(x,y) + s(x,y-1) + s(x-1,y) - s(x-1,y-1) \quad (1.3)$$

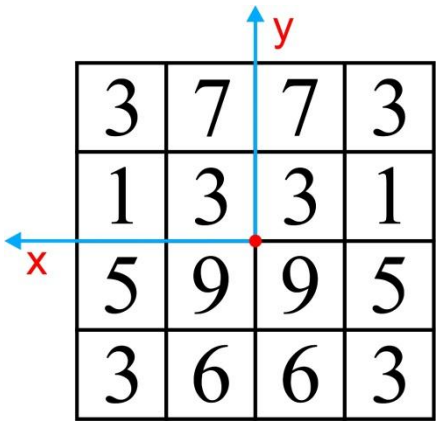
s(x,y) = merupakan nilai hasil penjulahan dari tiap-tiap pixel

i(x,y) = merupakan nilai intensitas diperoleh dari nilai pixel dari citra masukan

s(x-1,y) = merupakan nilai pixel pada sumbu x

s(x,y-1) = merupakan nilai pixel pada sumbu y

s(x-1,y-1) = merupakan nilai pixel diagonal



Gambar 3.9 Arah Perhitungan *Integral Image* [IKH08]

Pada pemrograman di Matlab, terebih dahulu harus disiapkan sebuah *matrix buffer* yang ukurannya sama dengan aslinya. Artinya jika pada kotak-kotak fitur tersebut terdapat *pixel* dengan *matrix* 4x4 maka disiapkan *matrix buffer* dengan ukuran yang sama dan diberi nilai nol. Tujuannya adalah untuk mempersiapkan memori dan juga untuk mempercepat komputasi. Pengaruhnya akan terasa pada saat pengolahan citra *image* dengan resolusi besar. Contoh dari perhitungan *integral image* dengan citra masukan pada gambar 3.9, akan dijelaskan pada tabel 3.4.

Tabel 3.3 Perhitungan *Integral Image*

Nilai pixel	Keterangan
i(x,y)=3 0 0 0	Nilai intensitas *pixel* (1,1) adalah 3 atau i(x,y) = 3. i(x,y) = 3 s(x-1,y) = 0 (di luar batas *matrix*) s(x,y-1) = 0 (di luar batas *matrix*) s(x-1,y-1) = 0 (di luar batas *matrix*) s(x,y) = i(x,y) + s(x,y-1) + s(x-1,y) – s(x-1,y-1), makas diapatkan nilai untuk *pixel* (1,1) adalah : s(x,y) = 3 + 0 + 0 - 0 = 3
s(x,y)=3 i(x,y)=7 0 0	I(x,y) = 7 s(x-1,y) = 3 s(x,y-1) = 0 (di luar batas *matrix*) s(x-1,y-1) = 0 (di luar batas *matrix*) s(x,y) = i(x,y) + s(x,y-1) + s(x-1,y) – s(x-1,y-1), makas diapatkan nilai untuk *pixel* (1,2) adalah : s(x,y) = 7 + 3 + 0 - 0 = 10

Nilai pixel	Keterangan
s(x,y)=3 s(x,y)=10 i(x,y)=1 0	I(x,y) = 1 s(x-1,y) = 0 (di luar batas *matrix*) s(x,y-1) = 3 s(x-1,y-1) = 0 (di luar batas *matrix*) s(x,y) = i(x,y) + s(x,y-1) + s(x-1,y) – s(x-1,y-1), makas diapatkan nilai untuk *pixel* (2,1) adalah : s(x,y) = 1 + 0 + 3 - 0 = 4
s(x,y)=3 s(x,y)=10 s(x,y)=4 i(x,y)=3	I(x,y) = 3 s(x-1,y) = 4 s(x,y-1) = 10 s(x-1,y-1) = 3 s(x,y) = i(x,y) + s(x,y-1) + s(x-1,y) – s(x-1,y-1), makas diapatkan nilai untuk *pixel* (2,2) adalah : s(x,y) = 3 + 10 + 4 – 3 =

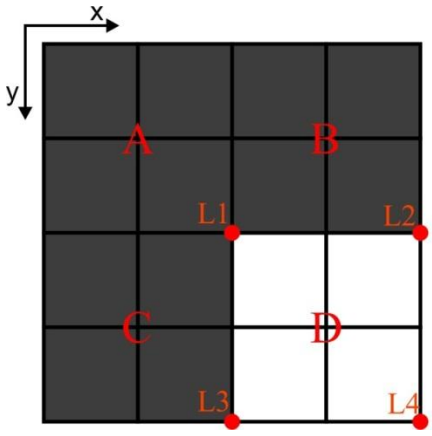
Apabila dilakukan perhitungan untuk semua *pixel* yang terdapat dalam kotak-kotak fitur, maka akan didapatkan hasil perhitungan dari *integral image* dapat dilihat pada gambar 3.10.

3	10	17	20
4	14	24	28
9	28	47	56
12	37	62	74

Citra Integral

Gambar 3.10 Hasil Perhitungan *Integral Image*

Setelah didapatkan hasil dari perhitungan *integral image*, selanjutnya akan dilakukan perhitungan untuk wilayah tertentu.



3	10	17	20
4	14	24	28
9	28	47	56
12	37	62	74

Gambar 3.11 Menghitung Pixel Pada Daerah Tertentu [IKH08]

Untuk menghitung jumlah *pixel* pada daerah D seperti pada gambar 3.11 di atas, maka digunakan rumus 1.4

$$D = L1 + L4 - (L2 + L3) \quad (1.4)$$

Contoh :

L1 = 14, L2 = 28, L3 = 37, L4 = 74 , maka jumlah *pixel* pada daerah D adalah :

D = 14 + 74 – (28 + 37)

= 23

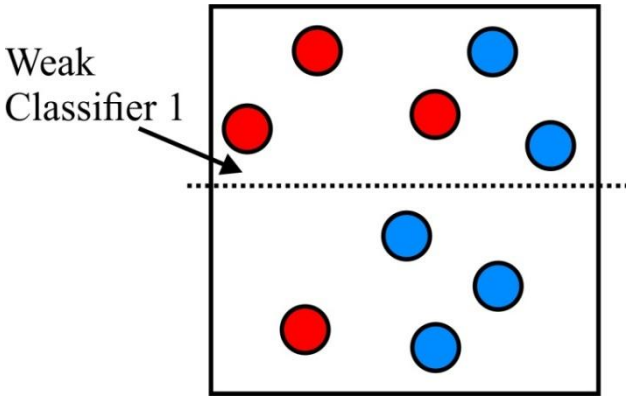
Apabila sudah didapatkan nilai *integral image* dari sebuah citra masukan dan nilai jumlah *pixel* pada daerah tertentu, maka hasil tersebut akan dibandingkan antara nilai *pixel* pada daerah terang dan daerah gelap. Jika selisih nilai *pixel* pada daerah terang dengan nilai *pixel* pada derah gelap di atas nilai ambang (*threshold*) maka daerah tersebut dinyatakan memiliki fitur.

2. Klasifikasi bertingkat

c. *Adaboost (Adaptive Boosting)*

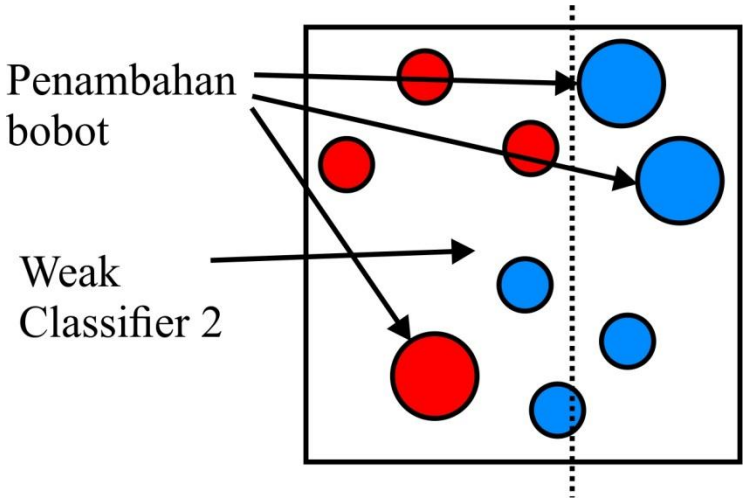
Adaptive boosting merupakan teknik yang digunakan untuk mengkombinasikan banyak *classifier* lemah untuk membentuk suatu gabungan *classifier* yang lebih baik. Proses dari *adaptive boosting* akan menghasilkan sebuah *classifier* yang kuat dari *classifier* dasar. Satuan dari *classifier* dasar tersebut disebut dengan *weak learner*. Setelah sebelumnya dilakukan pemilihan fitur *Haar*, pada proses selanjutnya dalam deteksi wajah *Viola-Jones*, dengan menggunakan algoritma *adaboost* fitur pada sebuah *image* akan dideteksi kembali. Tujuannya untuk mengetahui apakah ada fitur wajah pada daerah dengan klasifikasi fitur yang lemah. Pada *classifier* lemah akan dilakukan perhitungan dan dibandingan dengan *classifier* lainnya secara acak. Selanjutnya dilakukan kombinasi atau penggabungan pada *classifier* lemah untuk membentuk suatu kombinasi yang linier.

Pada gambar 3.12 di bawah ini menunjukkan beberapa classifier yang lemah pada sebuah fitur *image*. Lingkaran merah menunjukkan sebuah *classifier* yang lemah sedangkan lingkaran biru menunjukkan *classifier* kuat. Daerah dengan banyak fitur lemah diklasifikasikan sebagai daerah dengan klasifikasi yang lemah.



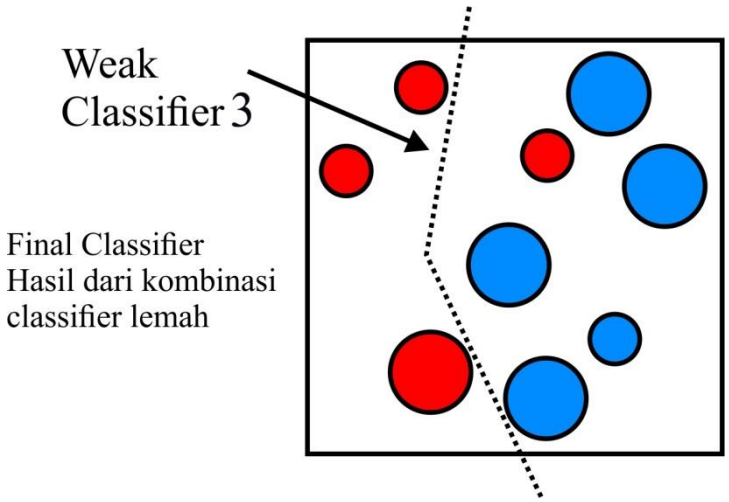
Gambar 3.12 *Classifier* Lemah [IKH08]

Pada gambar 3.12 didapatkan beberapa fitur dengan klasifikasi yang lemah maka bobot dari fitur tersebut akan di gabungkan untuk meningkatkan bobot dari fitur tersebut agar bisa menjadi fitur dengan *classifier* yang kuat. Hasil dari proses penggabungan *classifier* lemah dengan *classifier* kuat dapat dilihat pada gambar 3.13.



Gambar 3.13 Hasil Kombinasi Dari *Classifier* Lemah [IKH08]

Apabila masih terdapat *weak classifier* pada sebuah fitur setelah dilakukan kombinasi atau penggabungan pada sebuah daerah dengan klasifikasi yang lemah, maka daerah tersebut tetap dianggap sebagai *weak classifier* yang berarti tidak terdapat fitur wajah pada daerah tersebut. Hasil akhir dari penggabungan classifier pada algoritma *adaboost*, dapat dilihat pada gambar 3.14.



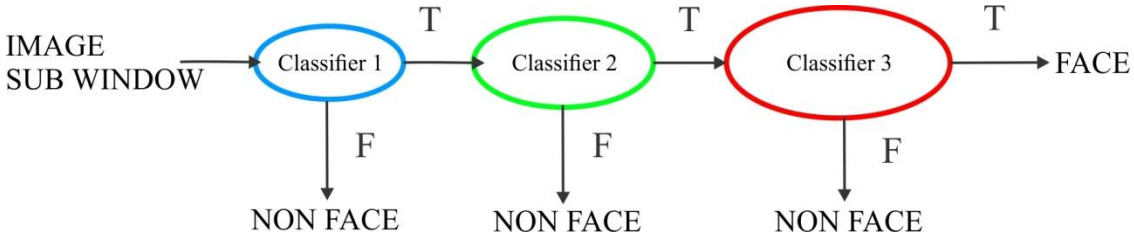
Gambar 3.14 Hasil Kombinasi Linier Dari *Classifier* Lemah [IKH08]

d. *Cascade classifier*

Cascade classifier melakukan proses dari banyak fitur fitur dengan mengorganisir dengan bentuk klasifikasi bertingkat. Terdapat tiga buah klasifikasi untuk menentukan apakah benar atau tidak ada fitur wajah pada fitur yang sudah dipilih.

Pada klasifikasi filter pertama, tiap subcitra akan diklasifikasi menggunakan satu fitur. Jika hasil nilai fitur dari filter tidak memenuhi kriteria yang diinginkan, hasil tersebut akan ditolak.

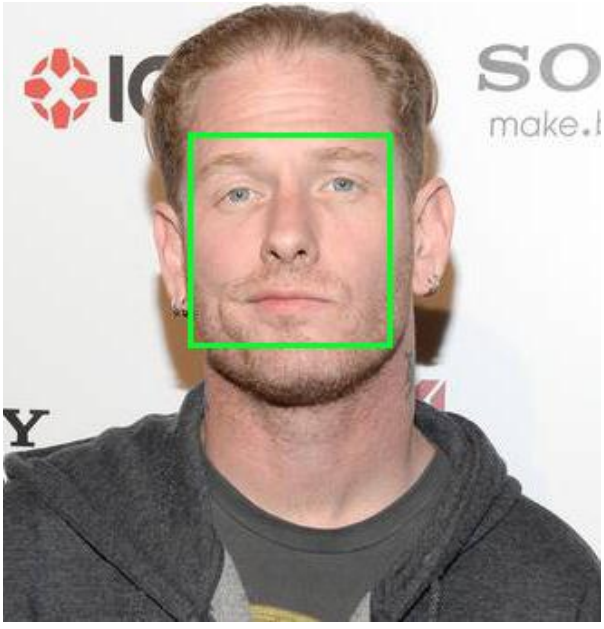
Algoritma kemudian bergerak ke *sub window* selanjutnya dan menghitung nilai fitur kembali. Jika didapat hasil sesuai dengan *threshold* yang diinginkan, maka dilanjutkan ke tahap filter selanjutnya. Hingga jumlah *sub window* yang lolos klasifikasi akan berkurang hingga mendekati *image* yang dideteksi. Pada gambar 3.15 di bawah ini merupakan proses rangkaian filter yang dilalui oleh setiap *classifier*.



Gambar 3.15 Cascade Classifier [ILH10]

1. Pada filter pertama dipilih satu fitur *classifier* dengan persentase tingkat pendeteksian sebesar 100% dan sekitar 50% tingkat kesalahan.
2. Pada filter kedua dipilih lima buah fitur *classifier* dengan persentase tingkat pendeteksian sebesar 100% dan 40% tingkat kesalahan (20% kumulatif).
3. Pada filter ketiga dipilih 20 fitur *classifier* dengan persentase tingkat pendeteksian sebesar 100% dengan tingkat kesalahan sebesar 10% (2%kumulatif).

Setelah dilakukan serangkain proses seperti pemilihan fitur dan klasifikasi bertingkat maka akan didapatkan sebuah hasil pendeteksian. Hasil pendeteksian bisa berupa wajah atau bukan wajah. Pada saat proses klasifikasi bertingkat dilakukan maka, pada *image* tersebut akan ditandai dengan sebuah *rectangle* pada daerah wajah yang terdeteksi dan apabila tidak ada wajah tedeteksi maka, *image* tersebut tidak akan ditandai oleh sebuah *rectangle*. Pada gambar 3.16 di bawah ini merupakan contoh hasil pendeteksian dari proses akhir deteksi wajah dengan algoritma *Viola-Jones*.



Gambar 3.16 Hasil Pendeteksian

3.3 Perancangan

Perancangan merupakan tahapan untuk memodelkan aplikasi ke dalam sebuah rancangan kasar atau *mock-up*, dari beberapa kebutuhan yang sudah didapatkan sebelumnya. Dalam tugas akhir kali ini perancangan tersebut meliputi perancangan arsitektur dan perancangan antarmuka. Hasil dari rancangan tersebut akan digunakan untuk menyelesaikan aplikasi deteksi wajah.

3.3.1 Perancangan Arsitektur

Perancangan aplikasi deteksi wajah ini disusun oleh beberapa bagian-bagian penting untuk membangun aplikasi menjadi sebuah aplikasi yang bisa digunakan dengan baik. Setiap bagian-bagian tersebut memiliki pernanan masing-masing sesuai dengan fungsi dan kegunaannya. Bagian-bagian yang menjadi penyusun dalam pembuatan aplikasi deteksi wajah ini adalah :

1. *Image processing.*

Perangkat lunak *image processing* yang digunakan pada pembuatan aplikasi deteksi wajah ini adalah *Matlab* R2013a *(Matrix Laboratory).* Adapun peranan *Matlab* dalam pembuatan aplikasi ini adalah sebagai berikut :

a. Membuat tampilan GUI (*Graphical user interface*).
b. Membuat perintah-perintah untuk setiap fungsi-fungsi yang akan digunakan.
c. Menghubungkan dengan *library* yang digunakan untuk proses pendeteksian.

2. *Library*

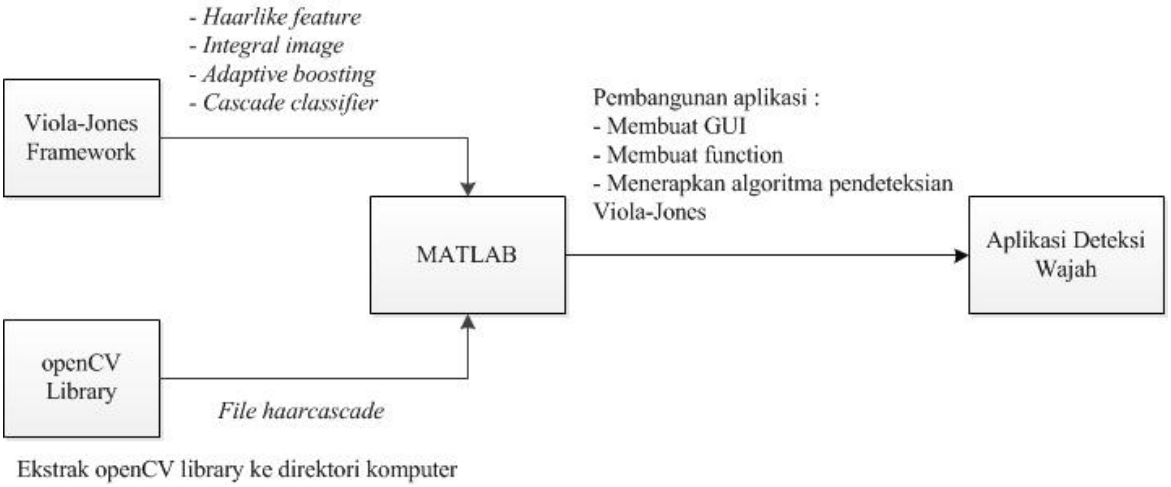
Library yang digunakan dalam pembuatan aplikasi deteksi wajah ini adalah *openCV library*. Pada library ini, algoritma pendeteksian yang akan digunakan dalam pembuatan aplikasi sudah dibangun ke dalam *library* tersebut.

3. Algoritma deteksi wajah.

Algoritma yang digunakan dalam pembuatan aplikasi deteksi wajah ini adalah algoritma *Viola-Jones.* Adapun peranan dari algoritma *Viola-Jones* dalam pembuatan aplikasi ini adalah sebagai berikut :

a. Melakukan pendeteksian wajah pada citra.
b. Menerapkan karakteristik dan metode pendeteksian pada aplikasi yang akan dibuat.

Ketiga bagian penyusun di atas merupakan bagian penting yang akan digunakan dalam pembuatan aplikasi deteksi wajah yang akan dibangun. Pada gambar 3.17 akan dijelaskan keterhubungan dari setiap bagian penyusun di atas dalam pembangunan aplikasi deteksi wajah.



Gambar 3.17 Diagram Perancangan Arsitektur

Gambar 3.17 di atas menunjukkan keterhubungan dari setiap bagian-bagian penyusun dalam pembangunan aplikasi deteksi wajah. Dalam pembangunan aplikasi dbutuhkan beberapa komponen penyusun seperti *framework* algoritma *Viola-Jones* dan *opencv library*. Pada *framework Viola-Jones* mencakup skema kerja dari algoritma seperti *haarlike feature*, *integral image*, *adaptive boosting* dan *cascade classifier*.

Selain *framework Viola-Jones*, *library opencv* juga digunakan pada pembangunan aplikasi. *Library opencv* digunakan untuk menghubungkan semua proses dari algoritma *Viola-Jones*, dalam melakukan proses pendeteksian wajah. *Library* tersebut terlebih dahulu harus di ekstrak ke direktori komputer supaya *file* yang dibutuhkan bisa digunakan untuk membangun aplikasi. Apabila proses ekstraksi sudah selesai, maka diperlukan sebuah fungsi untuk mengkonversi *file xml* pada *library* agar bisa digunakan pada *Matlab*. Fungsi untuk melakukan konversi *file xml* tersebut sudah tersedia pada *framework Viola-Jones*. semua komponen pendeteksian seperti *framework Viola-Jones* dan *library opencv* akan dimasukkan ke dalam *compiler* yang digunakan dalam pembanugnan aplikasi.

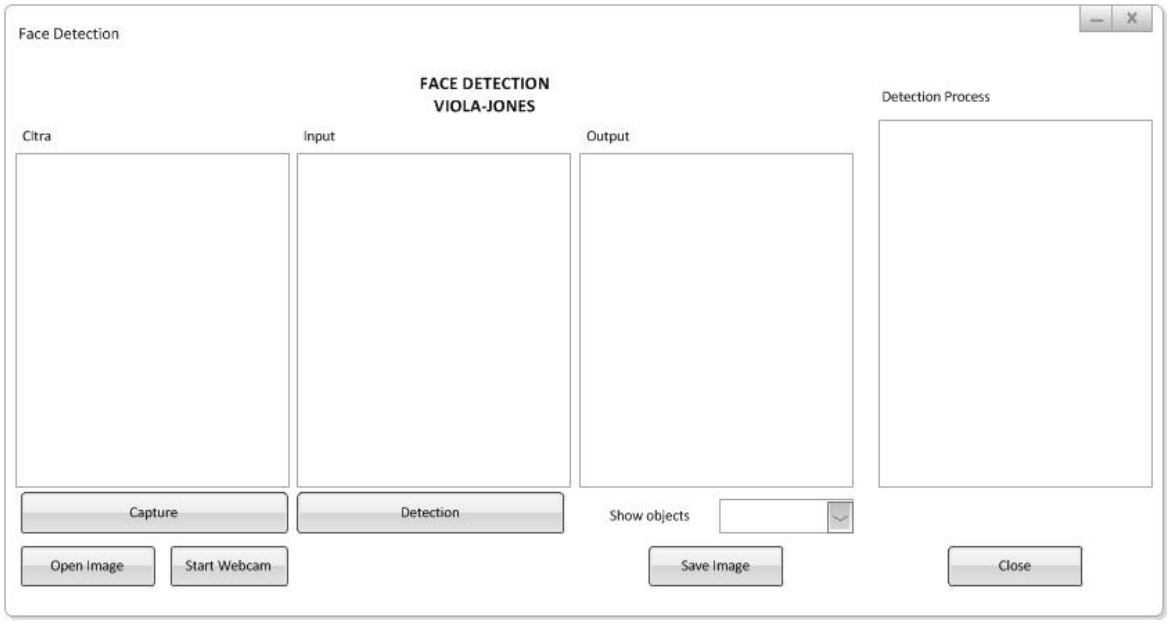
Untuk dapat membangun aplikasi deteksi wajah, maka proses selanjutnya adalah dengan melakukan implementasi pada *Matlab*. Program *Matlab* digunakan untuk pembuatan GUI *(Graphical user interface),* pembuatan perintah-perintah dan fungsi-fungsi, dan pemanggilan fungsi pendeteksian pada *openCV library.*

3.3.2 Perancangan Antarmuka

Perancangan antarmuka merupakan gambar rancangan dari aplikasi deteksi wajah yang akan dibuat. Rancangan tersebut meliputi tampilan awal ketika aplikasi dijalankan, tampilan ketika aplikasi lakukan proses pendeteksian, dan tampilan dialog interaksi dengan *user*.

Secara keseluruhan tampilan dari aplikasi deteksi wajah ini dibuat menjadi satu halaman yang terdiri dari empat buah panel yaitu panel citra, panel *input*, panel *output* dan panel proses pendeteksian. Berikut ini merupakan tampilan aplikasi deteksi wajah yang akan dibuat.

1. Tampilan aplikasi.



Gambar 3.18 Tampilan Aplikasi Deteksi Wajah

Gambar 3.18 di atas merupakan rancangan dari tampilan halaman aplikasi deteksi wajah ketika aplikasi dijalankan oleh *user*. Pada halaman aplikasi tersebut terdapat enam buah *button* yaitu *button capture, button open image, button start/stop webcam, button detection, button save image, dan button close*. Terdapat juga empat buah panel dengan fungsi-fungsi yang berbeda yaitu panel citra, panel *input*, panel *ouput* , dan panel *detection process*. Selain itu juga terdapat sebuah *list box* pada aplikasi yang akan dirancang. Berikut ini merupakan penjelasan fungsi dari setiap komponen-komponen yang ada pada aplikasi :

1. *Button capture* berfungsi untuk menangkap citra dari kamera yang sedang aktif.
2. *Button open image* berfungsi untuk mengambil foto masukan dari direktori komputer.
3. *Button start/stop webcam* berfungsi untuk menjalankan atau menghentikan kamera.
4. *Button detection* berfungsi untuk melakukan proses pendeteksian dari citra wajah yang akan dideteksi.
5. *Button save imgae* berfungsi untuk menyimpan area wajah yang terdeteksi oleh aplikasi.
6. *Button close* berfungsi untuk menutup aplikasi.
7. Panel citra berfungsi untuk menampilkan citra dari kamera.
8. Panel *input* berfungsi untuk menampilkan citra hasil capture dari kamera atau citra masukan dari komputer.

9. Panel *output* berfungsi untuk menampilkan citra hasil dari proses pendeteksian wajah.
10. Panel *detection process* berfungsi untuk menampilkan proses pendeteksian algoritma *Viola-Jones* dari citra yang dideteksi.
11. *List box* berfungsi untuk menampilkan hasil pendeteksian wajah berdasarkan urutan dari proses pendeteksian algoritma *Viola-Jones.*

3.3.3 Perancangan Algoritma

Untuk melakukan proses pendeteksian wajah pada sebuah citra, maka dibutuhkan sebuah algoritma yang akan digunakan sebagai perintah yang memuat berbagai fungsi-fungsi khusus. Dalam pembangunan aplikasi deteksi wajah ini fungsi untuk pendeteksian wajah sudah disediakan pada *library* yang digunakan yaitu *opencv* 2.4.10. Pada *library* tersebut sudah terdapat serangkaian proses pendeteksian wajah sesuai dengan algoritma *Viola-Jones*. Untuk mendapatkan *library opencv* bisa diunduh langsung pada situs resminya di http://www.opencv.org/.

Selanjutnya agar *library opencv* dapat digunakan dalam melakukan implementasi pembangunan aplikasi, maka *library* tersebut harus diekstrak terlebih dahulu pada direktori C:/ di komputer. Setelah *library* tersebut diekstrak, maka selanjutnya pada aplikasi MATLAB harus dilakukan pengaturan terlebih dahulu agar *library opencv* bisa digunakan. Cara pengaturannya yaitu pada aplikasi MATLAB harus dituliskan perintah `mex -setup` untuk mengkoneksikan *library* dengan aplikasi MATLAB.

Pada *library opencv* terdapat *file xml* yang bisa digunakan pada saat impelementasi pembangunan aplikasi pendeteksian wajah. *File* tersebut bisa dipanggil pada perintah pendeteksian dengan menuliskan *filename* pada baris program aplikasi. Beberapa *file xml* yang akan digunakan pada pembangunan aplikasi deteksi wajah *Viola-Jones* adalah sebagai berikut :

1. *haarcascade_frontalface_default.xml*
2. *haarcascade_frontalface_alt.xml*

Dari ke-empat *file xml* tersebut merupakan *haar cascade classifier* yang bisa digunakan untuk mendeteksi wajah frontal. Salah satu dari *file* tersebut dapat digunakan pada proses pendeteksian wajah. Masing-masing dari tipe *haar cascade classifier* tersebut akan memberikan hasil yang sedikit berbeda sesuai dengan kondisi lingkungan pada sebuat citra.

Selain menggunakan *file* yang sudah disediakan oleh *library* untuk melakukan proses pendeteksian, berbagai fungsi-fungsi lain juga bisa dibuat dalam pembangunan aplikasi deteksi wajah *Viola-Jones*. Untuk pembuatan fungsi bisa dilakukan dengan membuat menuliskan kode program pada *file* berekstensi *.m. Selanjutnya untuk pembuatan desain antarmuka aplikasi bisa dilakukan dengan menuliskan perintah `GUIDE` pada editor di aplikasi MATLAB. Hasil dari pembangunan desain antarmuka tersebut akan menghasilkan sebuah *file* berekstensi *.fig.