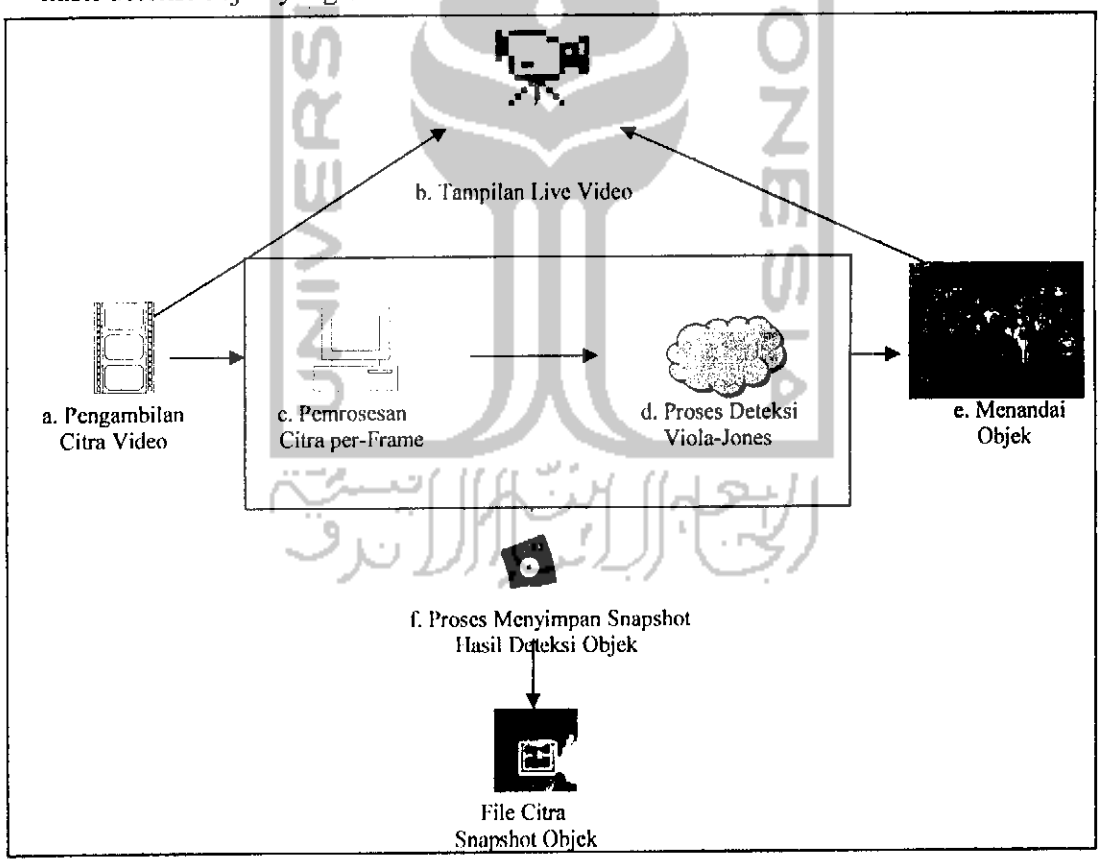
BAB IV
MODEL SISTEM

4.1. Model Sistem

Aplikasi yang dibangun adalah aplikasi yang dapat mengolah citra yang diambil dari kamera video, dan menganalisisnya untuk mengetahui keberadaan suatu objek. Ada beberapa proses yang terjadi pada jalannya aplikasi ini, yaitu pengambilan citra video, pemrosesan citra pada setiap *frame*, , pendeteksian objek menggunakan teknik Viola-Jones, dan kemudian menggambar persegi panjang pada objek wajah yang ditemukan dari citra video tersebut serta menyimpan *snapshot* hasil deteksi objek yang ditemukan.



Gambar 4.1 Diagram Proses

Diagram yang tertera pada gambar 4.1 menunjukkan gambaran proses yang terjadi pada aplikasi ini secara umum.

a. *Pengambilan Citra Video*

Proses pengambilan citra adalah proses yang pertama dilakukan pada aplikasi ini dan proses ini dilakukan terus menerus sampai aplikasi dihentikan. Proses ini menggunakan suatu pustaka dari framework C++ OpenFrameworks yang didalamnya sudah tersedia suatu *interface* untuk mengambil data dari *webcam* dan melakukan pengambilan citra video yang bersifat *streaming*. Pada proses ini dilakukan *streaming* video dari *webcam* dengan ukuran 320x240 *pixel* dan kemudian mengirimkan datanya ke proses lain untuk dianalisa.

b. *Tampilan Live Video*

Proses ini adalah proses yang dapat terlihat oleh pengguna sistem. Proses ini menerima data streaming video hasil dari proses **a** diatas dan menampilkannya ke layar. Komponen yang menangani proses ini adalah library *streaming video* dan *image processing* dari OpenFrameworks. Proses ini juga menerima hasil penandaan objek yang telah diketemukan berupa persegi panjang di sekitar objek.

c. *Pemrosesan Citra Per-Frame*

Selama proses pengambilan citra berjalan, *input* citra *streaming* dari *webcam* diambil setiap frame menggunakan library openFrameworks, kemudian dari *input* tersebut frame diproses menjadi suatu variabel OpenCV dan diubah menjadi citra *greyscale* melalui fungsi yang ada pada OpenCV.

Dari citra yang sudah diubah menjadi *greyscale* ini, akan ada fungsi yang bertugas memproses dan menganalisa citra tersebut. Kemudian pada citra *greyscale* ini dilakukan *histogram equalization* untuk menormalkan *brightness* dan menaikkan tingkat kontras pada citra tersebut, sehingga citra yang ada lebih mudah untuk dianalisa kedepannya. Setelah selesai diproses, citra ini diproses lagi oleh fungsi deteksi dengan menggunakan teknik Viola-Jones. Intinya, pemrosesan *real-time* yang terjadi sebenarnya dilakukan per-frame.

d. Proses Deteksi Viola-Jones

Disini dilakukan deteksi pada citra per-frame yang telah diproses sebelumnya dengan berdasar teknik yang telah dikembangkan oleh Viola-Jones dan menggunakan fitur Haar. Untuk membantu beberapa pemrosesan citra disini menggunakan framework *computer vision* dari Intel, yaitu OpenCV. Untuk mengenali objek, fungsi ini juga menggunakan suatu file xml "haarcascade_frontalface_alt.xml" yang berisi hasil pelatihan pembelajaran mesin berupa *cascade* atau *decision tree* yang dapat digunakan untuk mengenali objek.

Pertama-tama fungsi ini mengkonversi input citra *greyscale* yang semula bertipe IplImage yang merupakan suatu tipe variabel OpenCV untuk mendefinisikan suatu citra, menjadi suatu variabel *multi-channel* matrix agar nanti mudah diproses. Sebelum melakukan ini, fungsi menciptakan suatu *hidden cascade classifier* dari input file pelatihan xml yang telah disebutkan sebelumnya, dan bekerja bersama-sama dengan *cascade classifier* sebenarnya. Proses ini kemudian memanggil fungsi lain yang bertugas membuat suatu citra *integral* yang berupa matrix dari input citra, dan dimasukkan ke variabel lain. Dengan penghitungan citra *integral*, bisa dilakukan penghitungan cepat dari suatu *subregion*, membuat suatu ringkasan dari *matrix* citra, atau melakukan penghitungan standar deviasi (Bradski dan Kaehler, 2008). Citra *integral* ini merupakan salah satu tahap dalam proses deteksi Viola-Jones.

Pada proses pendeteksian objek ini juga menggunakan fungsi Canny Pruning dari OpenCV yang bertujuan untuk menemukan garis tepi yang menonjol dari objek dan memudahkan pada pendeteksian (Bradski dan Kaehler, 2008). Setelah dilakukan fungsi *canny pruning*, citra diubah kembali menjadi citra *integral*.

Setelah melakukan inisialisasi variabel-variabel dasar yang diperlukan dan memanggil set *cascade* pengklasifikasi dari file xml dan membuat set *hidden cascade*, akan disiapkan *cascade* untuk mendeteksi objek pada ukuran tertentu dan bagian tertentu dari citra, dan hal ini dilakukan pada suatu iterasi. Disini juga dilakukan penghitungan poin-poin pojok pada *hidden cascade* yang merupakan koordinat window objek original yang telah diskalakan dan hasil penghitungan dari poin-poin *cascade* asli, dan melakukan pengaturan pada citra *input* untuk diproses

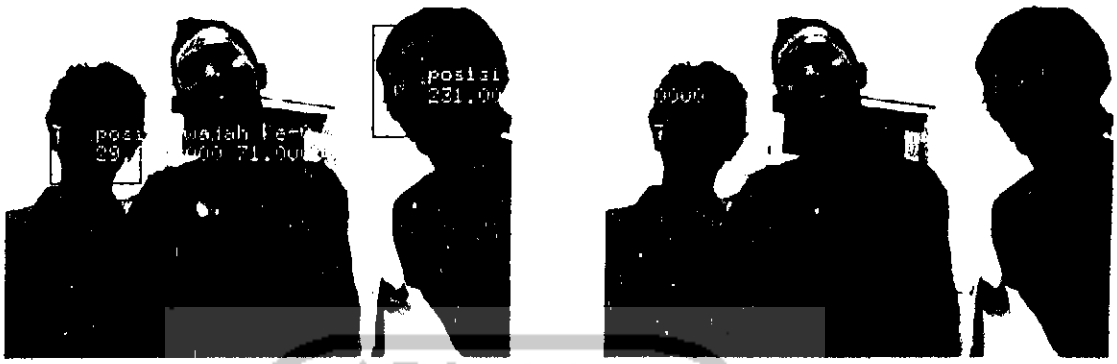
lebih lanjut. Kemudian dilakukan penghitungan pada poin-poin pojok yang digunakan untuk fitur persegi panjang Haar pada *node cascade*.

Kemudian ketika citra yang ada telah disiapkan, fungsi akan melakukan penghitungan antara citra *input* yang dianalisa, dengan fitur persegi panjang Haar yang telah disebutkan sebelumnya dalam suatu iterasi. Jika nilai balikan yang didapat dari penghitungan adalah positif, fungsi akan melanjutkan klasifikasi menuju klasifikasi yang lebih kompleks, sehingga akurasi keberadaan objek cukup mencapai *threshold*. Namun jika memberikan nilai negatif, maka fungsi akan mengabaikan dan maju ke poin koordinat pencarian berikutnya dari citra *input* kemudian melakukan penghitungan lagi, begitu seterusnya sampai mencapai poin koordinat akhir.

Setelah beberapa region yang lolos klasifikasi ditandai koordinat posisi dan ukurannya, region persegi panjang yang saling berdekatan akan disatukan menjadi satu persegi panjang dan dianggap sebagai satu kesatuan objek, kemudian nilai-nilainya akan dimasukkan ke suatu variabel nilai balikan fungsi.

e. Proses Menandai Objek Pada Tampilan

Fungsi dari OpenFrameWorks akan menerima nilai balikan dari pemanggilan fungsi pendeteksian objek yang berisi nilai-nilai tentang objek yang telah berhasil dideteksi. Dari nilai balikan tersebut, didapat koordinat pojok dari region objek, panjang dan lebarnya, dan juga titik tengahnya. Kemudian fungsi pendeteksi akan mengembalikan nilai-nilai tersebut ke aplikasi utama, dan fungsi dari OpenFrameworks akan menggambar persegi panjang pada frame sesuai posisi dan ukuran objek yang telah ditemukan, dan mengirimnya ke tampilan *live-video* pada proses **a**.



Gambar 4.2 Objek yang telah terdeteksi (Rowley et al,1998)

f. Proses menyimpan snapshot hasil deteksi objek

Selain menandai objek pada *live-video*, aplikasi juga akan memanggil fungsi lain yang bertugas melakukan pengambilan snapshot pada objek yang terdeteksi. Properti dari snapshot ini merupakan hasil nilai balikan dari fungsi pendeteksi objek, yang berisi koordinat, panjang, dan lebar dari region persegi panjang objek yang terdeteksi. Kemudian dari hasil snapshot ini akan disimpan sebagai file citra, dan disimpan ke dalam folder tertentu.

Gambar 4.3 Snapshot Hasil Deteksi