

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Pada bab ini berisi tentang implementasi dan pengujian meliputi penulisan kode program, pembuatan tampilan antarmuka, penerapan algoritma yang digunakan dan juga pengujian terhadap aplikasi yang dibangun.

4.1 Implementasi

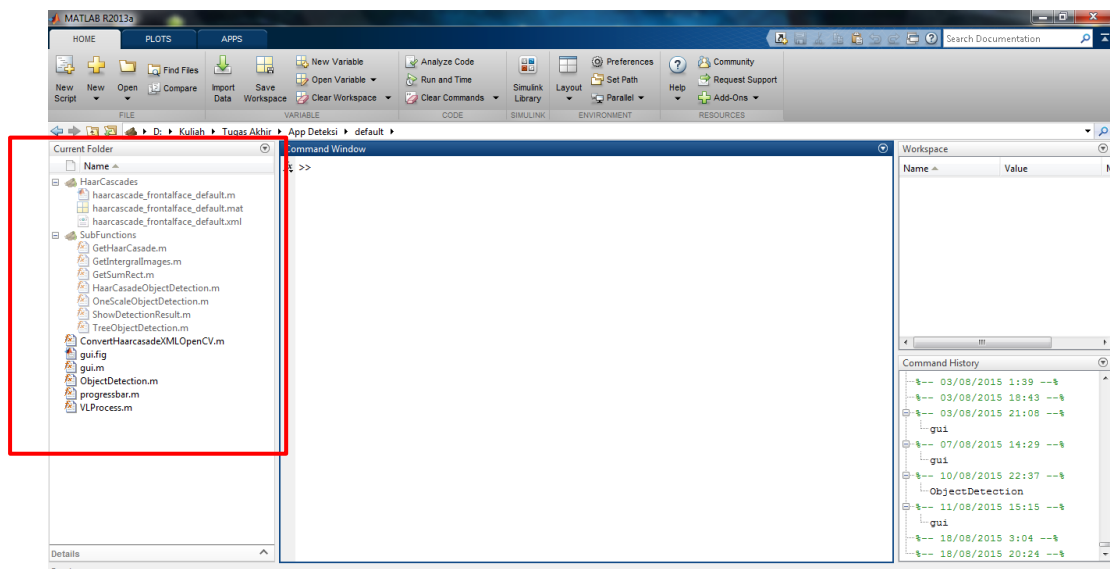
Setelah rancangan dari aplikasi yang akan dibangun sudah didapatkan, maka pada tahapan selanjutnya akan dilakukan implementasi terhadap hasil rancangan yang sudah ada. Tahapan implementasi tersebut adalah penerapan algoritma *Viola-Jones* ke dalam aplikasi, pembuatan tampilan antarmuka dan pembuatan fungsi-fungsi aplikasi dengan menggunakan MATLAB.

4.1.1 Penerapan algoritma *Viola-Jones*.

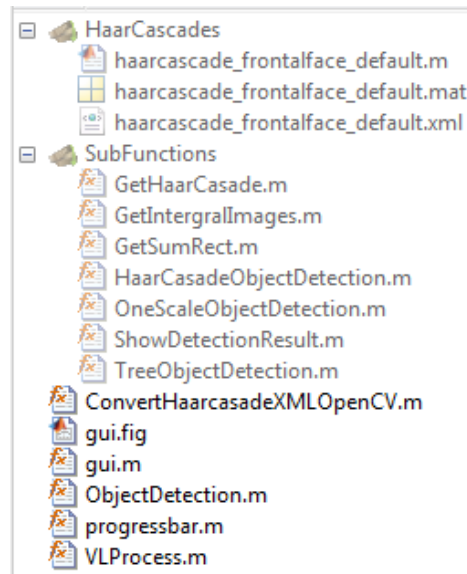
Algoritma *Viola-Jones* merupakan komponen penting dalam pembangunan aplikasi deteksi wajah yang sedang dilakukan. Untuk itu diperlukan sebuah keterhubungan antara fungsi-fungsi yang dibangun dengan algoritma *Viola-Jones*. Algoritma tersebut bisa didapatkan pada *library opencv* secara mudah dan gratis yang bisa diunduh pada situs resmi *opencv* yaitu www.opencv.org.

Apabila *library* tersebut sudah didapatkan, maka algoritma *Viola-Jones* yang diperlukan bisa digunakan dengan cara mengekstrak *library* tersebut ke direktori komputer. Setelah dilakukan ekstraksi *file* dari *library* tersebut, maka beberapa *file* yang diperlukan untuk membangun aplikasi deteksi wajah *Viola-Jones*, bisa diambil dan diterapkan ke dalam aplikasi yang akan dibangun.

Penerapan algoritma *Viola-Jones* dilakukan dengan cara mengambil beberapa fungsi dari *library opencv* yang dibutuhkan dan menyimpannya ke dalam aplikasi. Semua fungsi tersebut akan dihubungkan dengan cara menuliskan *filename* setiap fungsi ke dalam kode program aplikasi yang dibangun. Beberapa file yang digunakan dapat dilihat pada gambar 4.1 dan gambar 4.2.



Gambar 4.1 File *opencv* dan algoritma *Viola-Jones* pada MATLAB



Gambar 4.2 *File opencv dan algoritma Viola-Jones*

Dalam pembangunan aplikasi deteksi wajah *Viola-Jones*, *file-file* tersebut disimpan kedalam dua buah folder yang berbeda. Untuk *file* dengan nomor satu sampai dengan empat disimpan kedalam folder *haarcascade* yang menunjukkan bahwa folder tersebut berisikan *file-file haarcascade* yang berbeda. Semua *file haarcascade* tersebut bisa didapatkan dengan cara mengunduh di www.github.com. Untuk *file* dengan nomor urut lima sampai dengan sebelas disimpan ke dalam folder *SubFunctions*. Dalam pendeteksian wajah dengan menggunakan algoritma *Viola-Jones*, ada beberapa karakteristik dari pendeteksian dengan wajah tegak kedepan atau *frontal face*. Pada pembangunan aplikasi kali ini digunakan tipe *haarcascade_frontalface_default* dan *haarcascade_frontalface_alt*.

Setiap *file haarcascade_frontalface* tersebut merupakan *file* jenis *xml*, dan agar bisa dibaca oleh pemrograman Matlab maka diperlukan sebuah fungsi untuk melakukan *convert file xml* tersebut. Pada pembangunan aplikasi deteksi wajah ini *file convert* tersebut bisa didapatkan dari *library opencv* dan bisa langsung digunakan ke dalam aplikasi dengan cara menyimpan *file* tersebut ke dalam aplikasi yang sedang dibangun. Pada pembangunan kali ini *file* tersebut disimpan diluar folder *Haarcascade*.

Selain folder yang berisikan *file haarcascade*, dibuatkan juga folder *SubFunctions* yang di dalamnya menyimpan fungsi-fungsi *opencv* yang bisa digunakan untuk pendeteksian wajah dengan algoritma *Viola-Jones*. Masing-masing dari *file* tersebut saling berhubungan dalam proses pendeteksian wajah untuk setiap fungsi-fungsinya.

4.1.2 Pembuatan *Function*

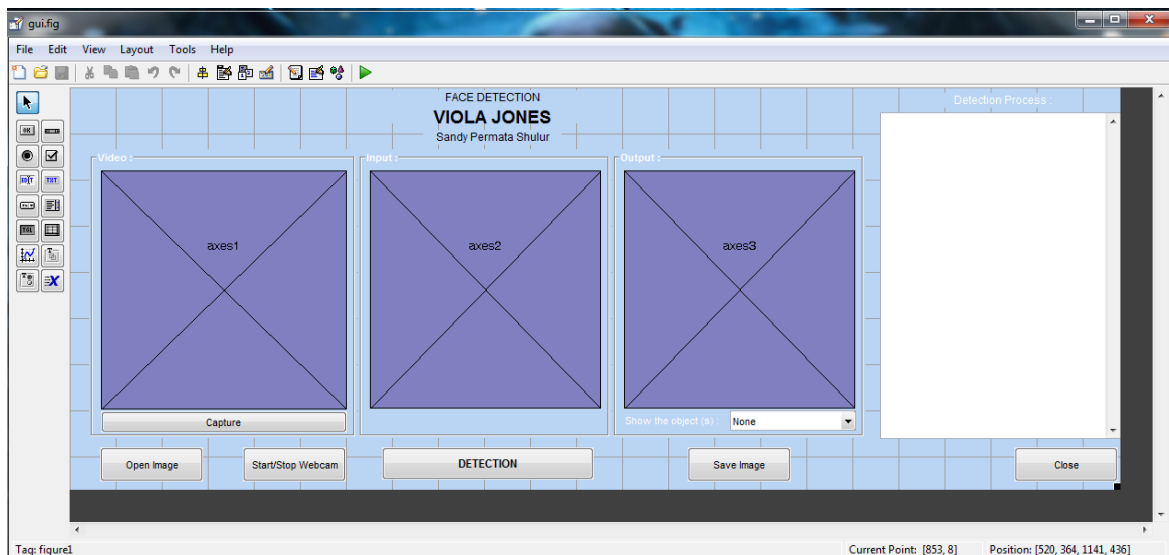
Dalam pembangunan aplikasi deteksi wajah *Viola-Jones* ini, selain fungsi-fungsi dari *library opencv* juga dibutuhkan fungsi-fungsi lain yang dibuat agar fungsional aplikasi bisa berjalan dengan baik. Fungsi-fungsi dibuat pada beberapa komponen dalam tampilan antarmuka aplikasi. Selain itu juga dibuat beberapa fungsi lain di luar tampilan antarmuka, seperti fungsi

untuk menampilkan *progress bar* dan fungsi untuk menghubungkan *converter haarcascade* dengan *file haarcascade* pada folder *HaarCascade*.

4.1.3 Tampilan Antarmuka

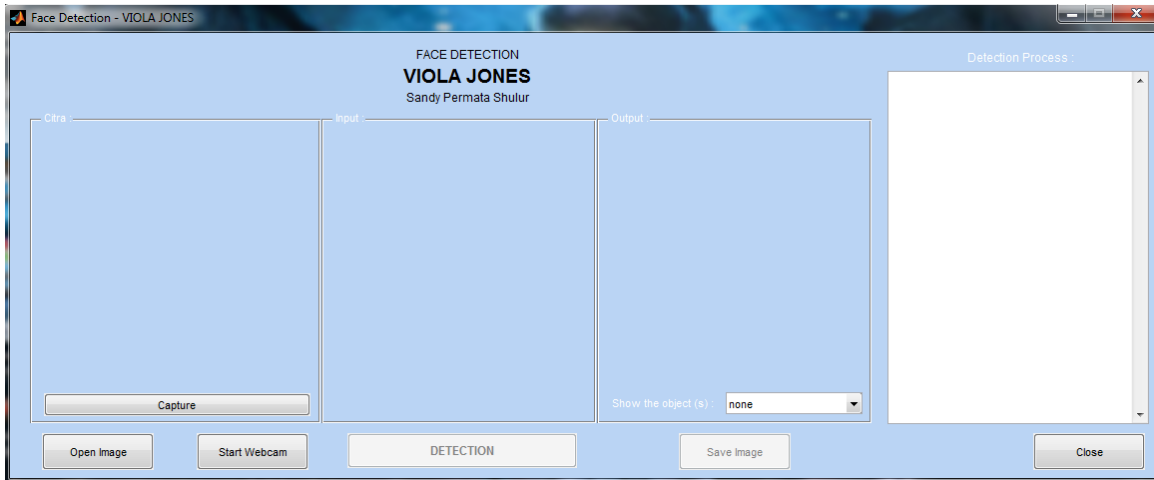
Sesuai dengan rancangan tampilan antarmuka dari aplikasi yang akan dibangun, maka pada tahapan implementasi ini tampilan antarmuka tersebut akan dibuat sesuai dengan hasil rancangan tersebut. Untuk membangun tampilan antarmuka, pada program matlab bisa dilakukan dengan cara menuliskan perintah *guide* di *command window*. Komponen antarmuka bisa dibuat dengan cara melakukan *drag and drop* komponen yang akan digunakan pada *figure* antarmuka dari aplikasi.

Tampilan antarmuka disimpan pada sebuah *file* dengan *filename* *gui.fig* dan untuk penulisan fungsi-fungsi dari setiap komponen tampilan antarmuka juga disimpan kedalam sebuah *file* dengan *filename* *gui.m*. Proses pembangunan tampilan antarmuka aplikasi deteksi wajah *Viola-Jones* dapat dilihat pada gambar 4.3.



Gambar 4.3 Proses Pembangunan Tampilan Antarmuka

Pada tampilan antarmuka yang dibuat terdapat enam buah *button* yang masing-masing mempunyai fungsi yang berbeda-beda. Untuk menuliskan fungsi pada setiap *button* bisa dilakukan dengan cara melakukan klik kanan pada *button* dan memilih *view callbacks*. Setelah itu penulisan fungsi bisa dilakukan jika sudah memilih *callback* pada menu yang tampil. Selain *button* juga terdapat tiga buah *axes* yang masing-masing akan menampilkan citra dari kamera maupun citra masukan dari komputer. Untuk menampilkan proses dari pendeteksian *Viola-Jones* maka pada tampilan antarmuka tersebut juga digunakan komponen *listbox*. Untuk menampilkan judul dari aplikasi digunakan tiga buah *label* yang disimpan dibagian atas aplikasi. Hasil dari tampilan antarmuka aplikasi deteksi wajah *Viola-Jones* dapat dilihat pada gambar 4.4




Gambar 4.4 Hasil Pembangunan Tampilan Antarmuka

4.2 Pengujian




Pada tahapan pengujian ini dilakukan untuk dua buah tipe *haarcascade* yang ada pada algoritma *Viola-Jones*. Tipe *haarcascade* yang akan digunakan dalam tahapan pengujian ini yaitu *haarcascade_frontalface_default* dan *haarcascade_frontalface_alt*. Pengujian ini dilakukan untuk melihat hasil dari pendeteksian pada setiap tipe *haarcascade* tersebut. Selanjutnya akan dibandingkan hasil dari setiap tipe pendeteksian tersebut meliputi waktu, dan akurasi pendeteksian wajah.

Pengujian dilakukan dengan menggunakan lima buah sampel citra foto. Masing-masing dari lima foto tersebut berasal dari *capture* dari *webcamera* atau foto masukan dari komputer. Selain itu properti jenis dari setiap sampel foto yang digunakan juga berbeda-beda. Sampel pengujian dapat dilihat pada tabel 4.1.


Tabel 4.1 Sampel Pengujian

No	Sampel	Nama File	Dimensi	Sumber
1.		1.jpg	911x816 pixel	Direktori komputer

Tabel 4.1 Sampel Pengujian

No	Sampel	Nama File	Dimensi	Sumber
2.		2.jpg	768x1024 <i>pixel</i>	Direktori komputer
3.		3.jpg	1435x1298 <i>pixel</i>	Direktori komputer
4.		4.jpg	550x412 <i>pixel</i>	Direktori komputer

Tabel 4.1 Sampel Pengujian

No	Sampel	Nama File	Dimensi	Sumber
5.		5.jpg	580x600 pixel	Web camera

Dari masing-masing sampel yang sudah didapatkan, selanjutnya akan dilakukan pengujian terhadap semua sampel tersebut. Pengujian dilakukan untuk mendapatkan hasil pendeteksian wajah dari aplikasi yang sudah dibangun dengan tipe *haarcascade* yang berbeda. Pemilihan kelima sampel tersebut karena setiap foto dari sampel memiliki karakteristik yang berbeda-beda. Adapun penjelasan dari pemilihan sampel-sampel tersebut adalah sebagai berikut :



1. Sampel 1.jpg dipilih karena pada foto tersebut terdapat dua buah wajah. Akan tetapi dari kedua wajah tersebut terdapat satu buah wajah yang bukan merupakan wajah manusia. Selain itu pencahayaan dari foto tersebut cukup terang dan memungkinkan aplikasi bisa mendeteksi wajah yang sedang menghadap frontal ke depan.
2. Sampel 2.jpg dipilih karena pada foto tersebut terdapat satu buah objek wajah yang menghadap kedepan dengan pencahayaan yang terang. Ukuran dari sampel yang digunakan juga cukup besar digunakan untuk mengetahui kecepatan proses pendeteksian pada foto tersebut.
3. Sampel 3.jpg dipilih karena pada foto tersebut terdapat tiga buah wajah. Masing-masing dari wajah tersebut menghadap kedepan. Akan tetapi ada dua buah wajah dengan posisi agak miring dan salah satunya menggunakan kacamata. Warna foto sampel yang digunakan dibuat hitam putih untuk mengetahui perbedaan proses pendeteksian dengan foto RGB. Ukuran dari foto yang digunakan cukup besar dengan pencahayaan yang cukup terang. Berbagai kriteria dari foto tersebut sengaja dipilih untuk mengetahui bagaimana proses pendeteksian wajah pada sampel tersebut.
4. Sampel 4.jpg dipilih karena foto tersebut bukan merupakan foto manusia. Akan tetapi pada foto tersebut terdapat tiga buah wajah binatang yang secara komponen hampir sama dengan wajah manusia. Dari kriteria tersebut akan dilakukan pengujian apakah aplikasi akan mendeteksi objek tersebut sebagai wajah atau bukan wajah.

5. Sampel 5.jpg merupakan foto yang berasal dari citra *web camera*. Foto tersebut merupakan hasil *capture* dari aplikasi yang sudah dibangun. Foto diambil dengan posisi wajah menghadap kedepan dengan pencahayaan yang agak sedikit gelap. Kualitas dari hasil *capture* fotopun bergantung pada spesifikasi dari *web camera*. Sampel ini digunakan untuk mengetahui perbedaan proses dan hasil pendeteksian wajah pada aplikasi.

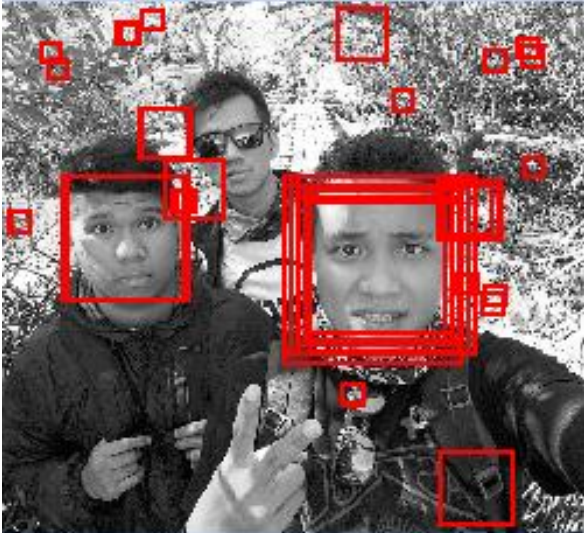

5.1.1 Pengujian menggunakan *haarcascade_frontalface_default.xml*

Tabel 4.2 di bawah ini merupakan hasil dari pengujian pendeteksian wajah dengan menggunakan tipe *haarcascade_frontalface_default.xml*.


Tabel 4.2 Hasil pengujian menggunakan *haarcascade_frontalface_default.xml*.

No.	Hasil Pendeteksian			Keterangan
1.				Nama File : 1.jpg Dimensi : 911x816 <i>pixel</i>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	
	true	tidak ada	3	
2.				Nama File : 2.jpg Dimensi : 768x1024 <i>pixel</i>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	
	true	ada	33	
	Waktu Pendeteksian	1 menit 35 detik		
	Waktu Pendeteksian	1 menit 42 detik		

Tabel 4.2 Hasil pengujian menggunakan *haarcascade_frontalface_default.xml*.

No.	Hasil Pendeteksian			Keterangan
3.				<p>Nama File : 3.jpg Dimensi : 1435x1298 pixel</p>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	true	ada	43	3 menit 35 detik
No.	Hasil Pendeteksian			Keterangan
4.				<p>Nama File : 4.jpg Dimensi : 550x412 pixel</p>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	false	tidak ada	0	38 detik

Tabel 4.2 Hasil pengujian menggunakan *haarcascade_frontalface_default.xml*.

No.	Hasil Pendeteksian			Keterangan
5.				Nama File : 5.jpg Dimensi : 580x600 <i>pixel</i>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	true	tidak ada	1	53 detik



Dari lima buah sampel yang sudah diuji dengan menggunakan *haarcascade_frontalface_default.xml* bisa disimpulkan bahwa secara umum didapatkan hasil yang cukup baik. Pendeteksian pada citra foto yang di dalamnya terdapat wajah, secara keseluruhan terdeteksi namun masih ada beberapa objek wajah yang tidak terdeteksi ataupun objek bukan wajah yang terdeteksi sebagai wajah. Selain itu tidak terdeteksi wajah pada citra yang menampilkan foto binatang.

Untuk waktu pendeteksian pada foto wajah memerlukan waktu yang cukup lama diatas 60 detik. Hal tersebut mungkin terjadi karena algoritma *Viola-Jones* mendeteksi citra per-*pixel* sesuai dengan ukuran citra yang dideteksi.

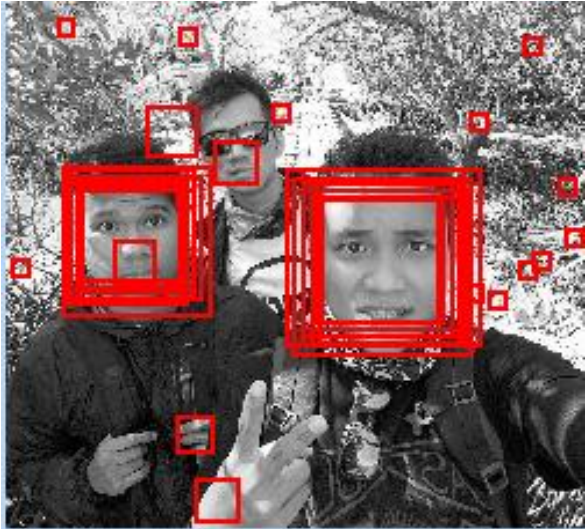

5.1.2 Pengujian menggunakan *haarcascade_frontalface_alt.xml*

Untuk melihat perbedaan hasil pendeteksian antara *haarcascade_frontalface_default* dengan *haarcascade_frontalface_alt*, maka dilakukan pengujian kembali dengan menggunakan sampel yang sama. Tabel 4.3 di berikut ini merupakan hasil dari pengujian pendeteksian wajah dengan menggunakan tipe *haarcascade_frontalface_alt.xml*.

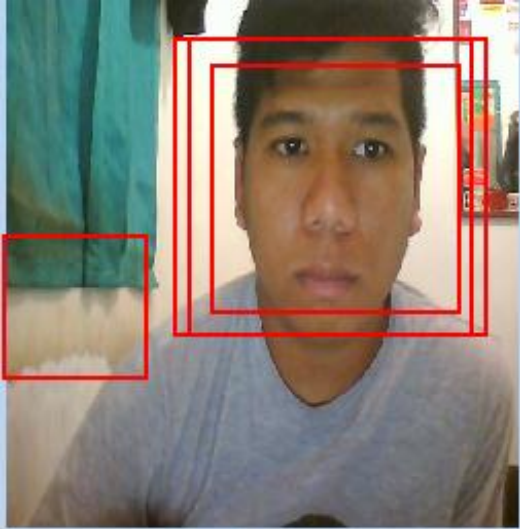
Tabel 4.3 Hasil pengujian menggunakan *haarcascade_frontalface_alt.xml*.

No.	Hasil Pendeteksian			Keterangan
1.				Nama File : 1.jpg Dimensi : 911x816 pixel
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	true	ada	14	53 detik
No.	Hasil Pendeteksian			Keterangan
2.				Nama File : 2.jpg Dimensi : 768x1024 pixel
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	true	ada	30	59 detik

Tabel 4.3 Hasil pengujian menggunakan *haarcascade_frontalface_alt.xml*.

No.	Hasil Pendeteksian			Keterangan
3.				<p>Nama File : 3.jpg Dimensi : 1435x1298 pixel</p>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	true	ada	43	2 menit 7 detik
No.	Hasil Pendeteksian			Keterangan
4.				<p>Nama File : 4.jpg Dimensi : 550x412 pixel</p>
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	false	tidak ada	0	25 detik

Tabel 4.3 Hasil pengujian menggunakan *haarcascade_frontalface_alt.xml*.

No.	Hasil Pendeteksian			Keterangan
5.				Nama File : 5.jpg Dimensi : 580x600 pixel
	Wajah Terdeteksi	Terdeteksi Bukan Wajah	Jumlah Objek Wajah Terdeteksi	Waktu Pendeteksian
	true	ada	4	27 detik

Dari lima buah sampel yang sudah diuji dengan menggunakan *haarcascade_frontalface_alt.xml* bisa disimpulkan bahwa secara umum didapatkan hasil yang tidak jauh berbeda dengan menggunakan *haarcascade_frontalface_default.xml*. Pendeteksian pada citra foto yang di dalamnya terdapat wajah, secara keseluruhan terdeteksi namun masih ada beberapa objek wajah yang tidak terdeteksi ataupun objek bukan wajah yang terdeteksi sebagai wajah. Selain itu juga tidak terdeteksi wajah pada citra yang menampilkan foto binatang.

Berbeda dengan tipe haar yang digunakan pada pengujian sebelumnya, waktu pendeteksian dengan menggunakan *haarcascade_frontalface_alt.xml* cenderung lebih cepat walaupun pada sampel 3.jpg memakan waktu lebih lama dibandingkan dengan pengujian menggunakan *haarcascade_frontalface_default.xml*. Akan tetapi secara keseluruhan waktu pendeteksian wajah menggunakan *haarcascade_frontalface_alt.xml* cenderung lebih cepat di bandingkan dengan pendeteksian menggunakan *haarcascade_frontalface_default.xml*.