BAB IV

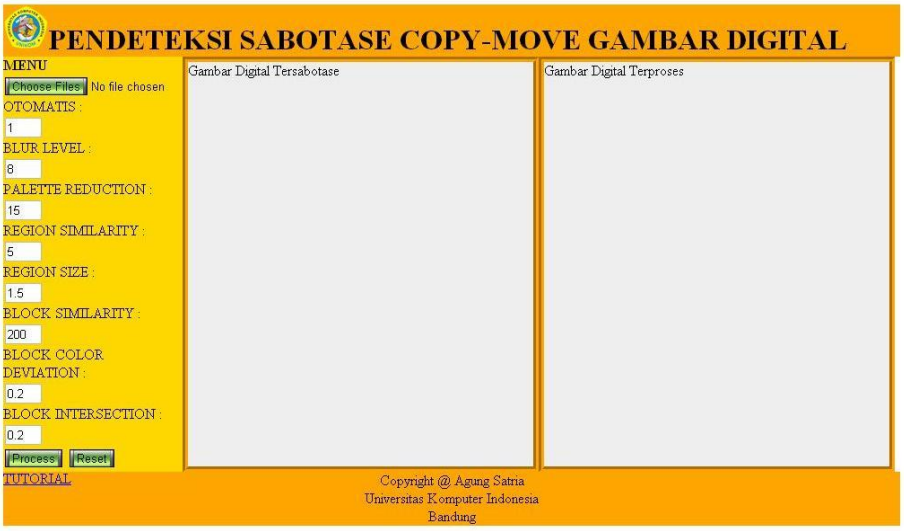
HASIL DAN PENGUJIAN

Program pendeteksi sabotase gambar digital ini memiliki kemampuan mendeteksi objek di dalam sebuah gambar digital yang merupakan hasil duplikasi dari objek lain di dalam gambar digital tersebut. Secara sederhana program ini mengekstrak sebuah gambar digital ke dalam *grid pixel* (blok piksel) lalu mengelompokkannya dan mencocokkannya satu persatu.

Di dalam antarmuka program terdapat beberapa pengaturan yang harus diperkirakan *user* dengan gambar digital yang akan diproses. Pengaturan tersebut berupa parameter – parameter sebagai berikut :

1. Otomatis adalah pencarian dengan adanya perhitungan jumlah daerah identik yang ditemukan oleh aplikasi (1).
2. Blur level adalah menyatakan tingkat level dari *blur* (10).
3. Palette reduction adalah nilai tingkatan palet dari gambar digital (15).
4. Region similarity adalah level *threshold* pada bagian yang identik (5).
5. Region size adalah ukuran bidang *threshold* (1.5).
6. Block Similarity adalah ukuran blok *threshold* dari bagian yang identik (200).
7. Block Color Deviation adalah level blok warna *threshold* yang terdeviasi (0.2).
8. Block intersection adalah ukuran perpotongan blok *threshold* (0.2).

Program utama telah memiliki nilai standar dari keseluruhan parameter di atas, jadi *user* bisa saja hanya menentukan lokasi *(directory local)* dimana letak gambar digital yang akan diproses. Parameter – parameter tersebut tersusun seperti pada gambar 4.1 berikut ini :



Gambar 4.1. Tampilan antarmuka program

Pertama kali yang harus dilakukan oleh *user* adalah menentukan lokasi gambar digital dengan menekan *button "Choose File"* (pada *browser Google Chrome)* yang telah disediakan, setelah itu *user* sudah bisa langsung memproses gambar digital tersebut dengan menekan *button* "Proses" tanpa harus men-set parameter – parameter terlebih dahulu, namun jika tidak ditemukan adanya manipulasi objek berupa duplikasi, maka *user* bisa melakukan ulang proses dengan men-set beberapa parameter yang ada.

4.1 Analisa Hasil

Aplikasi "*Digital Image Forensics*" ini telah mampu mendeteksi adanya objek yang terduplikasi dari objek lain yang diambil dari bagian gambar digital tersebut. Jika ada kelompok blok piksel yang cocok maka program akan memberikan warna cyan serta memberi *border* dengan warna magenta (tergantung parameter). Disamping itu juga kita bisa melihat adanya objek atau bidang yang terduplikasi dengan menganalisa adanya kemiripan bentuk arsiran warna cyan dari hasil proses aplikasi.

4.1.1 Sabotase Duplikat Objek

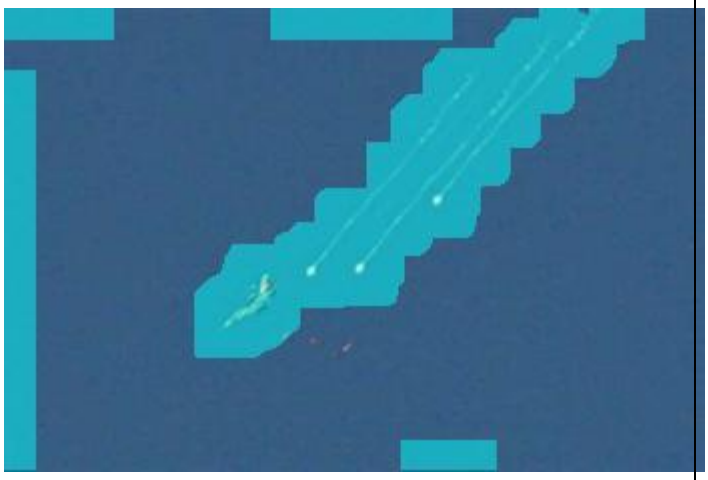
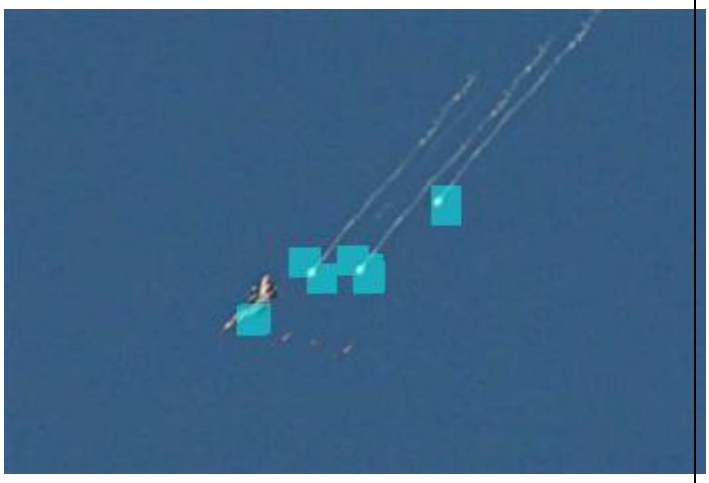
Dalam sabotase objek terdapat kasus menduplikat sebuah objek dengan maksud menambah jumlah objek pada sebuah gambar digital. Berikut akan disajikan sebuah contoh kasus, beserta perbedaaan output aplikasi terhadap perubahan nilai suatu parameter :



Gambar 4.2. Gambar digital tersabotase.[13]

Gambar 4.2 merupakan contoh kasus penduplikasian objek *flares* yang dikeluarkan oleh pesawat F-16. Menurut informasi yang didapat dari sumber sebuah situs, pesawat F-16 mengeluarkan satu buah *flare*, namun kenyataan yang diterima oleh publik melalui media informasi, *flares* tersebut berjumlah tiga buah sehingga publik bahwa itu merupakan sebuah missiles yang dikeluarkan oleh pesawat F-16. Berikut ini dapat dilihat pada tabel 4.1 pengaruh perbedaan nilai dari salah satu parameter yaitu *"Block Color Deviation"* terhadap output dari aplikasi :

Tabel 4.1 Perbedaan output aplikasi terhadap perubahan nilai parameter I

Percobaan	Output Aplikasi	Parameter yang Berubah
1		• Otomatis=0 • Block Color Deviation= 0.001
2		• Otomatis=0 • Block Color Deviation= 0.11

3		• Otomatis=0 • Block Color Deviation= 0.12

Pada percobaan yang ditunjukkan pada tabel 4.1 nomor 1 dilakukan dengan nilai parameter *block color deviation* yang cukup rendah sehingga menghasilkan output arsiran warna cyan yang cukup banyak. Hal ini akan menyulitkan *user* dalam mengidentifikasi adanya objek yang teduplikasi. Pada percobaan kedua nilai parameter *block color deviation* dinaikkan menjadi 0.11 dan menghasilkan daerah arsiran cyan menjadi lebih sempti dari bila dibandingkan dengan percobaan sebelumnya. Output pada percobaan kedua sudah bisa membantu *user* dalam mengidentifikasi adanya objek yang terduplikasi yaitu sebuah *flare* yang dikeluarkan oleh pesawat F-16. Pada percoban ketiga dengan menaikkan satu level nilai parameter *block color deviation* ternyata tidak menghasilkan berupa arsiran.

4.1.2 Sabotase Menutupi Objek

Pada sub-bab ini akan menunjukan hasil yang telah dilakukan pada gambar digital untuk kasus dimana sebuah objek ditutupi objek lain, tetapi yang menutupi objek tersebut tidak dikatakan sebagai objek namun hanya sebagai daerah.

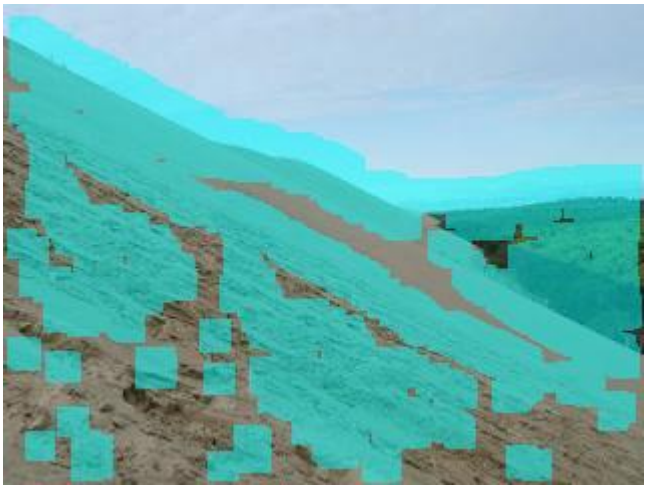
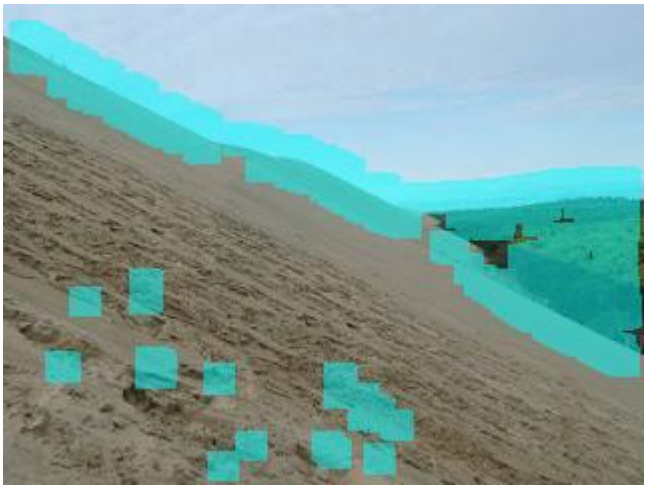
Berikut adalah gambar digital original yanga akan disabotase :

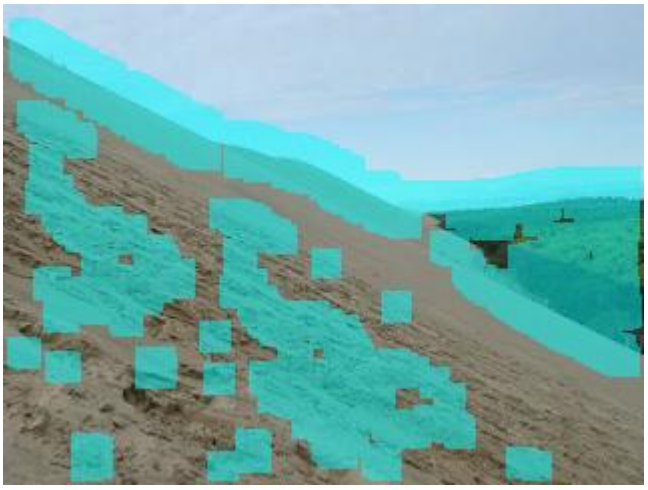
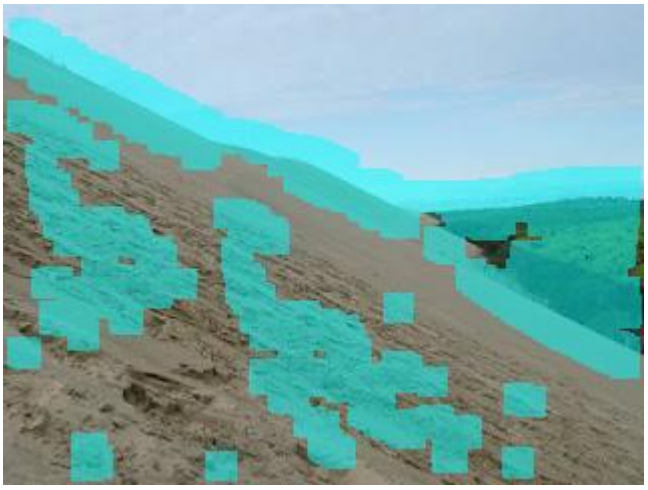


Gambar 4.3.Gambar digital original

Pada tabel 4.2 akan membandingkan perubahan nilai dari sebuah parameter yang diaplikasikan pada satu buah gambar digital.

Tabel 4.2. Perbedaan output aplikasi terhadap perubahan nilai parameter II

Percobaan	Output Aplikasi	Parameter yang Berubah
1		• Otomatis=0 • Block Color Deviation=0.05
2		• Otomatis=0 • Block Color Deviation=0.07

3		• Otomatis=0 • Block Color Deviation=0.06
4		• Otomatis=0 • Block Color Deviation=0.06 • Blur Level=7

Dapat dilihat bahwa sabotase yang terjadi adalah seorang pria yang telah ditutupi oleh sebagian daerah berpasir. Gambar digital pada daerah berpasir, untuk nilai parameter dari *block color deviation* cukup berpengaruh besar pada hasil arsiran. Untuk mendapatkan kemiripan yang lebih pada arsiran daerah terduplikasi pada percobaan ke-empat dilakukan dengan merubah nilai parameter *blur level* yang secara *default*-nya bernilai 8 dirubah menjadi 7. Hal ini menyebabkan daerah arsiran menjadi lebih sempit namun tetap identik.

4.1.3 Sabotase Duplikat dan Tutup Objek

Pada kasus kali ini merupakan contoh kasus duplikat objek untuk menutupi objek lain. Berikut perbedaan antara gambar digital original dengan gambar digital tersabotase :

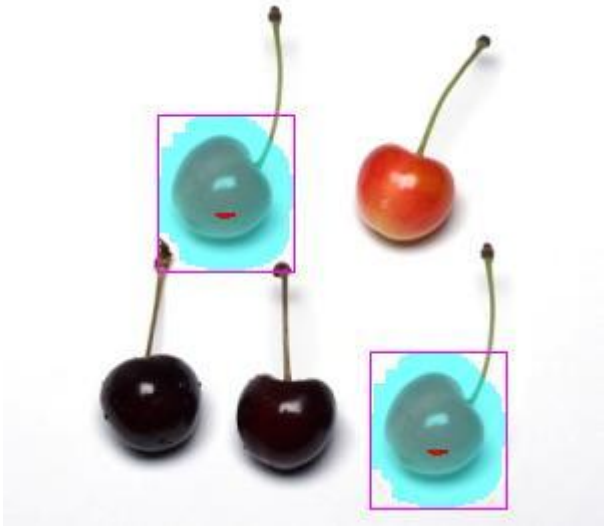
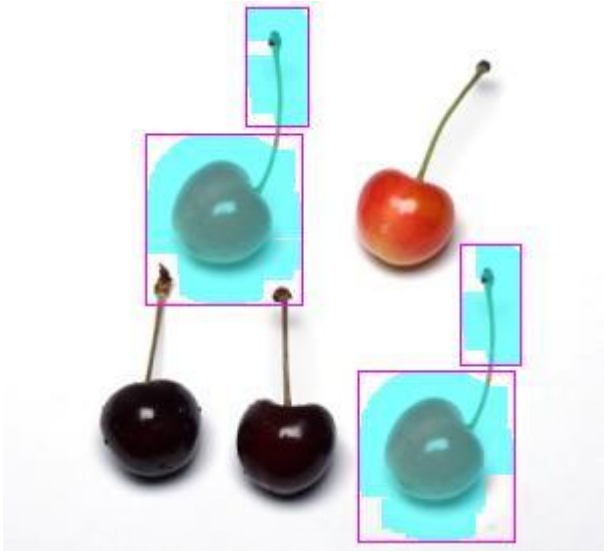


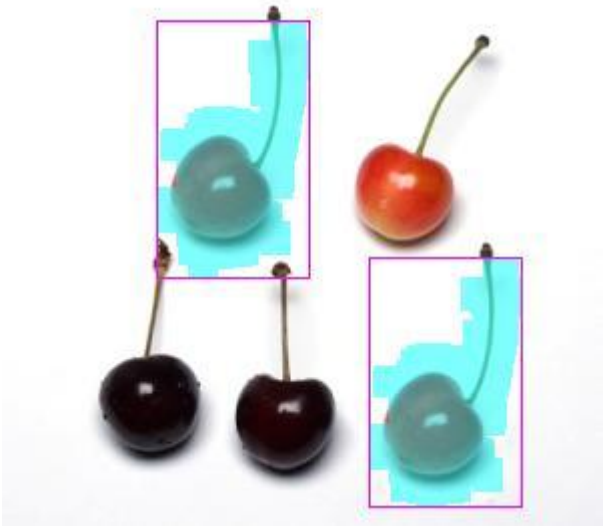
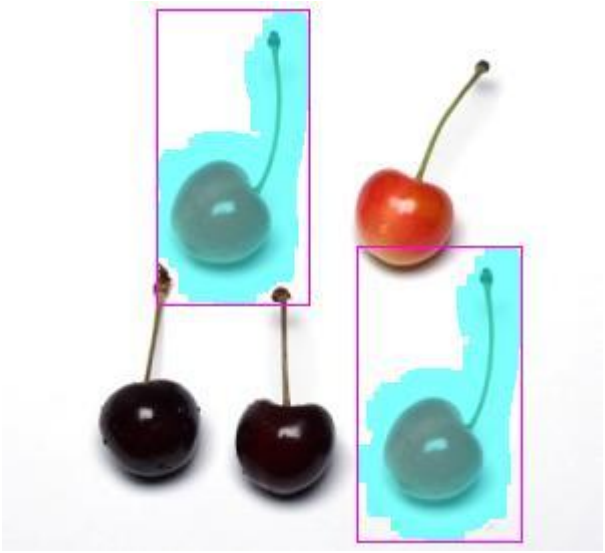
(a) (b)

Gambar 4.4. (a) Gambar digital tersabotase, (b) Gambar digital original

Tabel berikut akan memberikan perbandingan hasil output pada perubahan beberap parameter :

Tabel 4.3. Perbedaan output aplikasi terhadap perubahan nilai parameter III

Percobaan	Output Aplikasi	Parameter yang Berubah
1		• Palettte Reduction = 10
2		• Block Color Deviation = 0.05 • Palette Reduction = 30 • Block Similarity = 100 • Region Similarity = 2

3		• Block Color Deviation = 0.05 • Palette Reduction = 10 • Block Similarity = 100 • Region Similarity = 2
4		• Block Color Deviation = 0.05 • Palette Reduction = 10

Border magenta menunjukkan bahwa aplikasi menemukan objek yang terduplikasi. Ini bisa terjadi jika *user* memberi niali parameter kepada program dengan nilai 1. Pada percobaan 3 dan 4 nilai parameter dari *block similarity* dan *region similarity* mempengaaruhi *margin* arsiran dari objek yang terduplikasi, semakin besar nilainya, maka akan semakin besar pula *margin* arsiran.

Adapun rekap dari semua percobaan yang telah dilakukan dapat dilihat pada tabel 4.4 di bawah ini :

Tabel 4.4 Hasil rekapitulasi dari tabel pengujian

Hasil Pengujian	Input *Image*	Parameter yang berubah	Output *Image*	Pengamatan
Tabel 4.1	Gambar 4.2	• Otomatis = 0 • Block Color Deviation = 0.001	Percobaan 1	Daerah arsiran sangat luas
	Gambar 4.2	• Otomatis = 0 • Block Color Deviation = 0.11	Percobaan 2	Daerah arsiran mengecil signifikan
	Gambar 4.2	• Otomatis = 0 • Block Color Deviation = 0.12	Percobaan 3	Tidak menghasilkan daerah arsiran
Tabel 4.2	Gambar 4.3	• Otomatis = 0 • Block Color Deviation = 0.05	Percobaan 1	Dearah arsiran yang identik belum terlihat
	Gambar 4.3	• Otomatis = 0 • Block Color Deviation = 0.07	Percobaan 2	Daerah arsiran hanya berupa kotak – kotak
	Gambar 4.3	• Otomatis = 0 • Block Color	Percobaan 3	Daerah arsiran

		Deviation = 0.06		identik mulai terlihat
	Gambar 4.3	• Otomatis = 0 • Block Color Deviation = 0.06 • Blur Level=7	Percobaan 4	Daerah arsiran yang identik semakin terlihat jelas
Tabel 4.3	Gambar 4.4	• Palettte Reduction = 10	Percobaan 1	Tidak dapet mengidenti fikasi objek yang kecil
	Gambar 4.4	• Block Color Deviation = 0.05 • Palette Reduction = 30 • Block Similarity = 100 • Region Similarity = 2	Percobaan 2	Mampu mengidenti fikasi objek kecil maupun besar, namun belum sempurna
	Gambar 4.4	• Block Color Deviation = 0.05 • Palette Reduction = 10 • Block Similarity = 100 • Region Similarity = 2	Percobaan 3	Ada bagian kecil dari obejk yang tidak teridentifik asi
	Gambar 4.4	• Block Color Deviation = 0.05 • Palette Reduction	Percobaan 4	Teridentifikasi secara

		= 10		keseluruhan dari objek

4.2 Pengujian Normal dan Tidak Normal

Pada pengujian normal aplikasi akan diberikan data berupa gambar digital serta parameter sebagaimana mestinya, namun ada kalanya bisa saja ada *user* yang mungkin memasukkan data yang salah. Ketika ada *user* yang memasukkan parameter yang salah maka aplikasi harus mampu mengantisipasi hal ini dengan membri peringatan kepada *user* tersebut.

4.2.1 Pengujian Normal

Pemilihan gambar digital seharusnya berformat *.jpg serta parameter yang diberikan merupakan bilangan integer, maka jika kedua hal tersebut dilakukan aplikasi akan berjalan normal sebagaimana dengan mestinya dengan mengeluarkan output gambar digital seperti pada gambar di bawah ini :

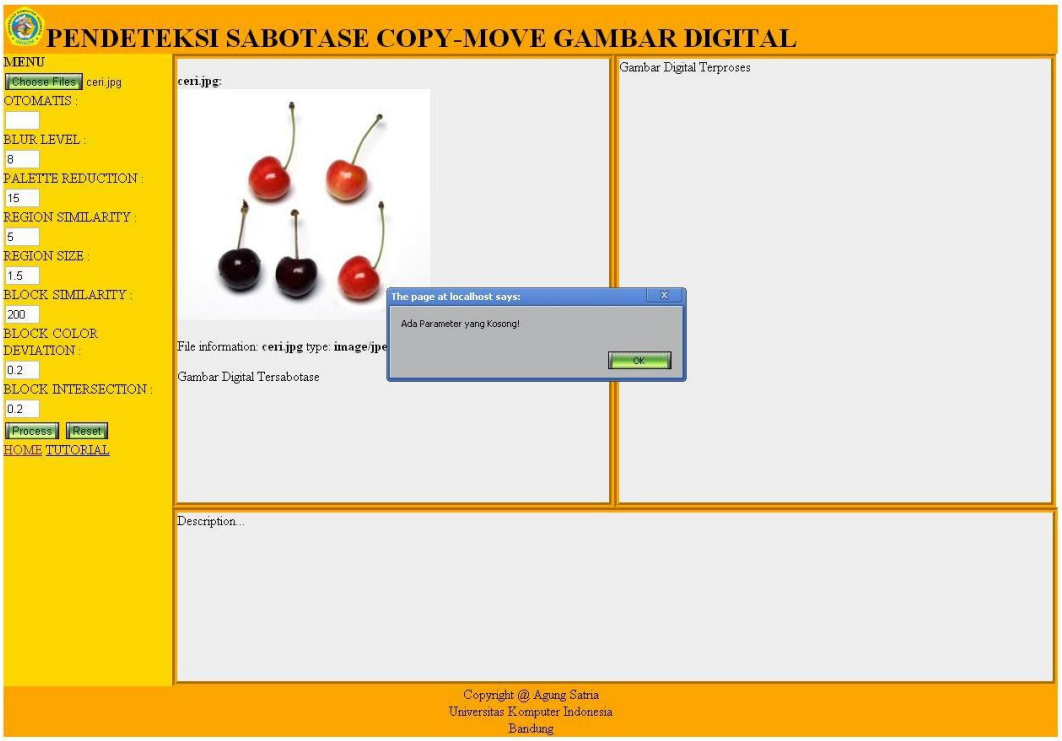


Gambar 4.5. Tampilan pengujian normal

Pada percobaan gambar 4.5 di atas *user* memasukkan gambar digital dengan nama "ceri.jpg", dan parameter *palette reduction* = 10, *block color deviation* = 0.05. gambar digital serta parameter tersebut memenuhi syarat dari aplikasi maka dari itu akan menghasilkan apa yang kita harapkan yaitu memberikan informasi adanya sabotase *copy-move* pada gambar digital seperti yang tampak pada gamabr 4.5 dia atas.

4.2.2 Pengujian Tidak Normal

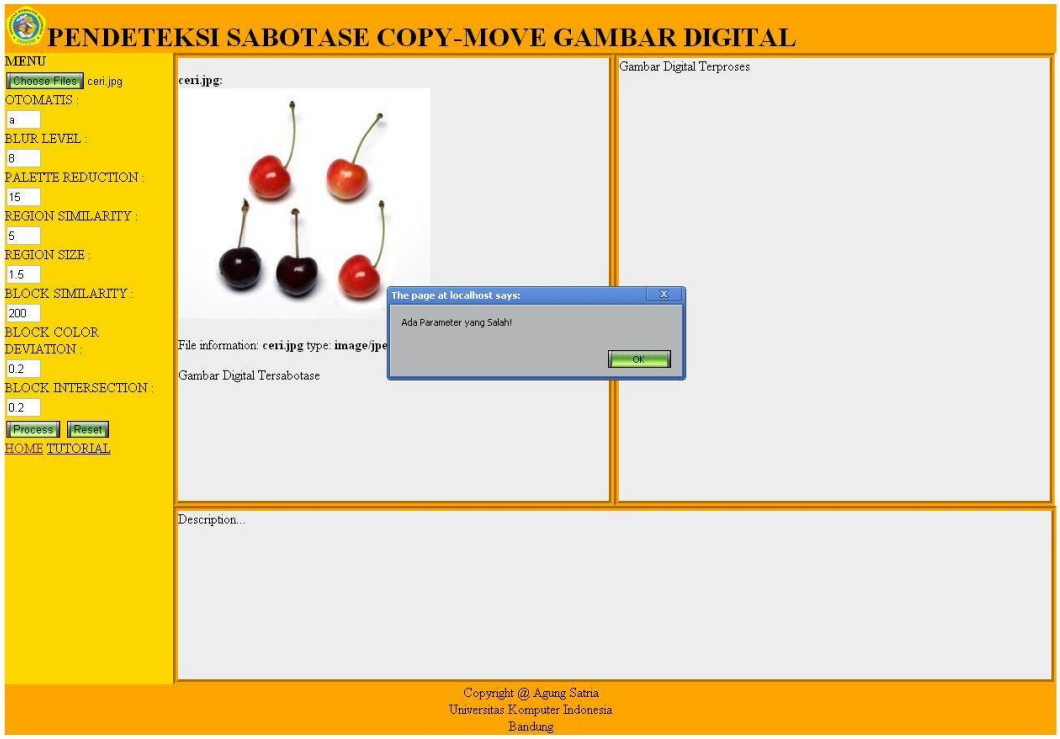
Ada beberapa kemungkinan kesalahan yang terjadi yang bisa dilakukan oleh *user* , terutama salah dalam melakukan set parameter. Program ini telah dilengkapi dengan pemeriksaan parameter yang tidak *valid* yang menyebabkan program tidak berjalan sebagaimana mestinya. Pada pengujian tidak normal diasumsikan bahwa *user* memasukkan sebuah parameter yang salah, dengan begitu aplikasi harus mampu memberikan peringatan kepada *user*, bahwa *user* telah salah dalam memasukkan parameter ataupun gambar digital. Berikut adalah peringatan dari aplikasi apabila ada *field* parameter yang tidak terisi :



Gambar 4.6. Tampilan pengujian tidak normal

Kesalahan *user* pada gambar 4.6 di atas adalah *filed* parameter ada yang kosong, dengan begitu aplikasi mengantisipasi dengan memberikan *alert* dengan kata – kata peringatan “Ada Parameter yang Salah!”, seperti tampak pada gambar 4.6 di atas.

Berikut jika kesalahan user adalah mengisi *filed* parameter dengan sebuah huruf, dalam hal ini misal adalah sebuah huruf “a” :



Gambar 4.7. Tampilan pengujian tidak normal

Kesimpulan dari hasil percoban data yang normal dan data yang tidak normal dapat dilihat pada tabel di bawah ini :

Tabel 4.5. Rekapitulasi percobaan data normal dan tidak normal

Kasus & Hasil Uji (Normal)			
Data Masukan	Diharapkan	Pengamatan	Keterangan
Image dan nilai parameter	Ada output gambar	Terlihat adanya kedua	✓

seluruhnya benar	digital hasil proses	gambar digtal	
Kasus & Hasil Uji (Data tidak Normal)			
Data Masukan	Diharapkan	Pengamatan	Keterangan
Ada *filed* nilai parameter yang kosong	Aplikasi mengeluarkan *alert*, agar proses tidak berlanjut	Aplikasi menampilkan *alert*"Ada Parameter yang Kosong!"	✓
Ada *filed* nilai parameter yang terisi oleh huruf	Aplikasi mengeluarkan *alert*, agar proses tidak berlanjut	Aplikasi menampilkan *alert*"Ada Parameter yang Salah!"	✓