BAB III

METODOLOGI DAN PERANCANGAN

3.1 Metodologi Penelitian

Metode penelitian yang digunakan dalam penelitian ini antara lain:

- Studi literatur

  Melakukan studi dan mencari informasi pada penelitian-penelitian terkait mengenai metode *no reference quality metrics*, *image processing* pada Microsoft Visual Studio, penggunaan *library* EmguCV pada bahasa pemrograman C# dan berbagai konsep lainnya. Studi literatur dapat bersumber pada jurnal ilmiah, buku, dan situs web.

- Perancangan

  Perancangan aplikasi terdiri dari pembuatan alur kerja aplikasi dalam bentuk diagram alir dan merancang desain antar muka aplikasi.

- Implementasi

  Melakukan pengembangan aplikasi sesuai dengan rancangan dan desain yang sebelumnya telah didefinisikan menggunakan bahasa pemrograman yang telah ditentukan.

- Uji coba dan evaluasi

  Melakukan uji coba terhadap aplikasi yang telah dirampungkan dan melakukan evaluasi berdasarkan hasil yang telah didapatkan dari kegiatan uji coba.

3.2 Spesifikasi Umum Kebutuhan Aplikasi

Spesifikasi umum yang dapat menunjang kebutuhan, masukan dan keluaran pada aplikasi *Image Forgery Detector* dapat diuraikan dan dijelaskan menggunakan diagram aplikasi.

3.2.1 Diagram Aplikasi

Pada aplikasi *Image Forgery Detector* ini terdapat beberapa runtutan proses yang digunakan untuk memproses masukan berupa gambar yang terkena efek *splicing* dan mengeluarkan hasil berupa gambar hitam putih yang mengindikasikan adanya kegiatan pemalsuan pada gambar. Adapun kegiatan yang dilakukan antara lain membuat *blocking map*, *activity map*, dan *zero crossing map* sesuai dengan perhitungan yang ada, kemudian menggabungkannya menjadi suatu gambar baru yang dapat mengindikasikan area adanya pemalsuan. Penjelasan lebih lanjut digambarkan pada diagram 3.1.

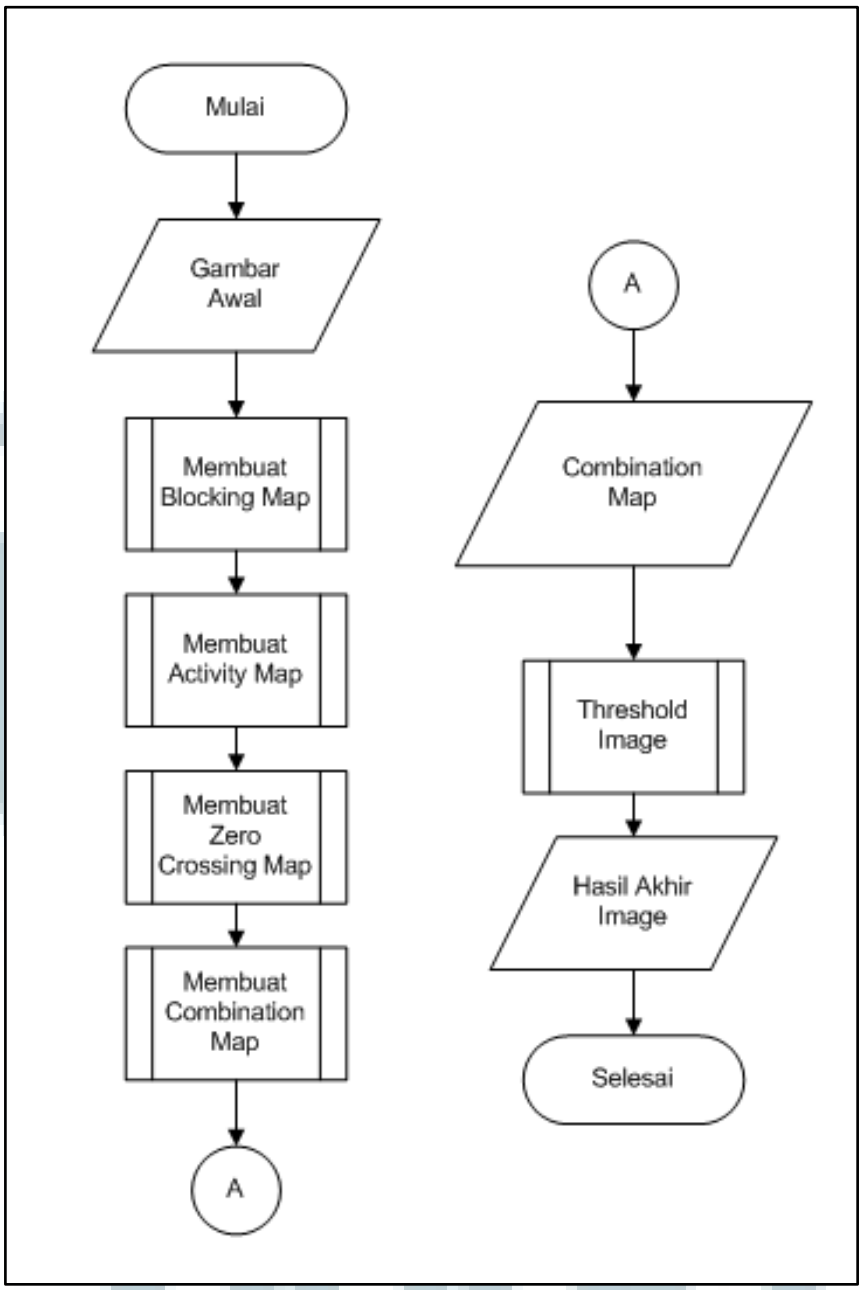


Diagram 3.1 Diagram alir aplikasi Image Forgery Detector

3.2.2 Fungsionalitas Sistem

Aplikasi ini mempunyai beberapa fungsionalitas antara lain:

- Menghitung koefisien *blockiness*, *pixel activity* dan *zero crossing rate* pada suatu gambar

- Mengkonversi komponen warna gambar dari RGB menjadi YCbCr untuk melakukan efek *false coloring* pada gambar sehingga mempermudah ekstraksi fitur pada gambar
- Menghasilkan gambar baru yang bersifat biner dengan menggunakan efek *thresholding* untuk mempermudah pendeteksian bagian yang dipalsukan pada gambar.

3.2.3 Masukan dan Keluaran Aplikasi

Masukan dari aplikasi ini adalah sebuah gambar dengan format lossy atau lossless, contohnya gambar dengan format JPG atau TIFF. Sedangkan keluaran dari aplikasi adalah gambar biner hitam putih yang memuat informasi daerah – daerah gambar yang terindikasi merupakan hasil pemalsuan dengan metode *image splicing*.

3.3 Perancangan Aplikasi

Aplikasi dikembangkan menggunakan piranti lunak Microsoft Visual Studio 2010 dengan menggunakan bahasa pemrograman C#. Berikut ini dijelaskan setiap *subroutine* yang ada pada aplikasi dalam bentuk diagram alir dan tampilan antar muka pengguna aplikasi.

3.3.1 Perancangan Subroutine

Desain *subroutine* atau fungsi – fungsi yang dipakai oleh Image Forgery Detector dapat dijabarkan dan digambarkan menggunakan diagram alir sebagai berikut.

A. Subroutine Load Image

Proses pertama yang dilakukan untuk memulai proses ektraksi gambar adalah dengan cara menentukan gambar mana yang ingin dipakai untuk dilakakukan proses pengecekan. Jika aplikasi tidak mampu menampilkan dokumen yang diminta maka aplikasi akan mengeluarkan *error message*, sebaliknya jika berhasil maka proses akan dilanjutkan. Setelah gambar berhasil di-*load* maka aplikasi akan membuat objek dengan tipe image dengan gambar yang terpilih sebagai inputnya. Setelah itu dibuat objek dengan tipe YCbCr Image sebagai *container* atau penampung. Hal ini dikarenakan source image mempunyai komponen warna RGB sedangkan untuk proses selanjutnya bermain dalam komponen warna YCbCr. Selain itu array dua dimensi dengan tipe data double juga diinisialisasikan. Array dibuat untuk menampung nilai perhitungan *blocking*, *activity* dan *zero crossing* untuk setiap komponen warna Y, Cb, dan Cr; sehingga terdapat total sembilan array dua dimensi dengan tipe data double yang mempunyai jumlah kolom dan baris sebesar ukuran panjang dan lebar gambar. Proses *load image* dapat dijelaskan oleh diagram 3.2.

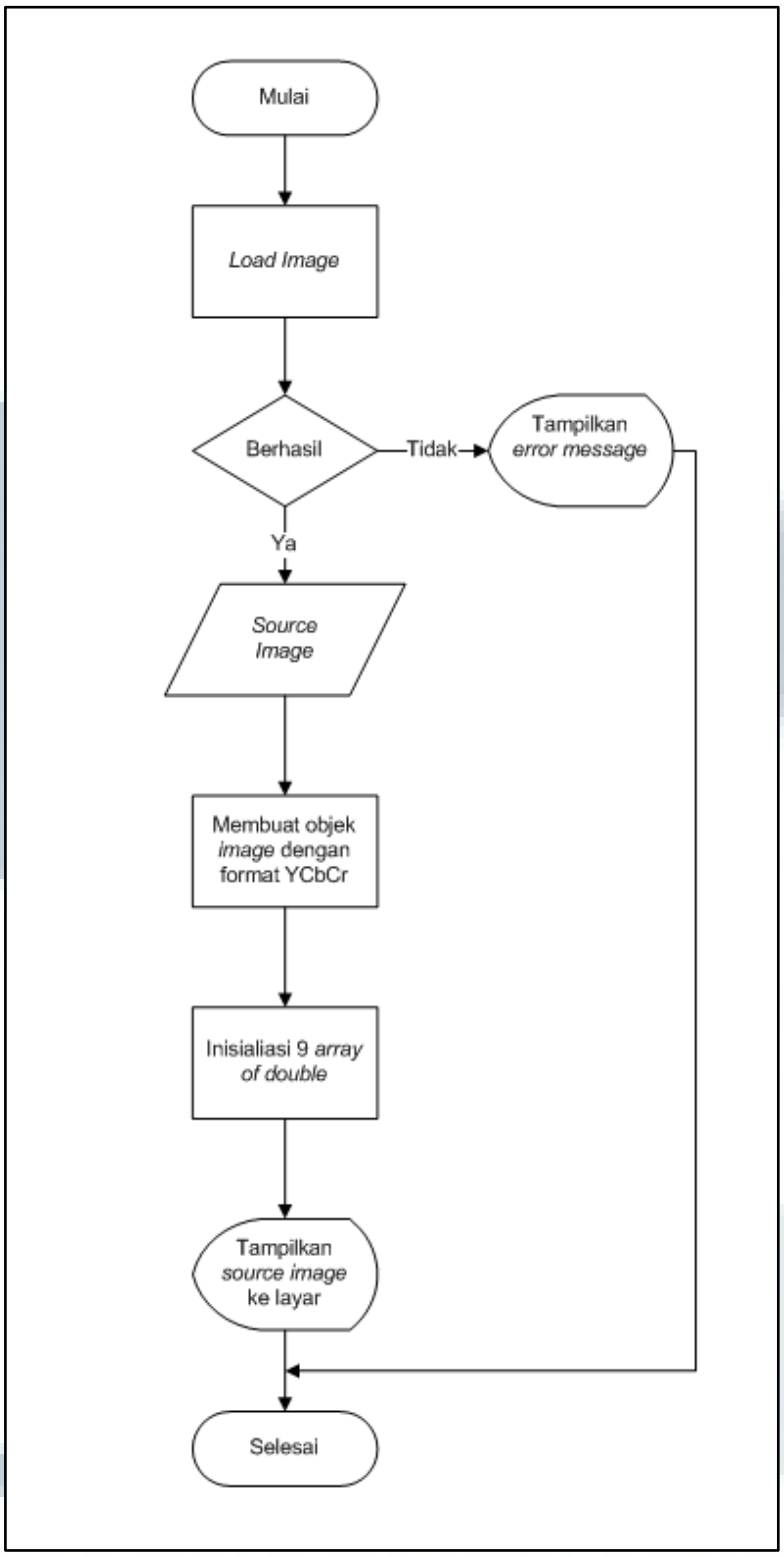


Diagram 3.2 Diagram alir *subroutine load image*

B. Subroutine Sum of Absolute Difference

Sebelum dapat melakukan proses pembuatan blocking map, activity map dan zero crossing map, tahap awal yang harus dilakukan adalah mencari nilai *Sum of Absolute Difference* atau SAD dari gambar tersebut. Hasil dari perhitungan SAD sangat penting karena akan banyak digunakan pada proses-proses selanjutnya. Nilai SAD didapat dengan cara menjumlah semua nilai selisih dari piksel gambar yang saling bersebelahan (*neighboring pixel*). SAD dihitung dengan menggunakan formula yang tertera pada rumus 2.6. Diagram alir dari SAD dapat dilihat pada diagram 3.3.

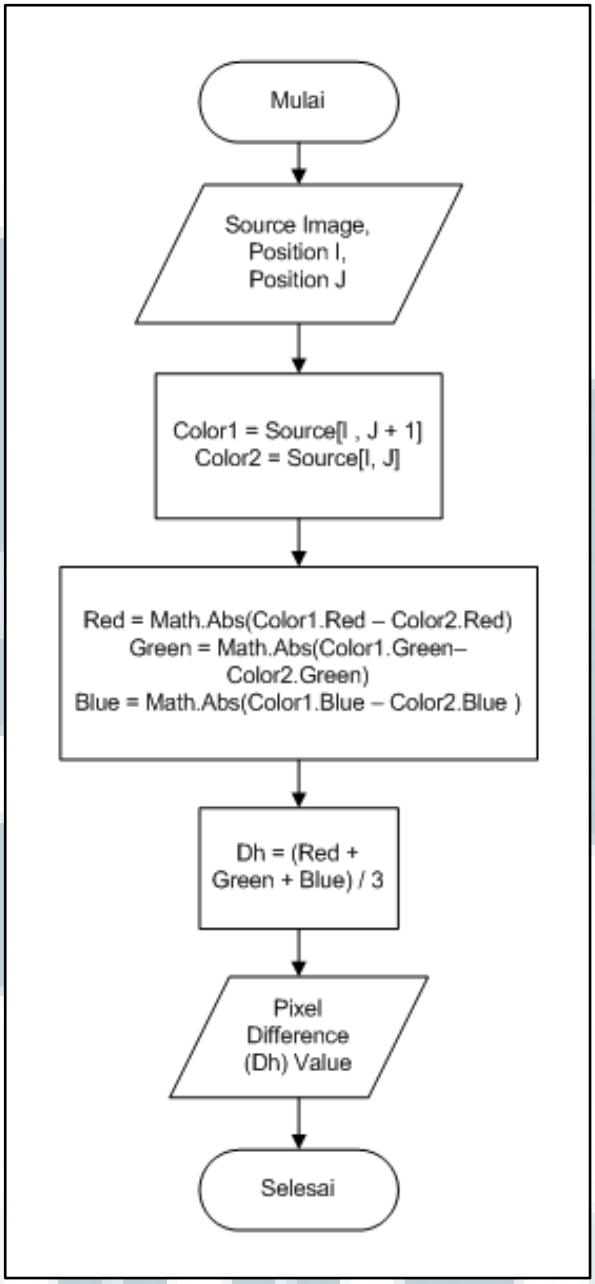


Diagram 3.3 Diagram alir *Sum of Absolute Difference*

C. Subroutine Blocking Map

Subroutine ini mempunyai input berupa *source image* dan mengembalikan nilai koefisien blockiness pada gambar. Untuk mendapatkan blocking map maka diperlukan beberapa runtutan proses. Proses diawali dengan menghitung nilai *Sum of Absolute Difference*. Setelah nilai SAD didapat maka selanjutnya adalah melakukan perhitungan koefisien *blockiness* sesuai dengan perhitungan pada rumus 2.2. Koefisien blockiness kemudian dikalikan ke setiap piksel pada komponen warna Y, Cb, dan Cr untuk mendapatkan *blocking map* untuk setiap komponen warna yang ada. Proses ini diulang sebanyak jumlah piksel yang terdapat pada gambar tersebut. Diagram alir proses pembuatan *blocking map* dapat dilihat pada diagram 3.4.

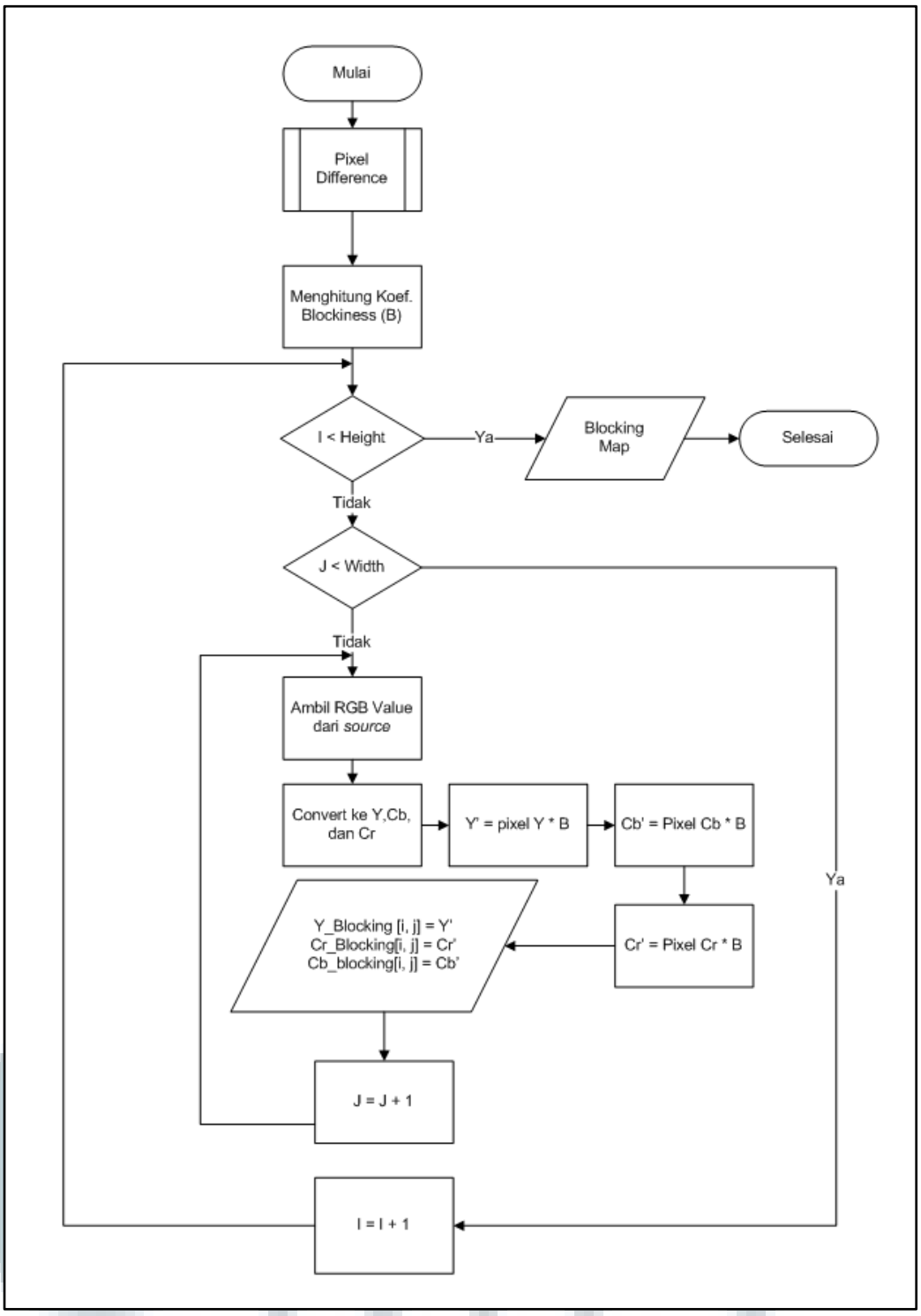


Diagram 3.4 Diagram alir *blocking map*

D. *Subroutine* Activity Map

Subroutine pada pembuatan *activity map* memiliki proses yang serupa dengan pembuatan *blocking map* hanya saja formula yang dipakai untuk menghitung koefien yang dipakai bukanlah koefisien blockiness pada piksel melainkan koefisien aktifitas piksel. Formula perhitungan koefisien aktifitas piksel dapat dilihat pada rumus 2.4. Diagram alir pembuatan activity map dapat dilihat pada diagram 3.5.

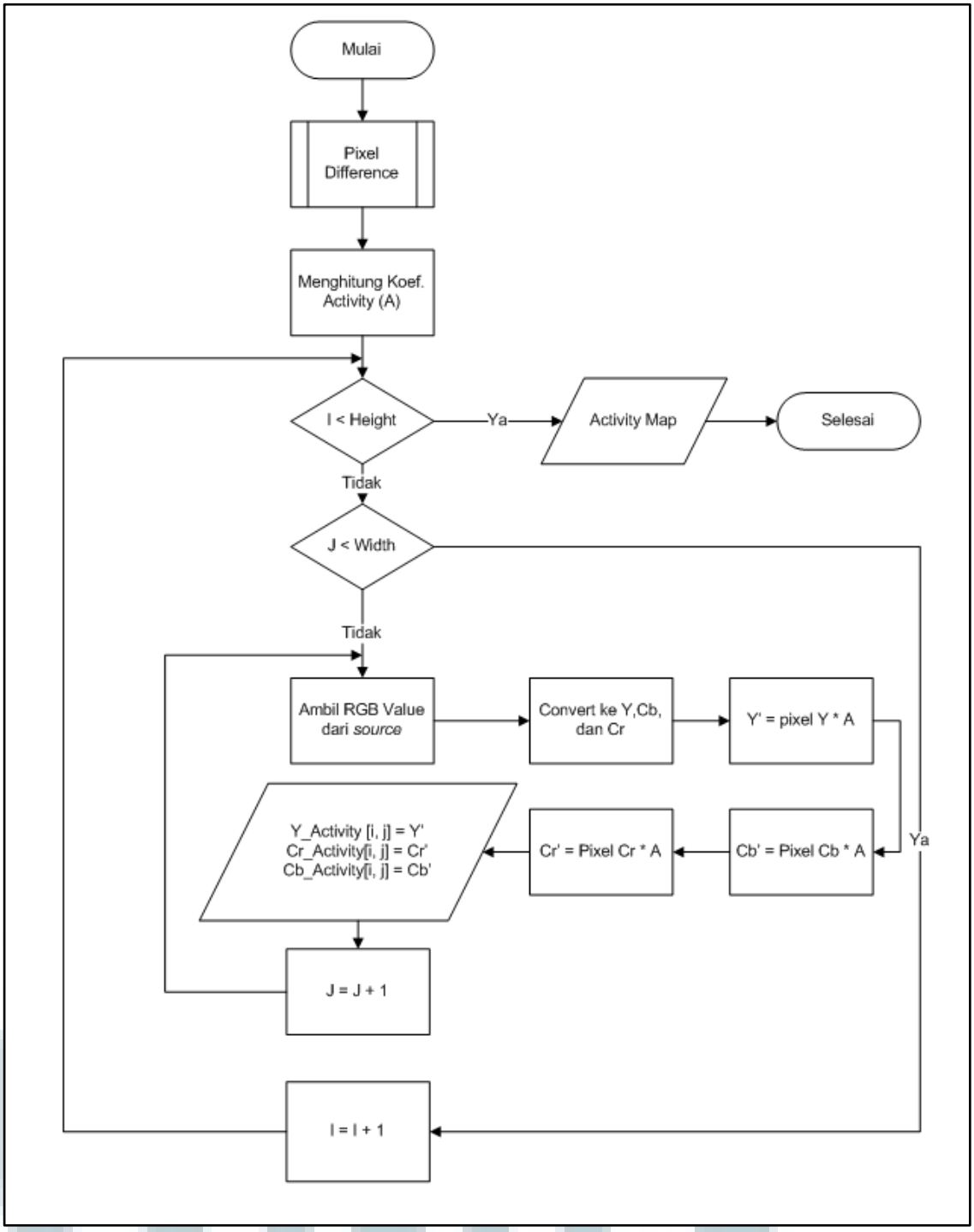


Diagram 3.5 Diagram alir *activity map*

E. Subroutine Zero Crossing Map

Proses pembuatan *zero crossing* map diawali dengan prosedur pengecekan nilai selisih piksel (Dh) pada setiap piksel yang ada didalam sebuah gambar. Nilai Dh kemudian digolongkan berdasarkan apakah nilai Dh melewati angka nol atau tidak. Jika ya maka nilai *zero crossing* pada Dh adalah satu, selain dari pada itu diberi nilai nol. Nilai *zero crossing* kemudian diakumulasikan untuk kemudian dipakai untuk menghitung koefisien *zero crossing* pada gambar. Formula perhitungan *zero crossing* dapat dilihat pada rumus 2.5.

Selanjutnya prosedur pembuatan *zero crossing map* tidak berbeda dengan pembuatan map lainnya, yaitu dengan cara mengalikan koefisien yang didapat dengan piksel – piksel disetiap komponen warna yang terdapat pada gambar. Diagram alir *zero crossing map* dapat dilihat pada diagram 3.6.

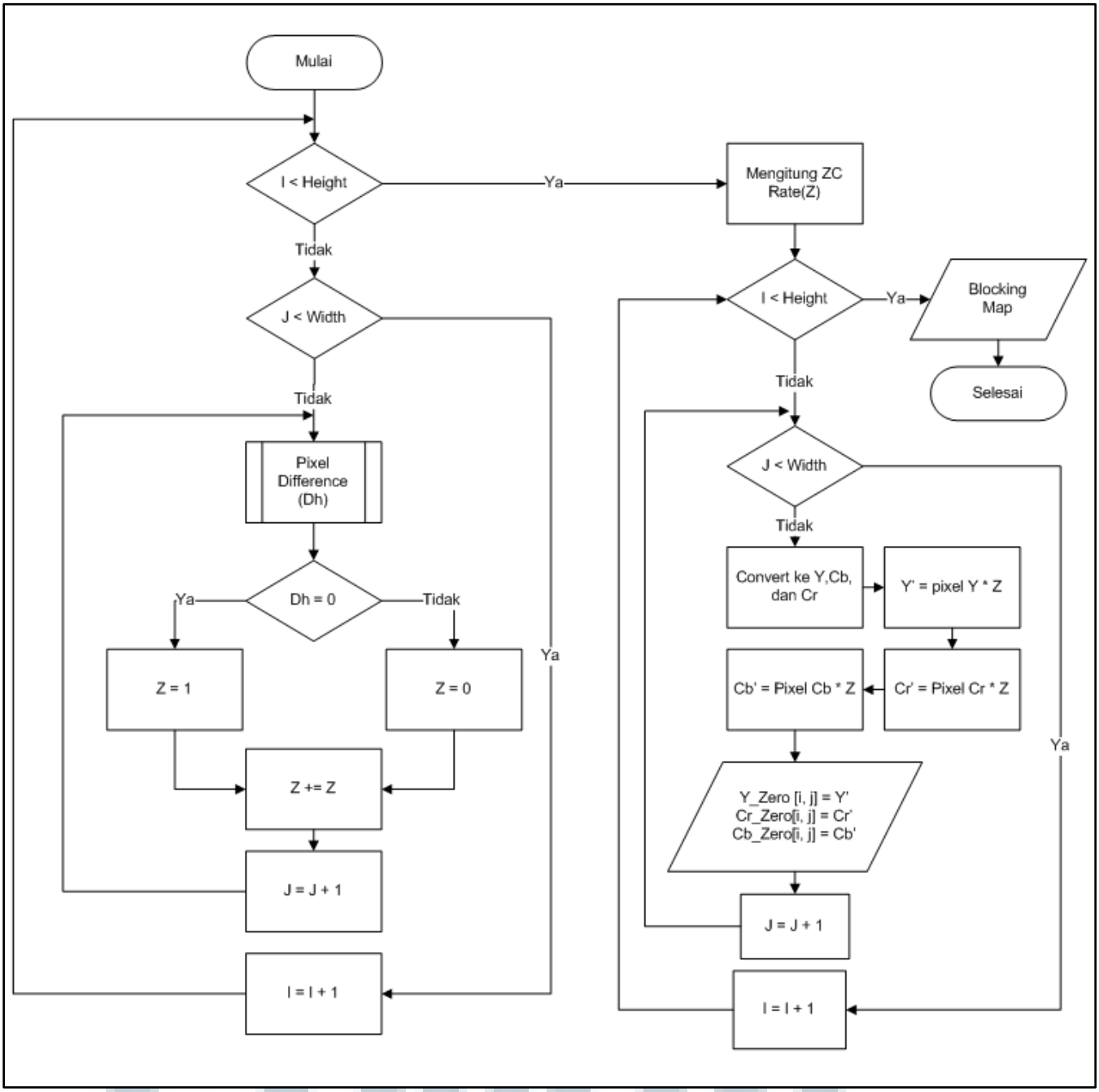


Diagram 3.6 Diagram alir zero crossing map

F. Subroutine Make Combination Map

Setelah *blocking*, *activity*, dan *zero crossing map* dibuat maka tahap selanjutnya adalah menggabungkan ketiga map tersebut menjadi suatu gambar baru yang diharapkan dapat melacak daerah yang terindikasi telah dipalsukan. Diagram alir pembuatan *combination map* dapat dilihat pada diagram 3.7.

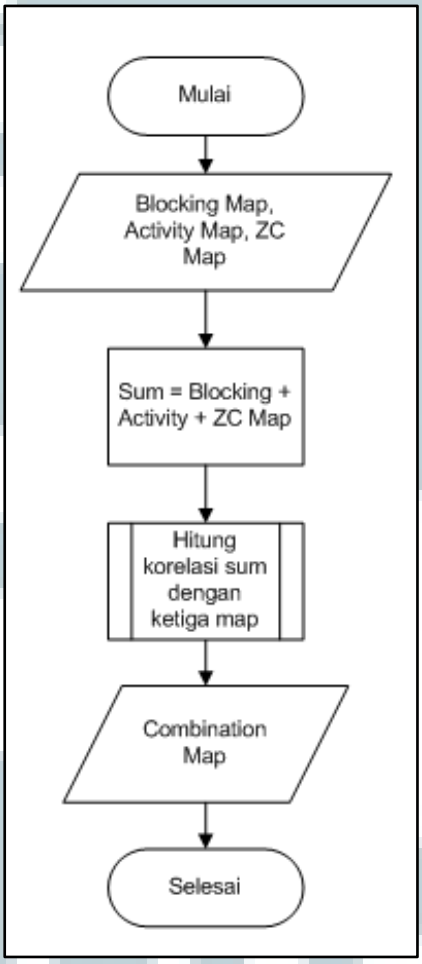


Diagram 3.7 Diagram alir *combination map*

Penggabungan *blocking map*, *activity map* dan *zero crossing map* akan menghasilkan *sum map* yang memuat nilai gabungan dari ketiga *map*

tersebut. Proses selanjutnya adalah menghitung korelasi antara *sum map* dengan setiap komponen warna yang ada pada *blocking map*, *activity map* dan *zero crossing map* dengan tujuan untuk menselaraskan satu piksel dengan piksel – piksel tetangganya. Proses ini bertujuan untuk membedakan *noise* pada gambar asli dengan *noise* pada bagian yang palsu. Setelah perhitungan korelasi selesai maka telah didapatkan *combination map*.

G. Subroutine Correlation

Fungsi dari *subroutine* ini adalah menghitung korelasi antar piksel yang ada pada *Sum map* dengan fitur map untuk setiap komponen warnanya. Oleh karena itu akan terdapat sembilan koefisien korelasi yakni koefisien korelasi untuk blocking map pada warna Y, Cb, dan Cr; koefisien korelasi untuk activity map pada warna Y, Cb, dan Cr dan koefisien korelasi untuk zero crossing map pada warna Y, Cb, dan Cr. Perhitungan korelasi dihitung berdasarkan rumusan koefisien korelasi linear Pearson. Diagram alir perhitungan korelasi dapat dilihat pada diagram 3.8.

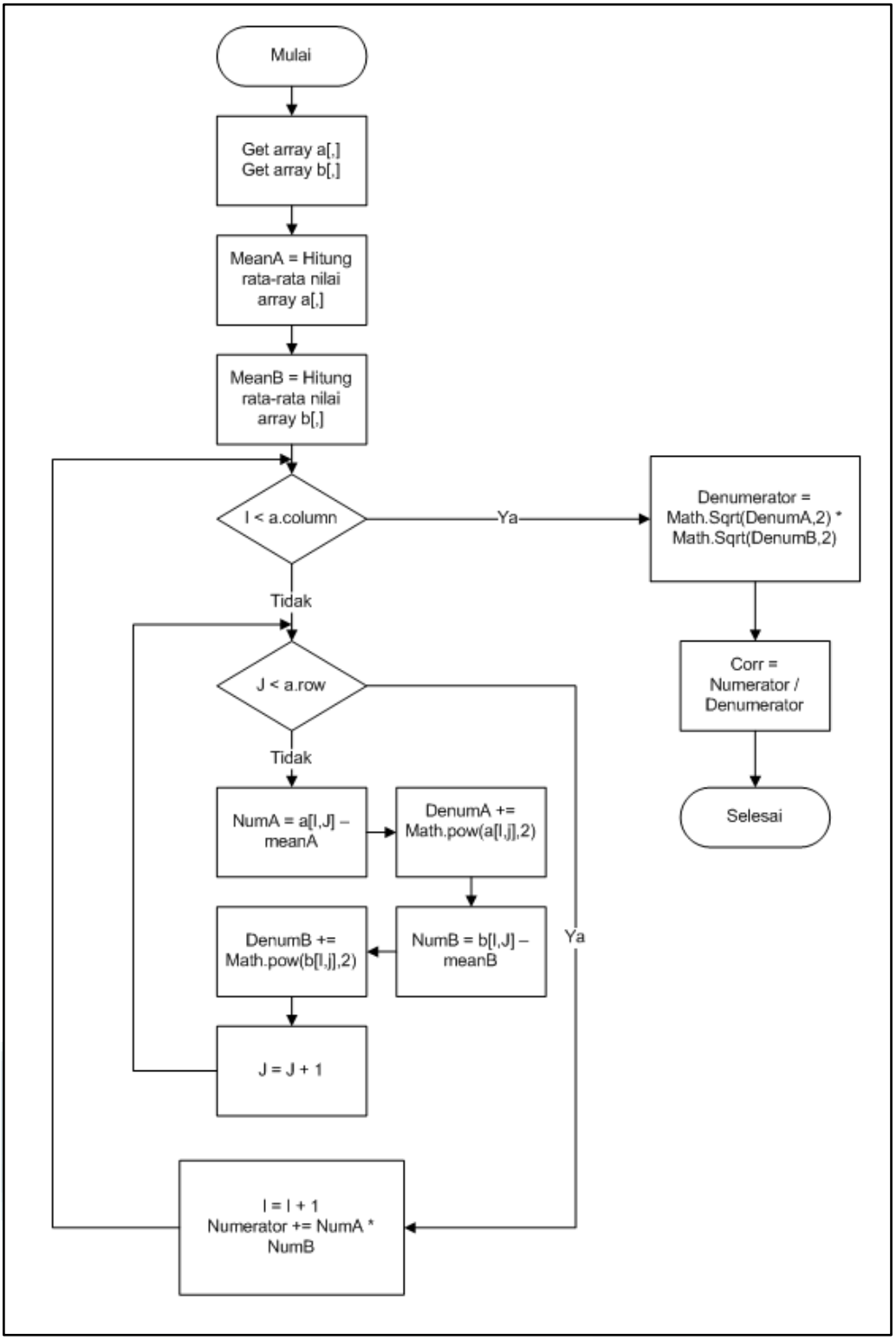


Diagram 3.8 Diagram alir perhitungan korelasi

H. Subroutine Threshold

Fungsi dari *subroutine* ini adalah untuk mengelompokan nilai – nilai piksel menjadi dua bagian. Pengelompokan dilakukan dengan tujuan untuk mempermudah interpretasi citra sehingga memudahkan pengguna membedakan mana bagian yang dianggap palsu dan mana yang bukan.

Proses awal *thresholding* adalah dengan cara menentukan nilai threshold awal. Tidak ada aturan khusus berapa nilai yang harus dipakai untuk nilai awal. Setelah itu nilai per piksel dibandingkan dengan nilai *threshold* tersebut. Apabila lebih besar maka akan dimasukan ke dalam kelompok bagian yang asli sedangkan sisanya dikelompokan ke dalam kelompok bagian yang terindikasi palsu. Proses threshold ini diiterasi sampai dengan bagian palsu (*tempered*) pada gambar terlihat. Diagram alir proses threshold dapat dilihat pada diagram 3.9.

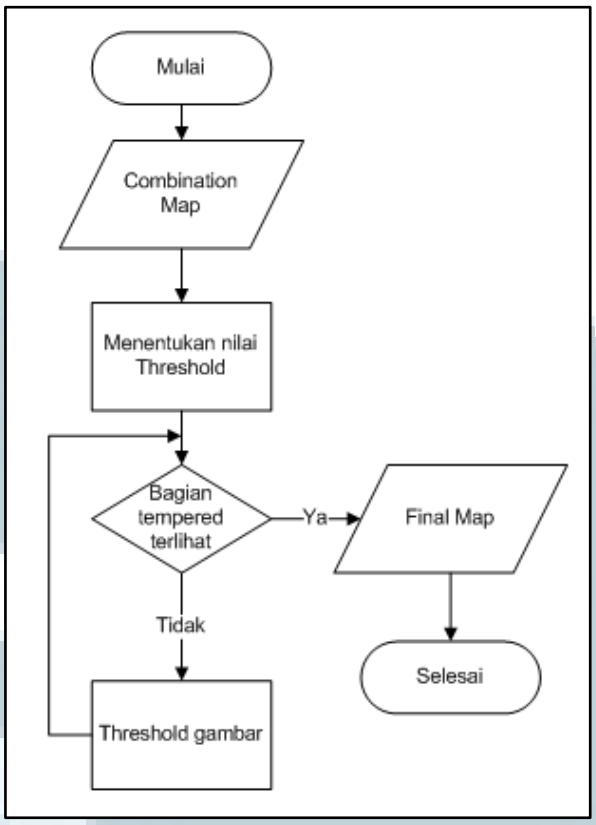
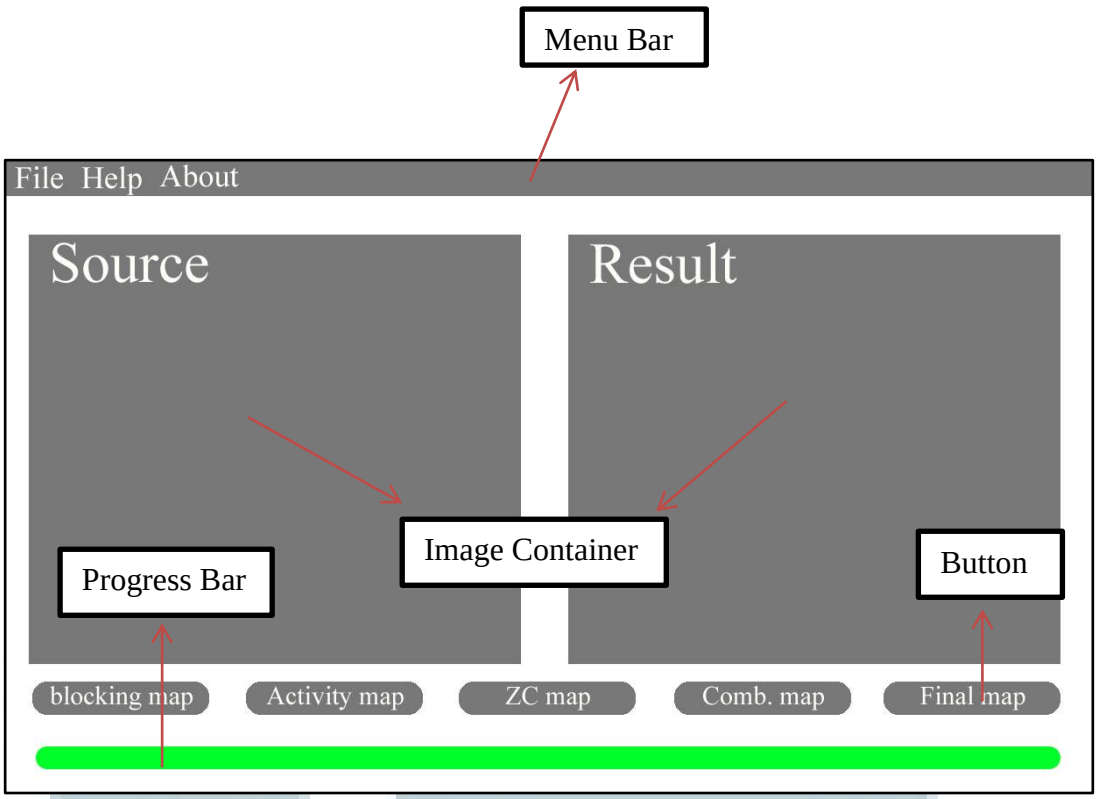


Diagram 3.9 Diagram alir proses *threshold*

3.3.2 Perancangan Antarmuka

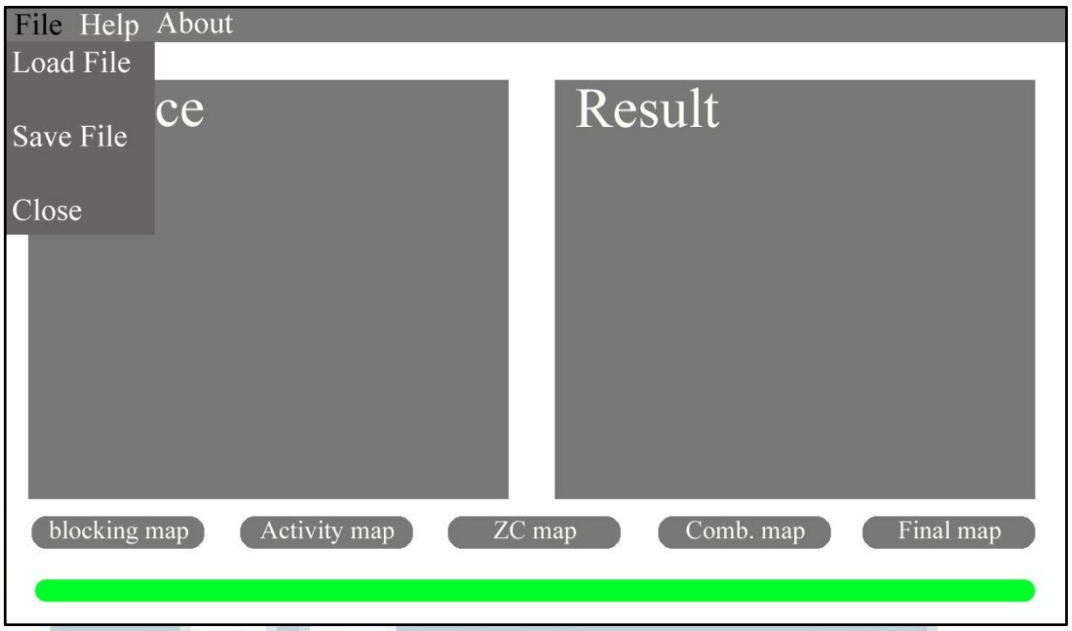
Tampilan antarmuka aplikasi *Image Forgery Detector* sepenuhnya dirancang menggunakan aplikasi Microsoft Visual Studio 2010. Keunggulan dari Microsoft Visual Studio adalah kemudahan yang ditawarkan untuk membuat desain antar muka yakni dengan menggunakan metode *drag and drop* sehingga programmer dapat langsung melihat dan mengubah secara langsung tampilan yang dibuat (*what you see is what you get*).

Tampilan antarmuka aplikasi dibagi menjadi tiga bagian, yaitu bagian atas, tengah dan bawah. Pada bagian atas aplikasi terdapat *menu bar* yang memuat menu *file, help* dan *about*. Sedangkan bagian tengah aplikasi terdapat dua *image container* yang letaknya saling bersebelahan. *Image container* di bagian kiri berfungsi untuk menampilkan gambar asal (*source image*), sedangkan bagian *image container* kanan berfungsi untuk menampilkan gambar hasil dari proses pemeriksaan (*result image*). Kedua *image container* diletakkan saling bersebelahan agar mempermudah pengguna melakukan kegiatan perbandingan antara kedua gambar. Di bagian bawah *image container* terdapat lima buah tombola atau *button*. Tombol – tombol tersebut antara lain tombol *blocking map, activity map, zero crossing (ZC) map, combination map* dan *final map*. Setiap tombol mempresentasikan proses yang dilakukan pada aplikasi pada *source image* dan menampilkannya ke *result image container*. Selanjutnya pada bagian bawah aplikasi terdapat *progress bar* yang berfungsi untuk memberi tahu pengguna progress daripada proses yang dilakukan oleh aplikasi. Desain antarmuka aplikasi *Image Forgery Detector* dapat dilihat pada gambar 3.1.



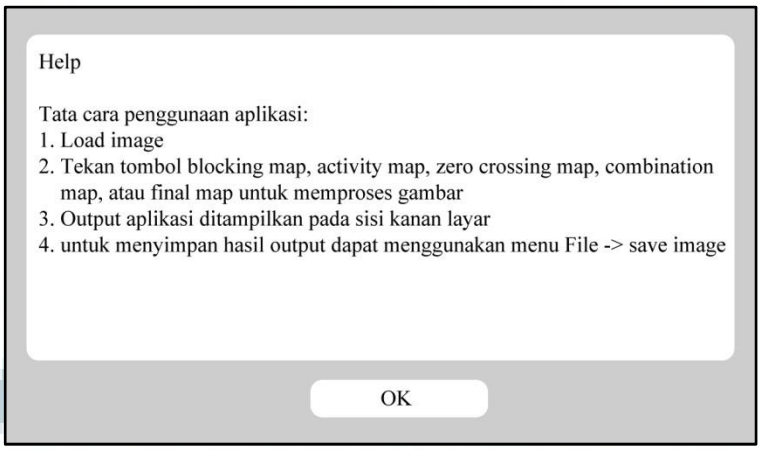
Gambar 3.1 Desain antarmuka aplikasi

Menu yang terdapat pada *menu bar* digolongkan berdasarkan jenisnya. Menu file menggolongkan menu yang sifatnya berinteraksi dengan sebuah file. Apabila menu file ditekan maka terdapat sub menu file antara lain load image yang digunakan untuk membuka *source image*, *save image* yang berfungsi untuk menyimpan gambar output dari aplikasi dan *close* yang digunakan untuk keluar dari aplikasi. Tampilan sub menu file dapat dilihat pada gambar 3.2.



Gambar 3.2 Tampilan antarmuka saat menu file ditekan

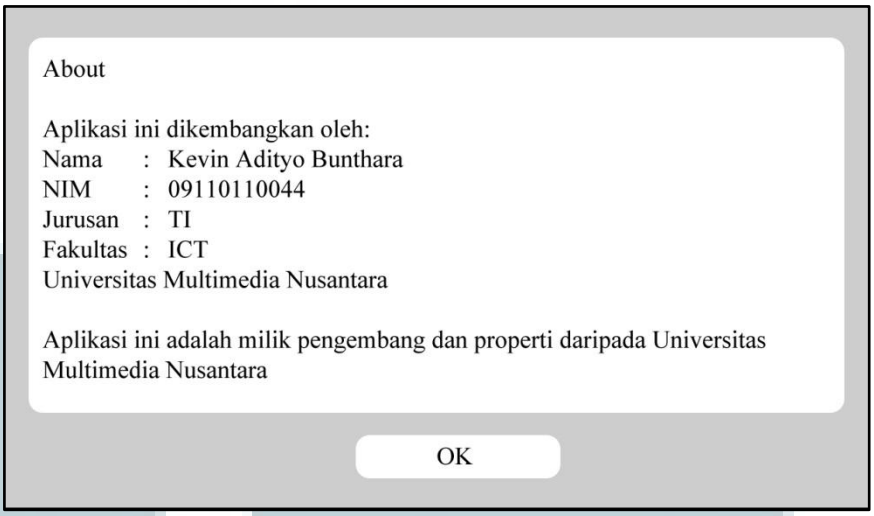
Kedua menu terakhir adalah menu *help* dan menu *about*. Pada saat menu *help* ditekan maka aplikasi akan mengeluarkan *dialog box* yang berisikan tata cara penggunaan aplikasi. Berikut adalah tampilan *dialog box* ketika menu help ditekan.



Gambar 3.3 Dialog box menu help

Menu about adalah menu yang berisikan informasi data pribadi pengembang aplikasi serta hak cipta penggunaan aplikasi. Pada saat menu about

ditekan maka aplikasi akan menampilkan sebuah dialog box seperti yang dapat dilihat pada gambar 3.4.



Gambar 3.4 Dialog box menu about