BAB III

METODE PENELITIAN

Untuk pengumpulan data yang diperlukan dalam melaksanakan tugas akhir, ada beberapa cara yang telah dilakukan, antara lain:

1. Studi kepustakaan

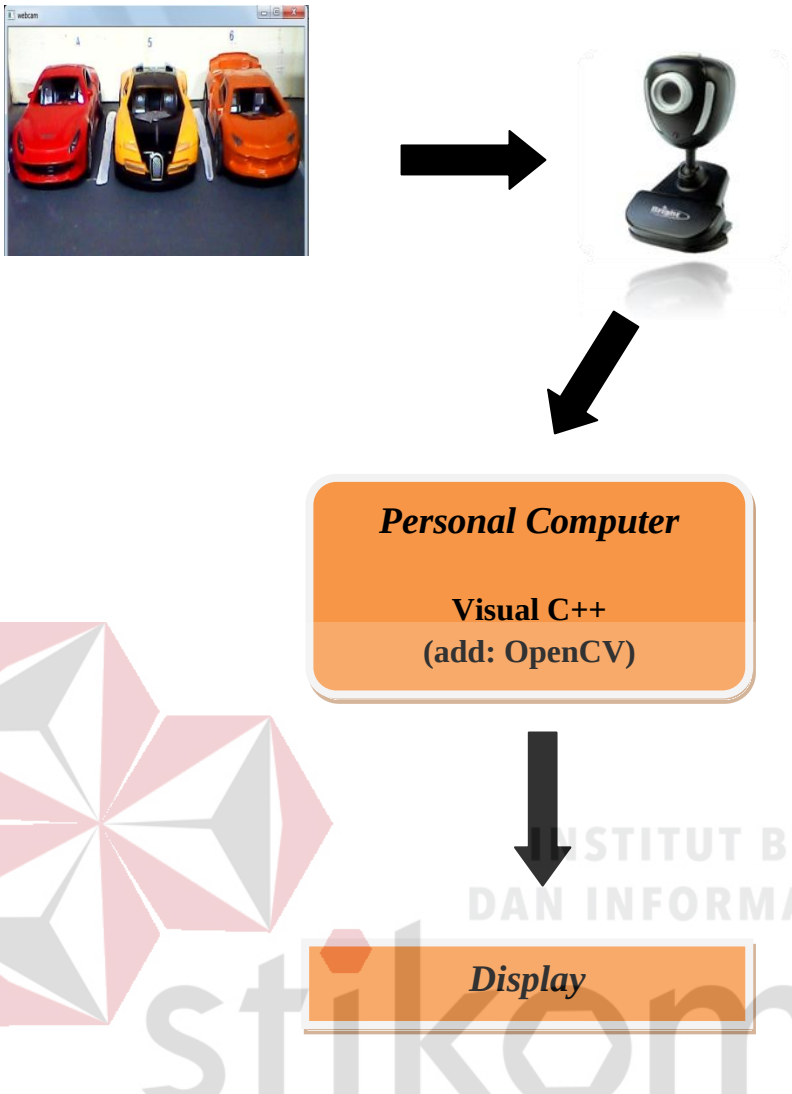
Studi kepustakaan berupa pencarian data-data literatur dari fungsi pada *library* OpenCV, melalui pencarian dari internet, dan konsep-konsep teoritis dari buku-buku penunjang serta metode yang akan digunakan untuk melakukan pengolahan citra.

2. Penelitian laboratorium

Penelitian laboratorium dilakukan dengan perancangan perangkat lunak,implementasi perangkat lunak, pengambilan data pengujian aplikasi, kemudian melakukan evaluasi dari data hasil pengujian.

3.1. Perancangan Sistem dan Blok Diagram Sistem

Model penelitian yang akan dilakukan adalah model penelitian pengembangan. Untuk mempermudah dalam memahami sistem yang akan dibuat dapat dijelaskan melalui blok diagram pada Gambar 3.1.



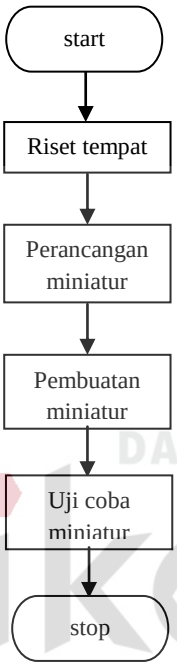
Gambar 3.1 Blok Diagram Sistem Secara Umum

Sebagai input, citra didapatkan dari kamera yang terpasang pada miniatur. Kemudian diproses menggunakan *console application* Visual C++ 2008 dengan memanfaatkan *library* OpenCV. Citra yang diperoleh adalah citra keadaan blok parkir mobil. Setelah mendapatkan citra tersebut, maka selanjutnya citra akan diproses untuk kemudian disubtraksi/dikurangkan (*subtraction*) dengan citra sampel (citra blok tanpa mobil). Dari hasil subtraksi tersebut dilakukan klasifikasi benda yang teridentifikasi merupakan mobil atau tidak. Pada akhirnya akan

ditampilkan informasi pada PC berupa *output* nomor tempat yang kosong (diasumsikan bahwa setiap tempat parkir terdapat nomor urut), selain itu juga citra yang diambil secara *streaming* akan ditampilkan pada layar PC.

3.2. Perancangan Perangkat Keras

Flowchart perancangan dan pembuatan miniatur sebagai berikut :



Gambar 3.2 *Flowchart* Pembuatan Miniatur

Perangkat keras pada Tugas Akhir ini berupa miniatur lantai 2 tempat parkir mobil di salah satu pusat perbelanjaan di Surabaya. Sebelum proses pembuatan miniatur telah dilakukan riset untuk mengetahui desain dan ukuran tempat parkir tersebut, selanjutnya dicari ukuran yang tepat untuk diterapkan dalam bentuk miniatur menggunakan skala perbandingan sehingga hasil program bisa mendekati hasil sesungguhnya apabila diterapkan secara *real*.

Dari hasil riset diperoleh data yaitu panjang lokasi parkir 50 m dengan tinggi 2,1 m dan lebar 9 m. Dari hasil tersebut ditentukan dimensi miniatur dan didapatkan skala perbandingan 1:25, sehingga panjang miniatur menjadi 2 m, tinggi 8,4 cm, dan lebar 36 cm. Hasil dimensi miniatur tersebut diperoleh dari penghitungan sebagai berikut:

Diketahui:

Panjang sesungguhnya (PS) : 50 m = 5000 cm

Tinggi sesungguhnya (TS) : 2,1 m = 210 cm

Lebar sesungguhnya (LS) : 9 m = 900 cm

Ketinggian Kamera sesungguhnya (KS) : 2,1 m = 210 cm

Jarak Kamera dari mobil sesungguhnya (JK) : 6 m = 600 cm

Skala 1:25

Dicari : Pm, Tm, Lm

Penyelesaian:

Panjang miniatur (Pm) = 5000/25 = 200 cm = 2 m

Tinggi miniatur (Tm) = 210/25 = 8,4 cm

Lebar miniatur (Lm) = 900/25 = 36 cm

Ketinggian kamera miniatur (Km) = 210/24 = 8,4 cm

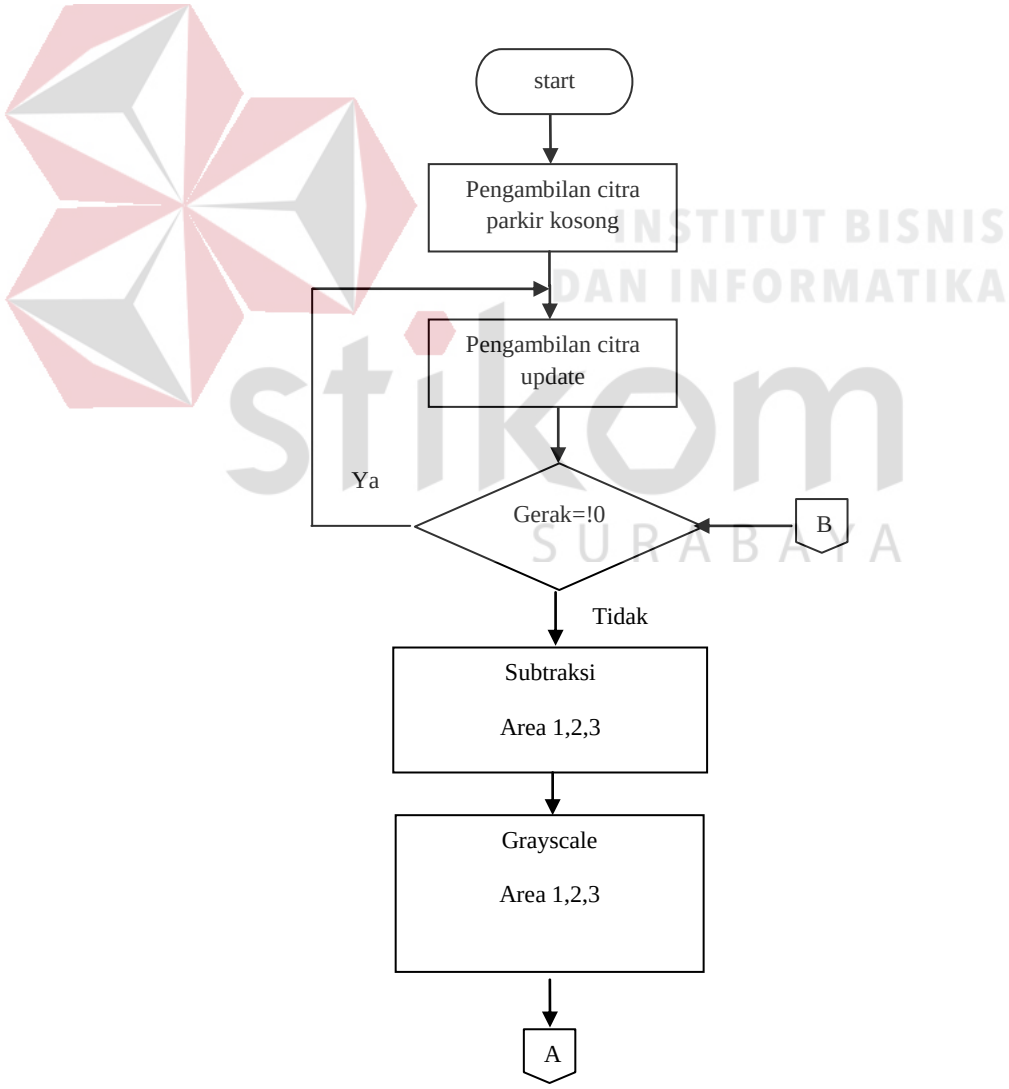
Jarak Kamera miniatur (Jm) = 600/25 = 24 cm

Karena webcam yang digunakan secara *default* sudah dalam kondisi diperbesar maka pada miniatur ditentukan jarak kamera menjadi 36 cm sehingga menghasilkan tampilan tiga slot tempat parkir yang tertangkap kamera.

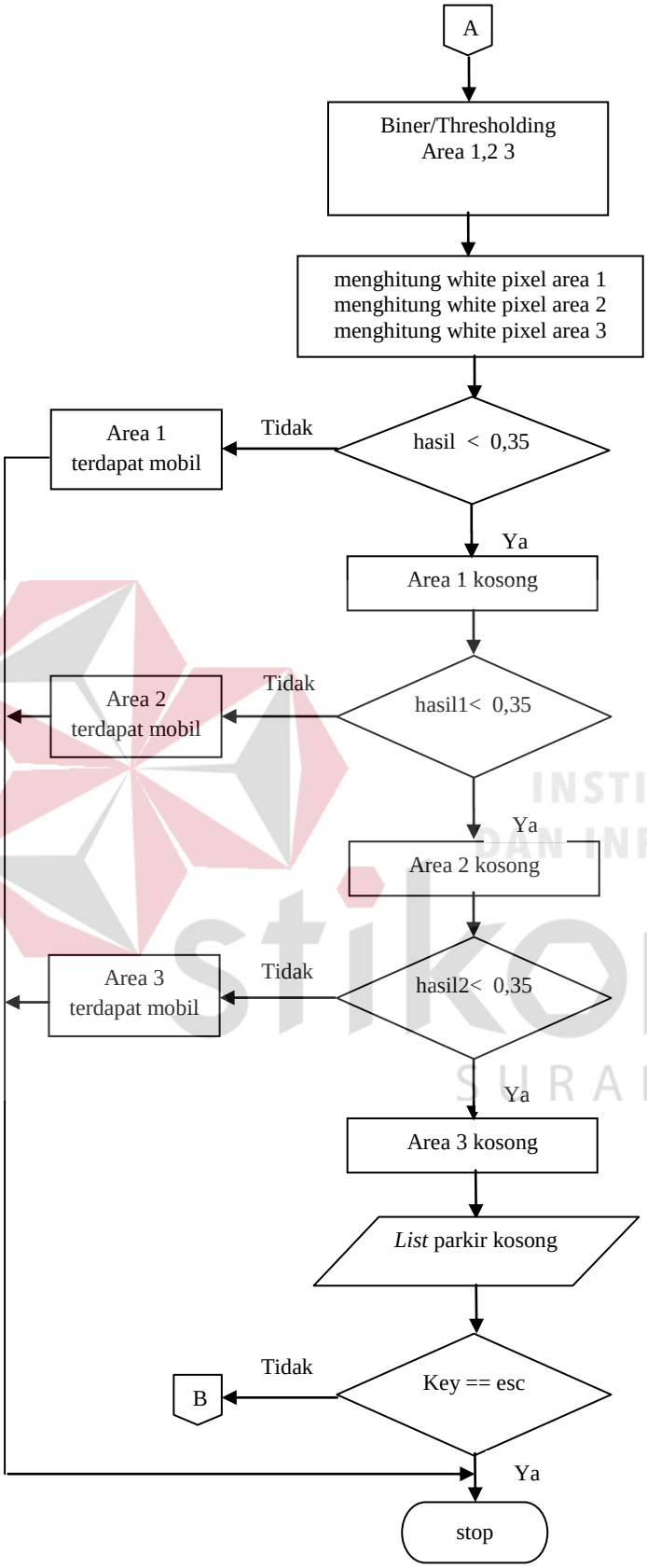
3.3. Perancangan Perangkat Lunak

Dalam perancangan perangkat lunak, *compiler* yang digunakan adalah Microsoft Visual C++ 2008. Untuk *library* yang digunakan pada pengolahan citra yaitu *library* OpenCV v2.3.

Kemudian dalam penulisannya atau dalam pembuatan program, akan meliputi bagian-bagian penting dalam setiap langkah-langkah per bagian sesuai dengan algoritma atau logika sekuensial dari awal sampai *output*. Berikut adalah algoritma program secara global.



Gambar3.3 *Flowchart* sistem secara global



Gambar3.4 *Flowchart* sistem secara global (lanjutan)

Gambar 3.3 dan Gambar 3.4 adalah *Flowchart* sistem secara global atau keseluruhan. Dimulai dari pengambilan citra sampel berupa kondisi parkir yang masih kosong. Proses berikutnya adalah kamera melakukan *streaming* dan mengambil/menyimpan citra kondisi parkir setiap 5 menit sekali (*update* data). Kemudian menentukan titik koordinat area yang menjadi acuan dalam proses pengolahan citra selanjutnya. Setelah itu mengambil dan mengakses nilai RGB per *pixel* dari citra sampel dan citra *update* (citra hasil *capture* setiap 5 menit sekali). Setelah didapatkan nilai RGB per *pixel* dari masing-masing citra dan selanjutnya nilai RGB dari citra sampel akan dikurangkan (*subtraction*) dengan nilai RGB dari citra *update*. Setelah didapatkan hasil pengurangan nilai RGB maka citra hasil pengurangan/subtraksi diubah dalam bentuk citra abu-abu atau biasa disebut *Grayscale*. Proses selanjutnya adalah citra yang sudah dalam bentuk *Grayscale* diubah menjadi bentuk citra binary melalui proses *thresholding* agar dapat dihitung jumlah *pixel* putih pada area tempat mobil diparkir, dari hasil penghitungan tersebut didapatkan nilai yang akan dianalisis dan dikategorikan ada atau tidaknya mobil di area tersebut dan proses yang terakhir adalah hasil analisis akan ditampilkan di *output* berupa *list* nomor parkir yang kosong.

3.4. Pengambilan Citra Sampel

Citra sampel yang dimaksud adalah citra kondisi parkir saat dalam keadaan tanpa adanya mobil. Citra sampel tersebut digunakan untuk data yang akan disubtraksi dengan citra *update*. Pengambilan citra sampel dilakukan secara manual dan akan disimpan pada direktori D:\\pict_TA\\dika dan file tersebut

diberi nama background.jpg. Ketika program berjalan program akan memuat citra sampel menggunakan fungsi *cvLoadImage* dan disimpan dalam variabel img, format variabel img adalah *Iplimage.* Berikut potongan program untuk memuat citra sampel .

```
IplImage*img=cvLoadImage(  "D:\\pict_TA\\dika\\background.jpg"
 );
```

3.5. Penerimaan Data Citra

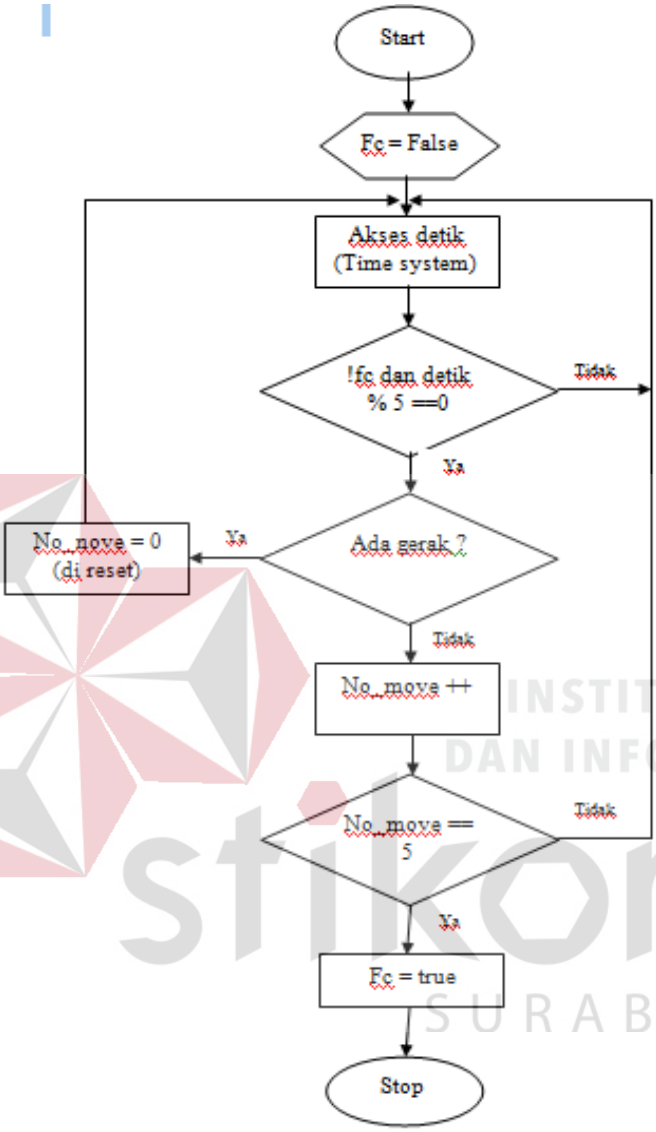
Setiap data citra yang dikirimkan dari kamera diakses dengan *pointer CvCapture* dan menggunakan fungsi *cvCaptureFromCAM(1).* Angka 0 pada fungsi *cvCaptureFromCAM(0)* merupakan indeks dari kamera yang digunakan. Berikut adalah potongan program untuk proses penerimaan data citra dari Kamera menggunakan.

```
CvCapture* capture = cvCaptureFromCAM(0);
```

Data citra yang ditangkap adalah data citra dengan ruang warna RGB dan disimpan langsung pada variabel Ipl*image* (*Intel Image Processing Library*) yaitu stuktur data untuk penyimpanan data citra pada OpenCV. Urutan *channel* data dalam *Iplimage* adalah BGR sehingga untuk menampilkan warna sesungguhnya.

3.6. Mendeteksi gerak

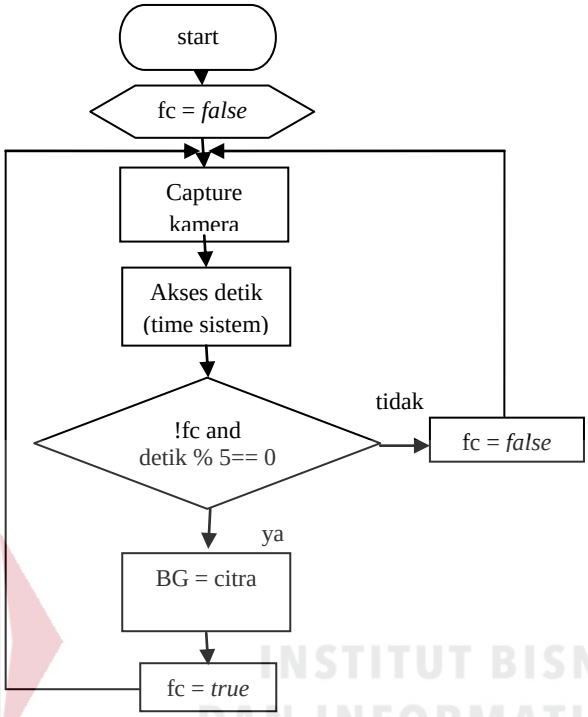
Flowchart proses mendeteksi gerak adalah sebagai berikut.



Gambar 3.5 *flowchart* mendeteksi gerak

3.7. Proses *Update* Citra Kondisi Parkir

Flowchart proses *update* data setiap 5 detik sekali adalah sebagai berikut.

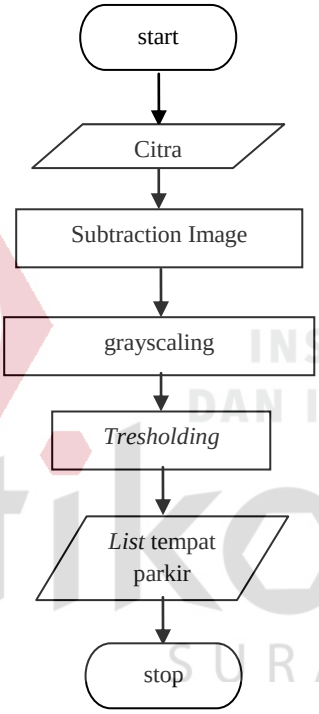


Gambar3.6 *Flowchart Update* Citra Kondisi Parkir (lanjutan)

Pada Gambar 3.5 dan Gambar 3.6 di atas, proses *streaming* dijalankan dan program juga mengakses detik yang terdapat pada *time system*, selain itu pada inisialisasi fc yang merupakan flag diberi nilai awal *false*. Ketika berada pada kondisi !fc dan menit mencapai 5 detik atau kelipatannya (nilai didapat dari detik dimodulus 5) maka program akan menyimpan hasil *capture* di variabel BG. Variabel BG adalah variabel dengan format *Iplimage* yang digunakan untuk menampung data citra hasil *update*. Proses selanjutnya variabel BG akan diolah sampai mendapatkan hasil yang diinginkan setelah itu nilai fc diberi nilai *true*. Tetapi jika waktu belum mencapai 5 detik maka program tidak akan mengeksekusi proses selanjutnya hanya menjalankan *streaming*.

3.8. Pengolahan Citra

Proses pengolahan citra adalah proses yang paling utama dalam pengerjaan program untuk sistem pada Tugas Akhir ini karena menggunakan Kamera sebagai sensor pendeteksinya. Berikut adalah *Flowchart* pengolahan citra secara garis besar :



Gambar3.7 *Flowchart* Pengolahan Citra

Metode utama yang digunakan pada proses pengolahan citra adalah metode *Background Subtraction image*. Untuk mendukung metode utama dilakukan juga proses pengolahan citra pendukung seperti konversi warna RGB menjadi bentuk abu-abu (*grayscaling*) dan konversi warna RGB menjadi hitam putih/biner (*thresholding*). Proses akses *pixel* juga merupakan proses yang penting untuk menunjang metode *Background Subtraction*. Untuk mempermudah proses

mendeteksi mobil maka pada *image* sampel maupun *image update* dibagi menjadi tiga area mobil (1 area berisi 3 mobil) dengan koordinat *pixel* yang telah ditentukan. Untuk proses pengolahan citra lebih detail lagi akan dijelaskan pada tiap-tiap sub bab.

3.8.1. Menentukan Koordinat Area

Proses menentukan koordinat masing-masing area dilakukan secara manual. Untuk mempermudah mencari koordinat yang tepat maka perlu menampilkan hasil koordinat yang dicoba dengan memberi tanda agar dapat terlihat posisi koordinatnya. Persegi adalah tanda yang akan digunakan untuk menandai masing-masing area pada program Tugas Akhir ini. Pada proses memberi tanda (*masking*) digunakan *function* bawaan *library* OpenCV yaitu.

*cvRectangle(CvArr*img,cvPoint(x,y),cvPoint(x1,y1),cvScalar(255,255,255),t,l).*

Variabel *CvArr*img* merupakan citra yang akan diberi gambar persegi, sedangkan untuk variabel *cvPoint(x,y)* berfungsi menentukan posisi titik sebelah kiri atas pada persegi yang akan dibuat dan fungsi *cvPoint(x1,y1)* adalah menentukan posisi titik sebelah kanan bawah pada persegi yang akan dibuat. *cvScalar(RGB)* berfungsi untuk menentukan warna persegi. Variabel *t* adalah variabel untuk mewakili nilai ketebalan garis yang diinginkan sedangkan variable *l* adalah untuk menentukan jenis garisnya.

Menentukan koordinat masing-masing area dilakukan agar dapat mengenali objek pada lokasi yang berbeda-beda dan dapat mengetahui area yang telah terisi mobil dan yang masih kosong.

3.8.2. Akses *Pixel*

Library OpenCV telah menyediakan function untuk *Background Subtraction* yaitu cvSub*(source1, source2, dest, NULL)* tetapi hasil subtraksi yang didapatkan tidak maksimal sehingga perlu dilakukan proses subtraksi secara manual. Dalam proses *Background Subtraction* secara manual langkah yang pertama adalah mengakses RGB per *pixel* pada citra sampel dan citra *update*. Berikut adalah potongan program untuk mengakses RGB per *pixel* dari citra sampel dan citra *update* :

```
int blue = ((uchar*)(img->imageData + img->widthStep*y))[x*3];
int green= ((uchar*)(img->imageData + img->widthStep*y))[x*3+1];
int red = ((uchar*)(img->imageData + img->widthStep*y))[x*3+2];
int b= ((uchar*)(BG->imageData + BG->widthStep*y))[x*3];
int g= ((uchar*)(BG->imageData + BG->widthStep*y))[x*3+1];
int r = ((uchar*)(BG->imageData + BG->widthStep*y))[x*3+2];
```

Variabel blue, green, dan red merupakan variabel untuk menampung nilai dari RGB per *pixel* pada citra sampel, sedangkan variabel b,g, dan r digunakan pada citra *update*. Variabel lainnya adalah img dan BG, img yaitu variabel yang menampung data citra sampel sedangkan variabel BG untuk citra *update*. Function `->imageData` berguna untuk mendapatkan data gambar, sedangkan function `->widthStep` berguna untuk memeriksa per *pixel*. Variabel x dan y adalah variabel yang digunakan untuk proses perulangan. Sedangkan untuk mengakses komposisi warna RGB pada *pixel* digunakan ketetapan `[x*3]` untuk warna biru, `[x*3+1]` untuk warna hijau, dan `[x*3+2]` untuk warna merah.

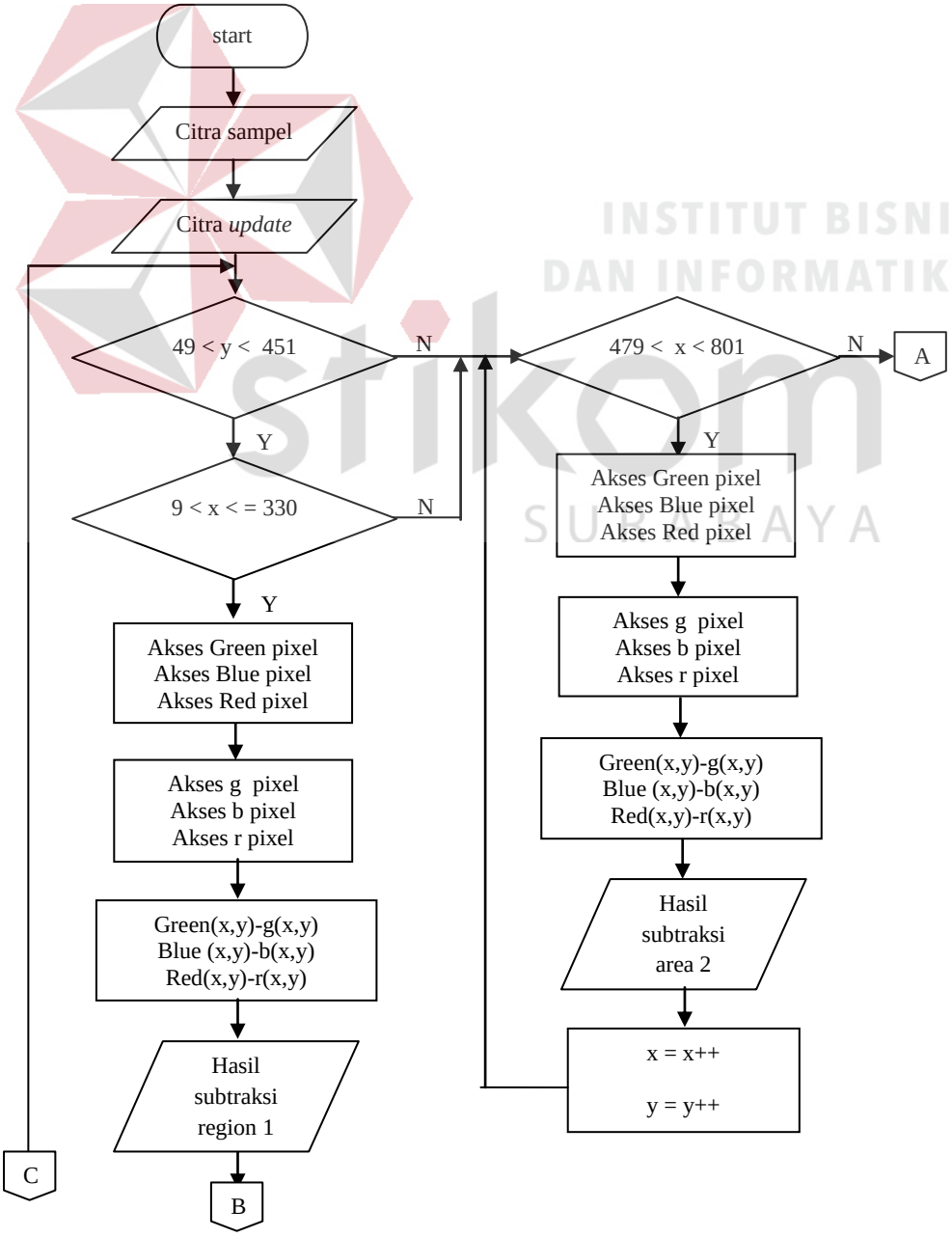
Pada saat sudah mendapatkan nilai RGB dari tiap-tiap *pixel* maka dilakukan proses subtraksi. Setelah proses subtraksi maka *pixel-pixel* hasil subtraksi tersebut harus dikembalikan ke bentuk semula yaitu dalam format *image* untuk diproses ke

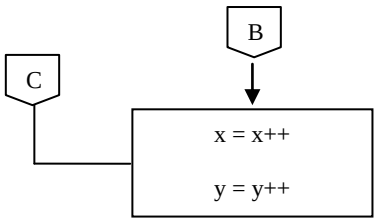
tahap selanjutnya. Berikut adalah potongan program untuk memasukkan *pixel* ke dalam bentuk *image*:

```
area-> imageData[region->widthStep*y+x*region-> nChannels+1]=sg;
area->imageData[region->widthStep*y+x*region-> nChannels+2]=sr;
area->imageData[region->widthStep*y+x*region-> nChannels+3]=sb;
```

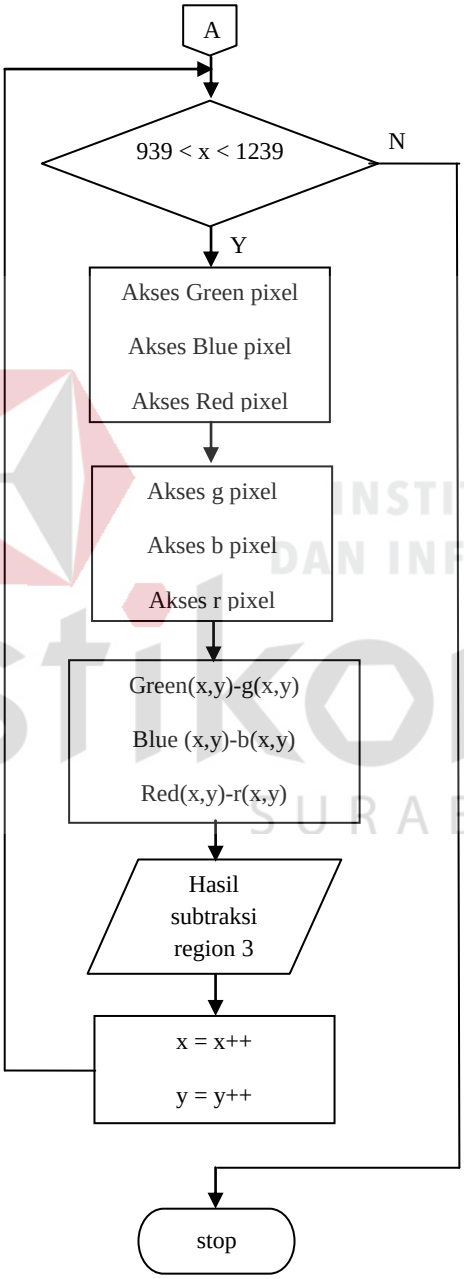
region adalah variabel berformat *image* yang digunakan untuk menampung hasil subtraksi. Sedangkan variabel sg,sr, dan sb adalah hasil subtraksi hijau, merah dan biru.

3.8.3. *Background Subtraction*





Gambar 3.8 *Flowchart Background Subtraction*



Gambar 3.9 *Flowchart Background Subtraction* (lanjutan)

Pada Gambar 3.8 dan Gambar 3.9 merupakan *Flowchart Background Subtraction*. Dalam *Flowchart* tersebut telah ditentukan koordinat *pixel* untuk masing-masing area/region dengan rincian berikut.

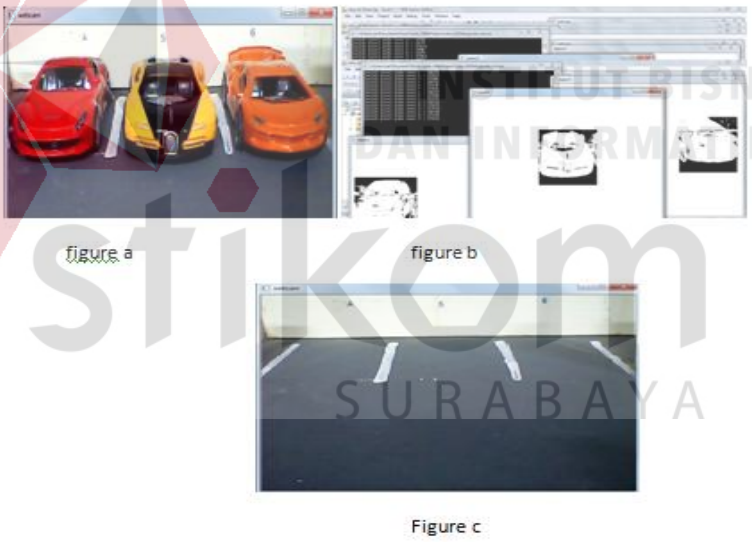
a. Area/region 1, berada diantara koordinat x,y(10<=x<=330 ,50<=y<=450)
b. Area/region 2, berada diantara koordinat x,y(480<=x<=800, 50<=y<=450)
c. Area/region 3, berada diantara koordinat x,y(940<=x<=1240, 50<=y<=450)

Pada awalnya *image* akan koordinat *pixel* gambar diperiksa satu persatu, apabila mendapatkan koordinat yang sama dengan koordinat area 1 maka proses selanjutnya adalah mengakses *pixel* RGB pada gambar sampel dan gambar *update*.

Pada *Flowchart pixel* RGB pada gambar sampel dinyatakan dengan Green *pixel*, Blue *pixel* dan Red *pixel*. Sedangkan pada gambar *update* dinyatakan dengan g *pixel*, b *pixel* dan r *pixel*. Setelah proses akses *pixel* maka *pixel-pixel* tersebut akan di subtraksi sesuai dengan koordinat dan jenis warnanya. Setelah didapatkan *pixel-pixel* hasil subtraksi maka langkah berikutnya adalah mengelompokkan kembali *pixel-pixel* hasil subtraksi dalam format *image* ke dalam variabel region1. Tetapi apabila koordinat *pixel* tidak sama dengan koordinat area 1 maka akan dilanjutkan dengan memeriksa apakah berada di koordinat area 2 atau area 3. Proses yang dijalankan apabila berada di koordinat area 2 atau area 3 sama dengan urutan proses yang berada di area 1. Region 2 adalah variabel berformat *image* untuk menyimpan hasil subtraksi area 2,

sedangkan region 3 adalah untuk menyimpan hasil subtraksi area 3.Jika koordinat yang di periksa tidak berada di koordinat area 1, area 2 dan area 3 maka program tidak memproses *pixel* tersebut.

Untuk melakukan proses *Background Subtraction*, maka harus didapatkan terlebih dahulu citra sampel untuk kemudian dijadikan acuan perbandingan dan pengurangan dengan citra baru yang ingin subtraksi. Dan *background subtraction* akan menemukan bagian citra baru berupa objek yang berbeda antara citra sampel dengan citra *update*. Berikut pada Gambar 3.10 adalah contoh *background subtraction*.



Gambar 3.10 Contoh *Background Subtraction*

Pada Gambar 3.10 terdapat citra dengan nama figure c yang merupakan citra sampel yang akan disubtraksi dengan citra *update* (figure a), pada figure a terdapat objek yang berbeda dari figure c yaitu terdapat mobil yang berada pada area tersebut sehingga ketika disubtraksi maka menghasilkan citra seperti pada figure b.

3.8.4. *Grayscaling*

Grayscale adalah suatu format citra atau gambar yang tiap-tiap *pixel* gambar hanya terdiri dari 1 *channel* warna. Proses perubahan warna dari RGB menjadi *Grayscale* bertujuan untuk mempermudah proses selanjutnya yaitu proses perubahan *Grayscale* menjadi biner. Sehingga setelah proses subtraksi berhasil dilakukan maka langkah selanjutnya ialah melakukan konversi format gambar dari RGB menjadi *Grayscale*. Untuk mengubah RGB menjadi *Grayscale* dapat digunakan rumus.

$Grayscale = 0.299R + 0.587G + 0.114B$

atau dapat menggunakan algoritma dengan merata-rata nilai ketiga buah *channel* RGB.

$Grayscale = (R + G + B) / 3$

Perubahan gambar RGB menjadi *Grayscale* menggunakan library openCV pada visual C++ menggunakan perintah sebagai berikut.

```
cvCvtColor(region1,gimask1,CV_RGB2GRAY);
```

Pada perintah tersebut sudah terdapat dua *frame* yaitu region1 dan gimask1. region1 adalah *frame* RGB hasil subtraksi sedangkan gimask1 adalah *frame* yang disediakan untuk *Grayscale* yang akan dibuat. Sehingga maksud dari potongan perintah tersebut adalah mengubah gambar region1 menjadi *Grayscale* dengan fungsi CV_RGB2GRAY lalu disimpan pada *frame* bernama gimask1.Tetapi terlebih dahulu dibuat deklarasi pointer untuk *image* graycale, yaitu dengan cara seperti berikut.

```
IplImage*gimask1=cvCreateImage(cvGetSize(region1),IPL_DEPTH_8U,1);
```

Pada potongan program tersebut terdapat IPL_DEPTH_8U yang artinya adalah tiap-tiap *pixel* bernilai 8 bit. Sedangkan angka 1 setelah koma dibelakang IPL_DEPTH_8U bermakna tiap-tiap *pixel* hanya terdiri dari sebuah *channel*.

3.8.5. *Thresholding*

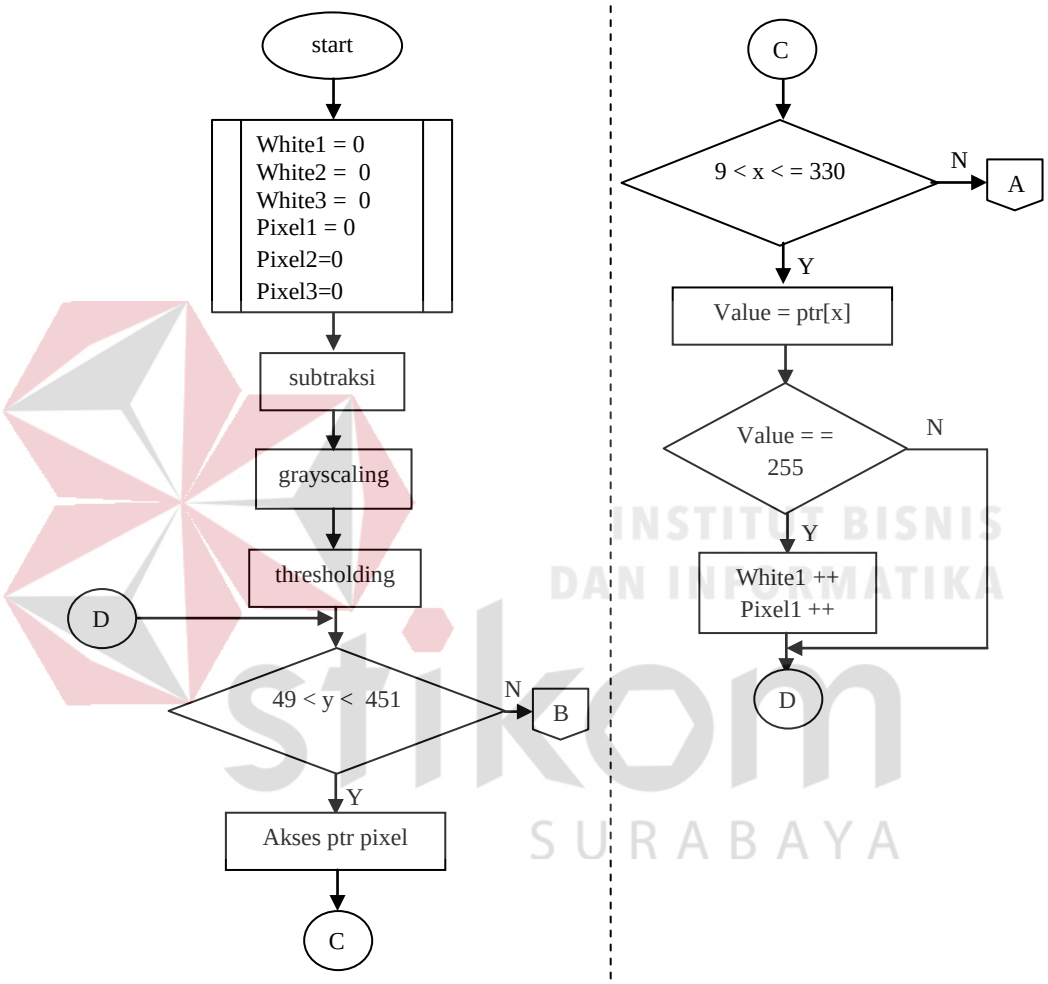
Untuk melakukan penghitungan *pixel* putih pada masing-masing region, maka data citra dikonversikan ke dalam citra biner dengan memanfaatkan *thresholding. Thresholding* adalah proses mengubah suatu citra berwarna atau berderajat keabuan (*Grayscale*) menjadi citra biner atau hitam putih, sehingga dapat diketahui daerah mana yang termasuk objek dan *background* dari citra secara jelas (Gonzales dan Woods, 2002). Citra hasil *thresholding* biasanya digunakan lebih lanjut untuk proses pengenalan obyek serta ekstraksi fitur. Tipe data dari hasil proses *thresholding* adalah tipe data *float*, yaitu antara 0 sampai dengan 1. Dengan parameter yang di set sebelumnya maka data citra yang jika melebihi batas yang ditentukan akan dibuat menjadi 1 atau putih dan jika dibawah batas yang ditentukan maka akan dibuat menjadi 0 atau hitam.

Namun pada *library* OpenCV telah disediakan *function* untuk memproses *thresholding*, yaitu dengan menggunakan `cvThreshold`. Berikut merupakan baris perintah *thresholding*.

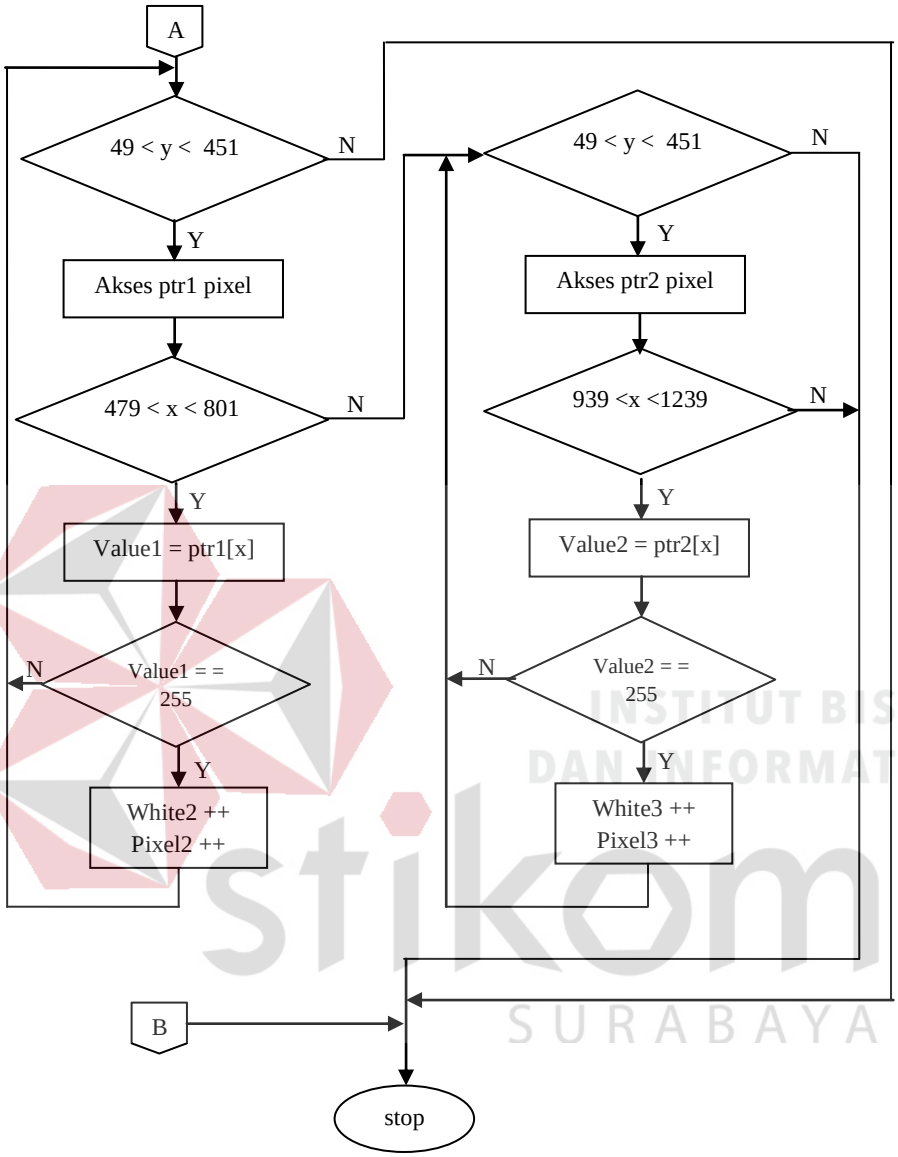
```
cvThreshold(gimask1, im_bw, 30, 255, CV_THRESH_BINARY);
```

Pada perintah tersebut gambar *Grayscale* dari *frame* gimask1 dirubah menjadi biner (CV_THRESH_BINARY) dengan *threshold* 40 dan disimpan pada *frame* im_bw. *Threshold* bertujuan mengubah *pixel* diatas *threshold* untuk menjadi *pixel* bernilai 255 sedangkan dibawah *threshold* dirubah menjadi 0, dengan demikian didapatkanlah gambar biner.

3.9. Penghitungan Piksel Putih (*Counting White Pixel*)



Gambar 3.11 *Flowchart Accounting White Pixel*



Gambar 3.12 *Flowchart Accounting White Pixel* (lanjutan)

Pada Gambar 3.11 dan Gambar 3.12 merupakan *Flowchart* proses untuk menghitung *pixel* warna putih pada 3 area dari gambar hasil subtraksi yang telah dibinerkan. Pada awalnya variabel *counter* (white1,white2, white3,pixel1,pixel2 dan pixel3) diberi nilai 0. Setelah melakukan proses subtraksi citra sampel dengan citra *update* dan telah dilakukan proses *Grayscale* serta *thresholding*, selanjutnya

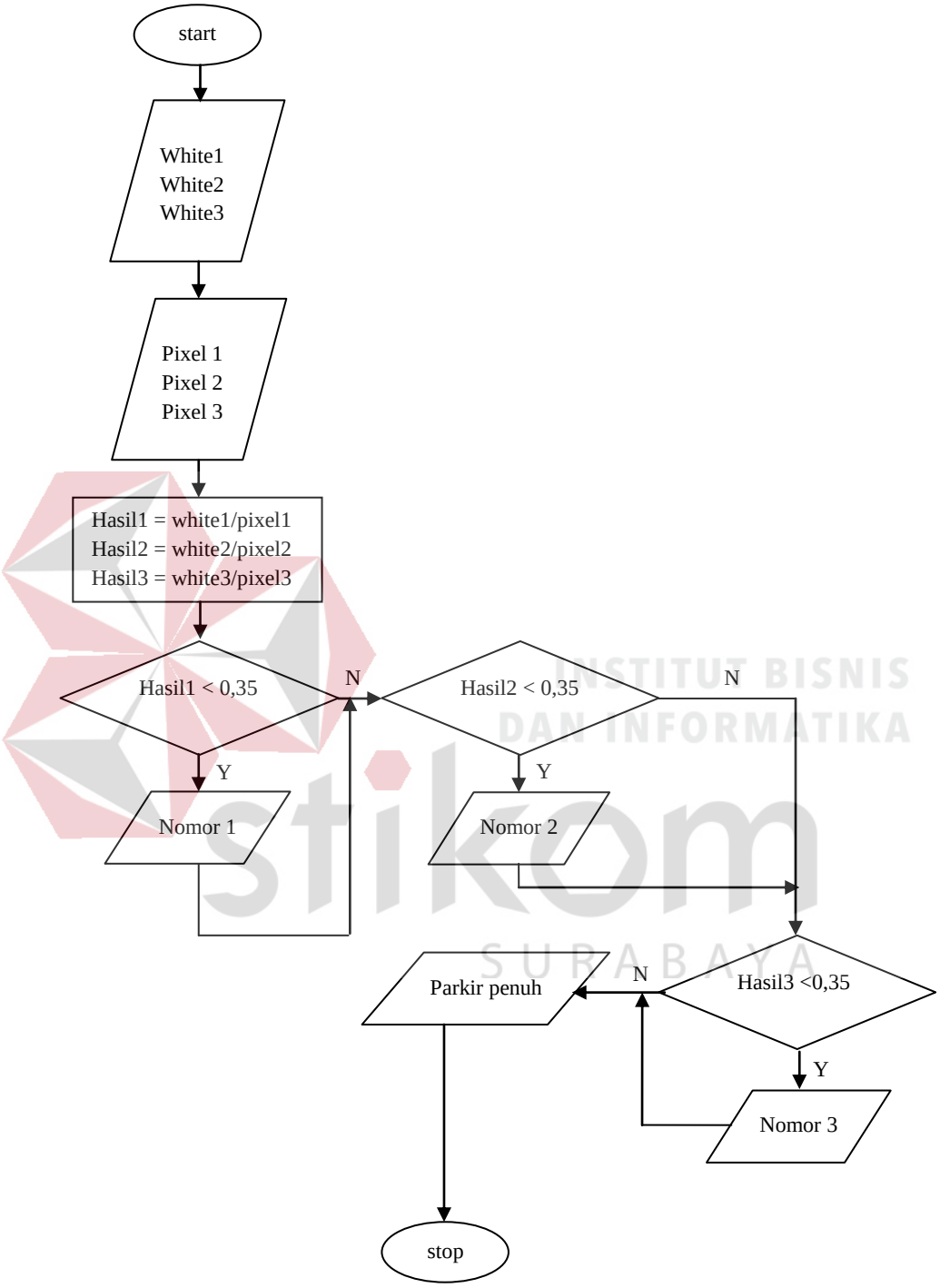
memeriksa koordinat *pixel*. Jika koordinat y tepat pada range koordinat y area 1 maka proses akan mengakses *pixel* dan nilainya dimasukkan pada variabel ptr. Selanjutnya jika koordinat x berada pada koordinat x area 1 maka nilai variabel ptr dimasukkan ke dalam variabel value untuk diperiksa apakah bernilai 255 (warna putih). Apabila value bernilai 255 maka nilai variabel *counter* white1 dan pixel1 ditambahkan 1. Tetapi apabila koordinat x tidak berada pada koordinat x area 1 maka koordinat akan diperiksa kembali untuk mengetahui apakah berada di area 2 atau area 3 atau tidak berada di koordinat 3 area tersebut. Untuk proses penghitungan *pixel* di area 2 maupun 3 sama seperti halnya di area 1. Sedangkan jika diluar koordinat 3 area tersebut maka tidak akan diproses lebih lanjut. Berikut adalah potongan program untuk menghitung *pixel* pada salah satu area.

```
for ( y = 50 ; y <= 450 ; y++)
{
      uchar* ptr = (uchar*)(im_bw->imageData+im_bw->widthStep*y );

    for ( x = 10 ; x <= 330 ; x++)
      {
            value = ptr[x];

            if (value == 255)
                  white1++;
                  pixel1++;
      }

}
```

Pada potongan program tersebut terdapat variabel im_bw yaitu variabel yang menampung *frame* biner hasil subtraksi area 1.

Penghitungan *pixel* sangat penting dilakukan untuk proses selanjutnya yaitu identifikasi benda. Karena identifikasi benda dilakukan dengan menghitung prosentasi *pixel* putih pada masing-masing area.

3.10. Identifikasi Benda



Gambar3.13 *Flowchart* Identifikasi benda

Pada *Flowchart* Gambar 3.13 untuk menentukan benda yang teridentifikasi merupakan mobil atau tidak maka terlebih dahulu total *pixel* putih (white1 untuk area 1, dst) dibagi dengan total keseluruhan *pixel* (pixel1 untuk area 1, dst) dari masing-masing area sehingga didapatkan prosentasi *pixel* putih pada tiap-tiap area. Kemudian nilai dari hasil bagi masing-masing area tersebut akan ditampung di variabel Hasil1 (untuk area 1), Hasil2 (untuk area 2) dan Hasil3 (untuk area 3). Setelah ditampung ke dalam variabel maka hasil akan dibandingkan dengan batas minimum prosentasi yang telah ditentukan yaitu 35% atau 0,35.

Jika prosentasi *pixel* putih pada area 1 kurang dari 0,35 maka area tersebut dianggap tidak ada mobil dan pada *list* nomor parkir akan ditampilkan nomor 1 untuk menginformasikan kepada pengguna parkir bahwa tempat parkir nomor 1 dapat ditempati. Jika prosentasi *pixel* putih lebih dari 0,35 maka area tersebut dianggap terdapat mobil dan akan dilanjutkan untuk mengeksekusi proses selanjutnya yaitu mengidentifikasi pada area yang lain. *List* parkir akan menginformasikan bahwa parkir telah penuh apabila prosentasi *pixel* putih pada semua area parkir melebihi 0,35. Berikut adalah potongan program untuk identifikasi benda.

```
if ( hasil1 < 0.35)
    cout << " 1 " << endl;
if ( hasil2 < 0.35)
    cout << " 2 " << endl;
if ( hasil3 < 0.35)
   cout << " 3 " << endl;
if ( hasil1 > 0.35 && hasil2 > 0.38 && hasil3 > 0.35)
   cout << " parkir penuh " << endl;
```

3.11. Metode Pengujian dan Evaluasi Sistem

Untuk mengetahui apakah aplikasi yang dibuat dapat berjalan sesuai yang diharapkan, maka akan dilakukan pengujian dan evaluasi sistem untuk setiap tahapan-tahapan dalam pembuatan aplikasi. Dimulai dari *streaming* citra, *update* citra kondisi parkir, menentukan koordinat pada tiap-tiap area, akses *pixel*, *Backgorund Subtraction*, *thresholding*, penghitungan *pixel* putih dan identifikasi benda.

3.11.1. Pengujian *Streaming* Citra

Untuk mengetahui apakah data citra sudah dapat diakses langsung melalui Kamera, maka dilakukan pengujian dengan cara menjalankan (*running*) program pemanggilan kamera dari Visual C++ 2008, yaitu untuk mengakses *console* Kamera secara langsung dari program. Kemudian citra yang tampil akan diuji apakah dapat menampilkan data citra secara *streaming*.

3.11.2. Pengujian *Update* Citra Kondisi Parkir

Untuk pengujian apakah program dapat melakukan *update* data berupa citra kondisi parkir secara otomatis maka akan ditampilkan waktu sistem. Hal ini dilakukan untuk mengetahui waktunya. Selain menampilkan waktu sistem, juga dilakukan cara lain yaitu membuka direktori lokasi tempat hasil *update* akan disimpan (D:\\Pict_TA\\dika\\update.jpg) yang bertujuan untuk mengetahui apakah gambar yang dimaksud akan berubah dan disimpan setiap 5 detik pada direktori tersebut.

3.11.3. Pengujian Penentuan Koordinat Pada Tiap Area

Untuk mengetahui posisi koordinat acuan pada masing-masing area maka program akan menampilkan *window image* berisi citra yang terdapat tanda persegi pada masing-masing area yang berfungsi untuk memberi batas area yang akan diproses datanya.

3.11.4. Pengujian Akses *Pixel*

Untuk mengetahui apakah program berhasil mengakses RGB dari tiap-tiap *pixel* maka akan ditampilkan *window image* yang menyimpan hasil akses ***pixel*** merah, hijau dan biru. Selain itu untuk mendukung pengujian maka output nilai pixel RGB juga ditampilkan.

.

3.11.5. Pengujian *Background Subtraction*

Untuk pengujian metode *Background Subtraction*, dilakukan dengan mengurangkan data citra sampel dengan citra *update*. Hasil subtraksi akan disimpan pada *frame*. Untuk mengetahui apakah program berhasil melakukan subtraksi maka akan ditampilkan 3 *window image* hasil subtraksi (hasil subtraksi area 1, area 2 dan area 3).

3.11.6. Pengujian *Thresholding*

Untuk pengujian proses *thresholding* maka ditampilkan 3 *window image* dari hasil subtraksi yang telah diubah menjadi gambar grayscale dan gambar

biner. 3 *window image* tersebut meliputi *window image* untuk area 1, area 2 dan area 3.

3.11.7. Pengujian Penghitungan *Pixel*

Untuk pengujian proses penghitungan *pixel* maka ditampilkan *output* hasil perhitungan sehingga dapat diketahui total *pixel* putih pada masing-masing area.

3.11.8. Pengujian Identifikasi Benda

Untuk pengujian proses identifikasi benda pada masing-masing area maka ditampilkan *output* berupa hasil bagi *pixel* putih dengan total *pixel* keseluruhan pada area. Selain itu juga ditampilkan *output* berupa *list* nomor tempat parkir yang masih kosong.

3.11.9. Evaluasi Sistem Keseluruhan

Setelah melalui seluruh proses pengujian di atas maka perlu dilakukan pengujian sistem secara keseluruhan. Dimulai dari melihat data citra yang ditangkap oleh Kamera, dan melihat tampilan data citra yang ditampilkan *window image.* Setelah itu, melalui tahap *update* citra, yaitu ketika waktu sistem menunjukkan detik ke-5 atau kelipatannya maka gambar yang tersimpan pada direktori D://Pict_TA//update.jpg akan berubah sesuai kondisi citra pada saat menit ke-5 atau kelipatannya dan apabila waktu belum mencapai 5 detik atau kelipatannya maka gambar yang tersimpan tidak akan berubah. Kemudian dilanjutkan dengan melihat hasil tahap pengolahan citra, yaitu ketika tiap 5 detik sekali, maka program akan memperbarui *list* nomor parkir. Kemudian sebagai

tambahan, kamera pada miniatur juga mengirimkan citra yang disorot untuk ditampilkan pada PC secara *streaming*. Jika keseluruhan sistem telah berjalan sesuai dengan langkah-langkah tersebut, maka secara keseluruhan sistem ini sudah dikatakan baik.