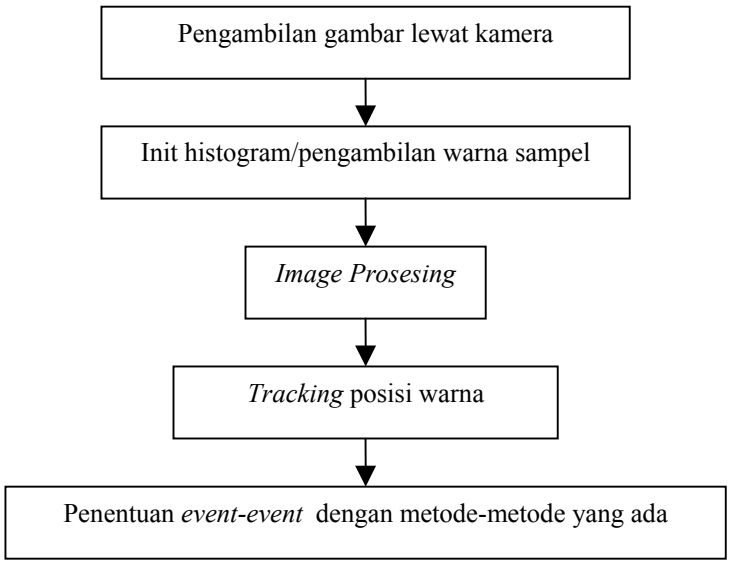
3. PERENCANAAN SISTEM

Pada bab ini yang dibahas adalah bagaimana cara pembuatan software secara umum. Mulai dari bagaimana cara mengambil sampel warna dan menjadikannya panduan untuk mencari dan menandai daerah pada obyek yang memiliki histogram warna. Juga bagaimana proses pembuatan tiap metode dilakukan. Mulai dari metode ROI, metode *tracking* dan metode *tracking* berlanjut.

3.1. Perencanaan Perangkat Lunak Secara Umum

Tugas akhir kali ini memiliki bagian-bagian atau diagram sebagai berikut:



Gambar 3. 1 Blok Diagram Program Secara Umum

Dari gambar 3.1 diatas terlihat bahwa hanya terdapat beberapa langkah saja. Sebenarnya masih ada langkah-langkah berikutnya, namun langkah-langkah ini tergantung dari metode apa yang akan dipakai dalam menentukan *event-event* pada mouse.

Untuk mengambil *event-event* pada mouse, telah dijelaskan bawa ada lima metode yang mungkin dapat digunakan yaitu:

1. Untuk *event-event* pergerakan, digunakan:
   a. metode seperti pada *joystick* (ROI).
   b. metode seperti pada mouse notepad (*tracking* berlanjut).
   c. metode touch screen (*tracking*).
2. Untuk *event-event* khusus, digunakan metode ROI/segmentasi.

Metode-metode ini dapat digabungkan satu dengan lainnya dan juga dapat ditambah dengan metode-metode lainnya yang mungkin lebih baik lagi, semua tergantung dari kegunaan dan kemudahan pemakai.

Untuk tugas akhir kali ini, akan dipakai penggabungan antara:

1. Metode ROI untuk pergerakan dan juga untuk *event* khusus.
2. Metode *tracking* berlanjut untuk pergerakan dan metode ROI untuk *event* khusus.
3. Metode *tracking* untuk pergerakan dan metode ROI untuk *event* khusus.

3.2. Pengambilan Gambar Oleh Kamera

Pengambilan gambar oleh kamera dilakukan oleh program yang telah ada yaitu camcap 1.3 (lihat gambar 3.2). Camcap 1.3 memakai teknologi direxT draw pada direct x untuk mengambil gambar pada kamera. Kemudian gambar tersebut diolah pada prosedure CallBackCam1 [*void CCamCapDlg::CallBackCam1(IplImage *frame)*]

Camcap 1.3 dapat dapat mengambil gambar dari 2 kamera sekaligus, juga dapat menjalankan file-file movie standart. Untuk menangani kamera 1 dipakai prosedure CallBackCam1 sedangkan untuk menangani kamera 2 dipakai procedure CallBackCam2 [*void CCamCapDlg::CallBackCam2(IplImage *frame)*]. Pada tugas akhir kali ini hanya dipakai kamera 1 saja.

Gambar yang telah ditangkap oleh kamera, dilewatkan ke dalam prosedure CallBackCam1 melalui sebuah variabel bernama frame [*void CCamCapDlg::CallBackCam2(IplImage *frame)*]. Variabel dan prosedure inilah yang akan kita pakai sebagai input untuk diolah lebih lanjut.

Gambar 3. 2 Camcap 1.3

3.3. Init Histogram/Pengambilan Warna Sampel

Awal dari pengolahan data adalah dengan mengambil warna dari tangan/obyek lain yang akan di-*tracking*. Untuk mengambil sample warna dari obyek yang akan di-*tracking* maka pada layar dibuat sebuah ROI dengan ukuran 40x40 pixel (lihat gambar3.3), bila pemakai menekan tombol init histogram (lihat gambar 3.4) yang telah disediakan maka gambar sampel pada ROI itu akan dipindahkan ke suatu variabel yang kemudian dari system warna berbentuk RGB (karena gambar dari kamera mempunyai system warna RGB) diubah ke system warna HSV [*iplRGB2HSV(frame, dest_maze_hsv)*]. Setelah itu image itu dipisahkan menurut komponen HSV-nya dan hasilnya disimpan dalam 3 variabel yang berbeda [*cvCvtPixToPlane(dest_maze_hsv, dest_maze_h, dest_maze_s, dest_maze_v, NULL)*].

Gambar 3. 3 ROI sampel warna

Gambar 3. 4 Tombol Init Histogram

Dari ketiga variabel itu dicari histogramnya masing-masing dan disimpan dalam bentuk array. Histogram ini kemudian disimpan terus untuk dipakai lebih lanjut. Histogram yang dibuat ada 3 yaitu histogram untuk nilai hue, saturation dan value, tetapi yang dimanfaatkan hanya histogram untuk hue saja karena informasi warna tersimpan disitu. Sedangkan histogram untuk saturation dan value akan dibuang.

Pembuangan histogram untuk saturation dan value ini disebabkan karena untuk *tracking* warna hanya akan memakai informasi dari nilai warnanya saja sedangkan nilai saturation dan value bila turut dihitung akan menyebabkan kekacauan yang disebabkan karena perbedaan pencahayaan dan sebagainya. Jadi histogram yang dipakai hanyalah histogram untuk hue saja.

Cara mencari histogram adalah dengan perintah:

[*//hitung nilai histogram dari sampel warna src_warna_h(hue)*
float thresh_h[1][2] = { {0, 255} };
float pthresh_h[1] = { thresh_h[0] };*
hist_h = cvCreateHist (1, histh_dim, CV_HIST_ARRAY, 0, 1);
cvSetHistBinRanges(hist_h, pthresh_h, 1);
cvCalcHist(&src_warna_h, hist_h, 0, 0);
//hitung nilai histogram dari sampel warna src_warna_s(saturation)
float thresh_s[1][2]= { {0, 255} };

float pthresh_s[1] = { thresh_s[0] };*

hist_s = cvCreateHist (1, hists_dim, CV_HIST_ARRAY, 0, 1);

cvSetHistBinRanges(hist_s, pthresh_s, 1);

cvCalcHist(&src_warna_s, hist_s, 0, 0);

//hitung nilai histogram dari sampel warna src_warna_v(value)

float thresh_v[1][2] = { {0, 255} };

float pthresh_v[1] = { thresh_v[0] };*

hist_v = cvCreateHist (1, histv_dim, CV_HIST_ARRAY, 0, 1);

cvSetHistBinRanges(hist_v, pthresh_v, 1);

cvCalcHist(&src_warna_v, hist_v, 0, 0);]

3.4. Image Prosesing

Yang dimaksud dengan image prosesing disini adalah diprosesnya image input dari kamera untuk proses *tracking*. Image prosesing yang dilakukan disini meliputi:

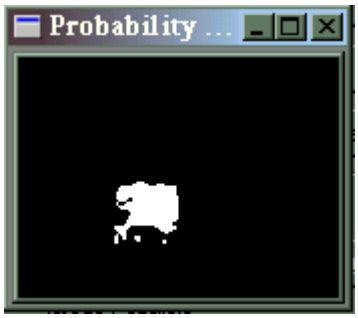
1. *Converting* dari RGB ke HSV.
2. Pemisahan tiap-tiap komponen image HSV.
3. Pencocokan tiap pixel komponen hue image yang menjadi input dengan histogram hue yang sesuai diberi nilai 255 sedangkan yang tidak diberi nilai 1. Hasilnya berupa image hitam putih, dimana yang sesuai dengan histogram hue diberi warna putih (255) sedangkan yang tidak diberi warna hitam (0). Perintah yang dipakai adalah: [*cvCalcBackProject(&dest_maze_h, img_calback_h, hist_h)*].
4. Pixel-pixel pada bagian saturation dan value di-*treshold*, yang nilainya dibawah 90 di-*blok*.

   [*cvThreshold(dest_maze_s,dest_maze_s,90,255,CV_THRESH_BINARY)*

   cvThreshold(dest_maze_v,dest_maze_v,90,255,CV_THRESH_BINARY)]. Hal ini disebabkan karena nilai saturation dan value yang terlalu rendah memang tidak diperlukan karena semakin mendekati ke arah hitam (untuk value/value dan saturation) dan putih (untuk saturation saja). Hal ini bisa kita ketahui bila kita mempelajari sistem warna HSV. Semakin rendah nilai value atau saturationnya maka semakin sedikit informasi warna yang dikandungnya

sehingga pada nilai tertentu dapat diabaikan. Hal inilah yang menyebabkan kita men-*treshold* nilai saturation dan value yang terlalu rendah untuk memudahkan proses komputasi.

5. Setelah semuanya dilakukan yang masih harus dilakukan adalah menggabungkan kembali hasil-hasil komputasi ketiganya menjadi satu. Hal ini dilakukan dengan cara ketiganya di-*AND*-kan menjadi satu. [*iplAnd(dest_maze_s, dest_maze_v, img_calback_and_sv)*
iplAnd(img_calback_h, img_calback_and_sv, temp)].
6. Setelah digabungkan menjadi satu, image tunggal hasil penggabungan itu kemudian di-*filter* dengan *median filter* untuk mengurangi *error*-nya. [*iplMedianFilter(temp1, temp, 3, 3, 1, 1)*].
7. Agar arah gerakan antara obyek, kamera dan layar maka perlu adanya *mirroring*. *Mirroring* dapat dilakukan sesuai dengan sumbu vertikal maupun horisontal sebagai sumbunya. [*iplMirror(temp,temp,1)*].



Gambar 3. 5 Hasil *Image Prosesing* pada Variabel Temp

8. Setelah semua langkah diatas dilakukan maka hasilnya disimpan (dalam variabel temp) dan ditampilkan pada window (gambar 3.5). [*cvvShowImage("Probability Image",display)*]. Hasil yang disimpan dalam variabel itu kemudian siap untuk diolah lebih lanjut sesuai dengan metode pengolahan *event* yang sesuai.

3.5. *Tracking* Posisi Warna

Tracking posisi warna seperti yang telah ditulis dalam bab 2 dilakukan dengan mencari titik berat dari suatu daerah. Hal ini dilakukan dengan memakai rumus seperti pada bab 2.

$$Wx = \frac{\sum_{x=0}^{k} x \times p_x}{jp}$$

$$Wy = \frac{\sum_{y=0}^{b} y \times p_y}{jp}$$

dimana Wx, Wy = koordinat x,y titik berat

x,y = koordinat x,y yang sedang dihitung

p_x, p_y = jumlah pixel pada x,y

jp = jumlah pixel pada image

Hasil image yang disimpan pada variabel temp itu sebenarnya adalah sebuah image binary yang hanya berisi nilai 0 (0/hitam) dan 1 (255/putih). 0 untuk warna latar sedangkan nilai 1 untuk warna benda hasil dari *tracking*.

Berdasarkan pada rumus, maka seharusnya yang dilakukan adalah:

1. Mencari jumlah seluruh pixel yang berwarna putih pada image (*jp*). [*valws = cvCountNonZero (temp)*]
2. Bila pada image itu kita buat koordinat cartiseius, maka bagian bawah image adalah sumbu x dan bagian kiri image adalah sumbu y.
3. Setiap pixel pada garis *x=n* dicari jumlah pixel yang berwarna putih. Kemudian di kalikan dengan *x*. hasilnya kemudian dibagi dengan *jp* dan didapatlah *Wx.*

```
[xs=1; //lebar ROI
ys=120; //tinggi ROI
jx=0; //tempat menyimpan hasil penjumlahan x*px
cy=0; //input koordinat cy
for (cx=0;cx<(160-xs);cx=cx+xs)
{
   size_roi_sensorl = cvRect(cx,cy,xs,ys); //kotak ROI
   size_image_rois = cvSize(xs,ys); //ukuran kotak ROI
//persiapan /image penyimpan roi
  semen = cvCreateImage(size_image_rois,IPL_DEPTH_8U,1);
   iplMirror(semen, semen, 0);
   semen->origin = IPL_ORIGIN_BL;
```

```
//ROI diambil dari image yang di-tracking
   cvSetImageROI(temp, size_roi_sensorl);
   cvCopyImage(temp,semen);
//hitung jumlah titik putih yang merupakan obyek
   valw = cvCountNonZero (semen);
   cvReleaseImage(&semen);
//hitung x*px dan tambahkan ke nilai sebelumnya
   jx=jx+cx*valw;
}
//hasil akhir koordinat x
cx=jx/valws;]
```

4. Setiap pixel pada garis *y=n* dicari jumlah pixel yang berwarna putih. Kemudian di kalikan dengan *y*. hasilnya kemudian dibagi dengan *jp* dan didapatlah *Wy.*

```
[xs=160; //lebar ROI
ys=1; //tinggi ROI
jy=0; //tempat menyimpan hasil penjumlahan y*px
cx=0;
for (cy=0;cy<(120-ys);cy=cy+ys)
{
   size_roi_sensorl = cvRect(cx,cy,xs,ys); //kotak ROI
   size_image_rois = cvSize(xs,ys); //ukuran kotak ROI
//persiapan /image penyimpan roi
  semen=cvCreateImage(size_image_rois,IPL_DEPTH_8U,1);
   iplMirror(semen, semen, 0);
   semen->origin = IPL_ORIGIN_BL;
//ROI diambil dari image yang di-tracking
   cvSetImageROI(temp, size_roi_sensorl);
   cvCopyImage(temp,semen);
//hitung jumlah titik putih yang merupakan obyek
   valw = cvCountNonZero (semen);
   cvReleaseImage(&semen);
```

```
    jy=jy+cy*valw;
}
cy=jy/valws;]
```

Hasil dari perhitungan titik berat seperti yang tampak pada potongan program diatas disimpan pada variabel cx dan cy.

3.6. *Event-Event* Pada Mouse

Event-event pada mouse dapat dibedakan menjadi 2 yaitu *event-event* pergerakan dan *event-event* khusus. Untuk membangkitkan *event-event* pada mouse sangatlah mudah karena windows telah menyediakan perintah khusus untuk melakukannya. Perintah ini merupakan salah satu perintah dalam windows API yang telah dijadikan salah satu perintah standart dalam visual c++. Demikian juga dengan perintah untuk membangkitkan *event-event* khusus untuk mouse juga telah disediakan oleh windows API namun telah juga dijadikan salah satu perintah standart oleh c++.

Untuk *event-event* pergerakan pada mouse perintah yang dipakai adalah [*SetCursorPos(sx,sy);*] dimana sx adalah koordinat x dari mouse sedangkan sy adalah koordinat y-nya. Titik pusat (0,0) pada layar terletak di pojok kiri atas. Jadi misalnya kita hendak menggerakkan kursor mouse ke kanan sejauh lima pixel maka kita hanya perlu menambahkan 5 pada nilai untuk koordinat x [*SetCursorPos(sx+5,sy);*] maka secara otomatis kursor mouse akan bergeser ke kanan sejauh 5 pixel. Sedangkan untuk menggerakkan kursor ke kiri 5 pixel kita hanya perlu mengurangi sx sebanyak 5. Hal ini berlaku juga untuk pergerakan vertikal, untuk menggerakkan naik kita harus mengurangi sedangkan untuk menggerakkan kursor ke bawah kita harus menambah.

Untuk *event-event* khusus, yaitu untuk *klik* kiri , *klik* kanan, *drag and drop*, serta *double klik* sebenarnya hanya memerlukan 1 perintah dan 4 buah konstanta sebagai parameter yang mengatur *event* apa yang harus dibangkitkan. Perintahnya yang telah disediakan oleh windows adalah [*mouse_eventt(konstanta,sx,sy,0,0);*]. Sx dan sy adalah koordinat dari kursor mouse tempat terjadinya *event* itu, sedangkan konstanta adalah *event* yang terjadi. Konstanta yang dapat diisikan adalah:

1. [*MOUSEEVENTTF_LEFTDOWN = &H2*] // tombol kiri ditekan
2. [*MOUSEEVENTTF_LEFTUP = &H4*] // tombol kiri dilepas
3. [*MOUSEEVENTTF_RIGHTDOWN = &H8*] // tombol kanan ditekan
4. [*MOUSEEVENTTF_RIGHTUP = &H10*] // tombol kanan dilepas

Jika kita lihat dari perintah yang tersedia maka untuk membangkitkan *klik* kiri misalnya maka kita harus melakukan 2 kali perintah yaitu untuk menekan tombol [*mouse_eventt(MOUSEEVENTTF_LEFTDOWN,sx,sy,0,0);*] dan melepaskan tombol [*mouse_eventt(MOUSEEVENTTF_LEFTUP,sx,sy,0,0);*]. Untuk *double klik* dilakukan dengan melakukan perintah *klik* kiri sebanyak 2 kali. Untuk *klik* kanan dilakukan dengan perintah menekan dan melepaskan tombol kanan. Masalah yang timbul adalah jika kita ingin melakukan *drag and drop*, untuk ini kita harus melakukan perintah penekanan tombol *klik* kiri secara terus menerus dan bila kursor telah ditempatkan pada tempat yang tepat untuk melakukan *drop* maka barulah perintah pelepasan tombol dilakukan. Masalah yang sering timbul adalah pada saat dilakukan penekanan tombol tidak boleh terjadi even mouse pergerakan atau penekanan tombol lain atau bahkan pelepasan tombol karena akan mengakibatkan terjadinya *drop*. Terkadang masalah juga timbul karena saat melakukan hal ini prosedur *drag and drop* kurang tepat sehingga tidak terjadi *drag*, sehingga saat kursor mouse bergerak obyek yang di *drag* tidak ikut bergerak.

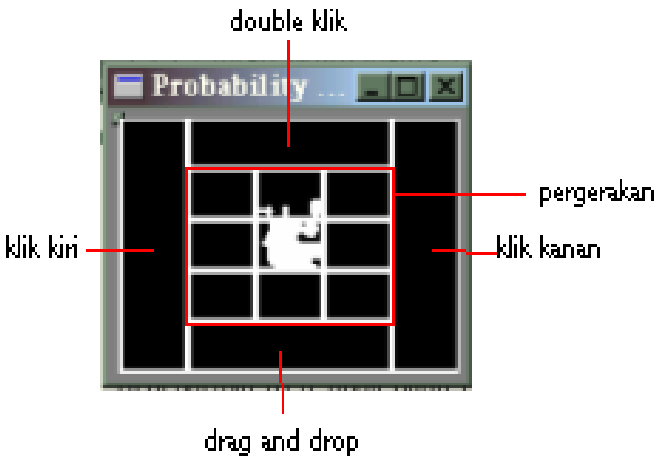
3.7. Metode Pengambilan Data Dan Penanganan *Event*

Seperti yang telah disebutkan di atas bahwa pada tugas akhir ini untuk mengambil data dan menangani *event-event* pada mouse akan dilakukan dengan 3 cara, yang akan dibahas pada sub bab ini.

3.7.1. Metode ROI Untuk Pergerakan dan Juga Untuk Even Khusus

Pada pokok bahasan ini akan dibahas bagaimana memakai ROI sebagai sarana untuk menentukan reaksi komputer dalam menanggapi input image yang masuk melalui kamera. Setelah image diproses dan dicari titik berat dari obyek yang di-*tracking* kemudian image tersebut dibagi-bagi menjadi beberapa daerah. Disini image yang ada dibagi menjadi 13 daerah:

1. Sembilan daerah dipakai untuk mengatur pergerakan yaitu kanan, kiri, atas, bawah, kanan atas, kanan bawah, kiri atas, kiri bawah dan juga tetap/tidak bergerak.
2. Selain itu juga ada 4 daerah ROI yang sangat penting, yaitu yang mengatur *event-event* khusus yaitu *klik* kiri, *klik* kanan, double klik, dan *drag and drop*.



Gambar 3. 6 Pembagian Daerah ROI Untuk Metode ROI

Pembagian daerah tersebut ditunjukkan oleh gambar 3.6. Sebenarnya ROI tidak akan ditampilkan pada suatu window tertentu, hal ini menyebabkan perlu adanya suatu tampilan yang nyata agar memudahkan pemakai untuk menggunakan ROI. Yang harus dilakukan hanyalah menggambar 13 kotak kecil yang besarnya tepat sama seperti ukuran ROI,

```
[       for (rx=32;rx<128;rx=rx+32)
        {
                for (ry=24;ry<96;ry=ry+24)
                {
// pembuatan kotak untuk pergerakan 9 kotak
                        pt1.x = rx;
                        pt1.y = ry;
                        pt2.x = rx+32;
                        pt2.y = ry+24;
```

```
cvRectangle(display,pt1,pt2,CV_RGB(255,255,255),2);
        }
}
pt1.x = 0 ; //kotak untuk  klik kiri
pt1.y = 0;
pt2.x = 32;
pt2.y = 120;
cvRectangle(display,pt1,pt2,CV_RGB(255,255,255),2);
pt1.x = 32;//kotak untuk double klik
pt1.y = 96;
pt2.x = 128;
pt2.y = 120;
cvRectangle(display,pt1,pt2,CV_RGB(255,255,255),2);
pt1.x = 128; //kotak untuk klik kanan
pt1.y = 0;
pt2.x = 160;
pt2.y = 120;
cvRectangle(display,pt1,pt2,CV_RGB(255,255,255),2);
pt1.x = 32; //kotak untuk drag drop
pt1.y = 0;
pt2.x = 128;
pt2.y = 24;]
```

Untuk bagian *drag and drop*, diberikan suatu kondisi bersyarat dalam penggambarannya.

```
[       if (dr)
// kotak bila drag aktif
        cvRectangle(display,pt1,pt2,CV_RGB(255,255,255),8);
        else
// kotak bila drop non aktif
        cvRectangle(display,pt1,pt2,CV_RGB(255,255,255),2); ]
```

Hal diatas ini dimaksukkan agar pemakai mengetahui kondisi *event* ini apakah sedang melakukan *drag and drop* atau tidak. Potongan program di atas

masih belum mengandung perintah untuk membuat ROI, progam di atas hanya berguna untuk menggambar kotak pada tampilan window agar memudahkan pemakai untuk melihat batas-batas ROI.

ROI yang dipakai dalam program tugas akhir ini dibuat dengan cara memanfaatkan koordinat dari obyek yang telah didapat dengan cara menghitung titik beratnya. Setelah titik berat obyek didapatkan maka dari titik berat itu dicari posisinya untuk menentukan di ROI yang mana titik berada. Setelah diketahui di ROI yang mana titik berat itu berada maka komputer memutuskan perintah apa yang harus dikerjakan bila titik berat berada pada ROI itu.

Contohnya bila suatu titik berat berada di ROI untuk menggerakkan kursor entah ke kanan, ke kiri, ke atas, ataupun ke bawah, maka komputer akan menggerakkan kursor mouse kearah yang dimaksud. Contoh lain lagi misalnya bila suatu titik berat berada di ROI untuk *klik* kiri, maka komputer akan melakukan perintah *klik* kiri .

Penentuan batas-batas daerah ROI dilakukan dengan cara if then else aja. Contohnya untuk ROI yang digunakan untuk menentukan *klik* kiri :

```
[if (cx<32) //klik kiri terjadi bila titik berat berada di koordinat <32
{
   if (ste!=1)
   {
// procedure untuk klik kiri
      mouse_eventt(MOUSEEVENTTF_LEFTDOWN,sx,sy,0,0);
      mouse_eventt(MOUSEEVENTTF_LEFTUP,sx,sy,0,0);
// klik kiri  membuat drag menjadi non aktif
      dr=false;
   }
   ste=1;
}].
```

Cara diatas ini dilakukan untuk setiap *event* yang akan ditangani oleh komputer, sehingga kesemuanya akan terdapat 13 *event* dan berarti terdapat 13 batasan-batasan.

Keuntungan dari metode ini adalah kestabilan dan pengontrolan dari kursor dapat dijaga dengan sangat baik. Selain itu juga memungkinkan pemakai dapat mengakses setiap pixel pada layar tanpa kesulitan. Kelemahannya adalah kelambatannya dalam memindahkan kursor ke suatu derah pada layar.

3.7.2. Metode *Tracking* Untuk Pergerakan dan ROI Untuk Even Khusus

Pada metode ini, hasil koordinat dari proses *tracking* langsung diambil dan diubah skala resolusinya sehingga dapat langsung dijadikan input bagi fungsi pergerakan mouse [*SetCursorPos(sx,sy);*]. Rumus yang dipakai untuk mengubah adalah:

$$xl = \frac{rl}{rk} \times xt$$

$$yl = \frac{rl}{rk} \times yt$$

dimana xl,yl = koordinat mouse pada layar

xt,yt = koordinat hasil *tracking*

rl = resolusi layar

rk = resolusi kamera

sebenarnya rumus diatas dapat diterapkan dengan baik pada program, namun sayangnya karena *tracking* menggunakan titik berat maka hasil *tracking* tidak akan mungkin dapat mencapai tepi-tepi layar. Hal ini membuat perlunya perubahan pada rumus yang ada sehingga tepi-tepi layar juga akan dapat diakses. Untuk itu rumus kemudian diubah menjadi:

$$xl = \frac{rl + tx}{rk} \times xt - kx$$

$$yl = \frac{rl + ty}{rk} \times yt - ky$$

dimana tx,ty = tambahan dari rasio perbandingan layar untuk menjangkau tepi kanan dan bawah layar.

kx,ky = pengurangan untuk menjangkau tepi kiri dan atas layar.

Dengan rumus ini maka seluruh bagian dari layar akan dapat dijangkau.

Dengan penggunaan dari rumus itu maka skala perbandingan antara pergeseran layar dan pergeseran kamera adalah:

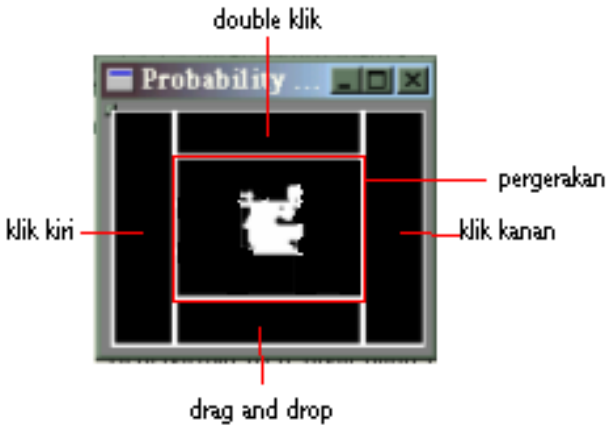
$$sxl : sxo = 1 : \frac{rl + tx}{rk} \times xt - kx$$

$$syl : syo = 1 : \frac{rl + ty}{rk} \times yt - ky$$

dimana sxl,syl = skala pada layar

sxo,syo = skala pada obyek

Untuk *event-event* khusus, metode yang digunakan sama dengan metode sebelumnya yaitu metode ROI. Jadi pada bagian tepi dari display window terdapat 4 daerah ROI sedangkan ditengahnya kosong dan dipakai untuk *tracking*. Gambar pembagian daerah pada window tampak seperti pada gambar 3.7.



Gambar 3. 7 Gambar Daerah ROI Untuk Metode *Tracking*

Hal yang perlu diingat dengan adanya penambahan 4 daerah ROI pada metode ini adalah terjadi perubahan resolusi pada kamera yaitu sebesar

$$rk2x = rk1x - 2 \times lr$$
$$rk2y = rk1y - 2 \times tr$$

dimana $rk2x$,$rk2y$ = resolusi kamera setelah penambahan daerah ROI

$rk2x$,$rk2y$ = resolusi kamera sebelum penambahan daerah ROI

lr = luas daerah ROI di kiri dan kanan

tr = luas daerah ROI di atas dan bawah

Jadi sebenarnya dengan penambahan ROI ini justru akan merugikan karena selain mengurangi resolusi kamera juga mengurangi kemudahan bagi pemakai.

Keuntungan dari metode ini adalah kemudahan bagi pemakai untuk menggunakannya dan kecepatannya dalam mengakses setiap sudut dari layar. Kelemahannya adalah ketidakstabilan dari kursor mouse. Jadi kursor mouse selalu berpindah-pindah dan tidak dapat tetap disuatu tempat. Selain itu juga ketidakakuratan dalam mengakses tiap pixel karena adanya perbandingan tersebut. Bila dilihat dari kelemahan dan keuntungannya sepertinya metode ini benar-benar berlawanan dengan metode sebelumnya.

3.7.3. Metode *Tracking* Berlanjut Untuk Pergerakan dan Metode ROI Untuk *Event* Khusus

Tracking berlanjut sebenarnya sama dengan *tracking* biasa, perbedaannya adalah pada *tracking* berlanjut data koordinat dari hasil *tracking* tidak diubah dan dipakai langsung untuk menjadi koordinat pada layar. Pada *tracking* berlanjut, data yang ada dibandingkan dengan data sebelumnya dan dicari selisihnya. Selisih ini lah yang kemudian dipakai untuk menambah dan mengurangi nilai koordinat x dan atau y pada layar.

Namun bila ini langsung dipakai maka nilai x dan y pada layar hanya akan dapat berubah dengan nilai maksimal sebesar resolusi pada kamera sehingga membuat *tracking* berlanjut menjadi sangat buruk karena kursor mouse hanya akan dapat bergerak sebatas resolusi pada kamera saja. Hal ini memang benar, namun pada *tracking* berlanjut setelah batas dari resolusi dicapai, obyek dapat digerakkan kembali ke belakang dan didorong lagi ke depan dan membuat kursor mouse bergeser lagi sejauh batas resolusi kamera. Jadi secara singkat metode ini sama dengan yang digunakan untuk menggerakkan kursor mouse pada notepad.

Pada notepad jari digeserkan berkali-kali untuk mencapai tempat yang diingini oleh pemakai, pada tugas akhir ini obyek dilalukan berkali-kali didepan kamera sehingga kursor mouse akan bergerak terus sampai posisi yang diinginkan. Keuntungan dari metode ini adalah setiap sudut dari layar akan dapat diakses dengan lebih baik karena tidak memakai perbandingan selain itu dengan metode ini pergerakan juga lebih cepat dilakukan bila dibandingkan dengan

metode ROI. Kelemahan dari metode ini adalah masih tidak dapat stabilnya kursor mouse yang merupakan kelemahan dari metode *tracking*. Kelemahan lain adalah perlunya latihan agar terbiasa melakukan akses mouse dengan metode ini karena memang sulit.

Jika selisih koordinat x,y hasil *tracking* sebelumnya dengan yang sekarang dicari dan ditambahkan ke posisi koordinat x,y pada layar maka semestinya yang terjadi adalah kursor mouse akan terus berputar-putar disekitar daerah kursor mouse awal dengan luas daerahnya sama dengan resolusi layar. Namun dengan suatu cara maka hal ini tidak terjadi, yang terjadi jika kursor telah sampai pada batas pergeseran maksimumnya ia akan dapat terus melanjutkannya dengan cara digeser lagi.

Caranya adalah dengan mendeteksi besar selisihnya. Bila perbedaan selisih kecil maka itu berarti pemakai sedang melakukan pergeseran pada mouse. Bila tiba-tiba terjadi perbedaan selisih yang cukup besar itu berarti pemakai sedang memindahkan obyek ke belakang untuk memulai kembali pergeseran selanjutnya.

Dengan cara inilah maka bila pemakai melakukan pergeseran kecil maka program akan memerintahkan untuk menggeser letak mouse. Namun jika ternyata terjadi pergeseran yang cukup besar ini berarti tidak perlu terjadi pergeseran mouse karena pemakai sedang mengembalikan posisi mouse ke posisi awal untuk memulai kembali pergeseran yang baru. Dengan cara inilah hal ini dapat dilakukan. Perintah untuk melakukan hal ini dapat dilihat pada potongan program berikut [*xl=cx-xl; //xl negatif ke kiri*

```
if (labs(xl)>4 && labs(xl)<15) // bila selisihnya antara 4-15
   sx=sx+xl*3; // pergeseran terjadi
xl=cx; // memori posisi untuk perhitungan berikutnya
yl=yl-cy;  //yl negatif ke atas
if (labs(yl)>3 && labs(yl)<10) // bila selisihnya antara 2-10
   sy=sy+yl*3; // pergeseran terjadi
yl=cy; // memori posisi untuk perhitungan berikutnya
ste=6;]
```

Pada potongan program ini jelas terlihat bahwa perintah pergeseran mouse terjadi hanya bila selisih dari koordinat *tracking* sebesar 4-15 pixel untuk x dan 3-10 pixel untuk y. Untuk *event-event* khusus sama dengan yang digunakan pada metode ROI sehingga tidak akan dibahas lebih lanjut. Sedangkan pembagian daerah ROI juga sama dengan metode *tracking* yaitu seperti yang ditunjukkan gambar 3.7.

Pada akhirnya, perkembangan metode ini dapat dibuat lebih banyak lagi variasinya dan bahkan dapat dikembangkan lebih lanjut sesuai dengan kebutuhan. Bahkan sangat dimungkinkan bila ada metode-metode lain yang mungkin akan jauh lebih baik dari yang telah digunakan disini. Akhirnya semuanya kembali bergantung pada kreativitas dan keinginan untuk mengembangkan baik penggabungan metode untuk *event-event* pergerakan dan *event-event* khusus, maupun mengembangkan variasi dan bahkan menciptakan metode lain yang lebih baik lagi.