BAB IV

HASIL AKHIR DAN ANALISIS

4.1 Spesifikasi Hasil Penelitian

- Alat mampu mengukur jarak
- Alat mampu mendeteksi objek bergerak

4.2 Analisis

Analisis merupakan salah satu bagian hal penting yang harus dilakukan untuk mengetahui apakah alat yang telah direncanakan mampu beroperasi dengan hal yang telah direnacakan dan diharapkan. Hal itu dapat dilihat dari hasil-hasil yang telah dicapai selama pengujian alat. Selain untuk mengetahui apakah alat sudah bekerja dengan baik sesuai dengan yang diharapkan, pengujian juga bertujuan untuk mengetahui kelebihan dan kekurangan dari alat yang dibuat. Hasil-hasil pengujian tersebut nantinya akan dianalisa agar dapat diketahui mengapa terjadi kekurangan. Pengujian pertama dilakukan secara terpisah, dalam artian pengujian tiap fungsi. Kemudaian dilakukan pengujian secara keseluruhan ketika semua fungsi sudah disatukan. Pengujian yang telah dilakukan pada bab ini antara lain :

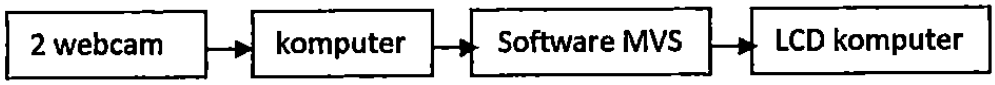
4.2.1 Pengujian Tiap Fungsi

Pengujian ini adalah pengujian awal yang harus dilakukan untuk memastikan webcam dapat bekerja dan mengambil gambar video secara *real time.*

Komponen yang terlibat :

- Komputer Lenovo dan Windows 7
- Webcam intopic 2 buah
- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9
- Kode Sumber Fungsi "Grab_Frame dan Post_Grab"

Alur Pengujian :



Gambar 4.1 Blok Pengujian Mendapatkan Gambar Video RGB

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam juga dipasang di atas dudukan aklrilik
- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;

Mat box4, box5;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);
box4 = fikri_project(Rect(0, 480, 320, 240));
box5 = fikri_project(Rect(320, 480, 320, 240));
if (cam_idx == 0) cams[cam_idx]->getFrame().copyTo(box4);
if (cam_idx == 1) cams[cam_idx]->getFrame().copyTo(box5);
imshow("fikri_skripsi/frame_1", box4);
imshow("fikri_skripsi/frame_2", box5);

cams[cam_idx]>getFinalFrame().copyTo(box4);

cams[cam_idx]->getFinalFrame().copyTo(box5);

imshow("fikri_skripsi/frame_1", box4);

imshow("fikri_skripsi/frame_2", box5);
```

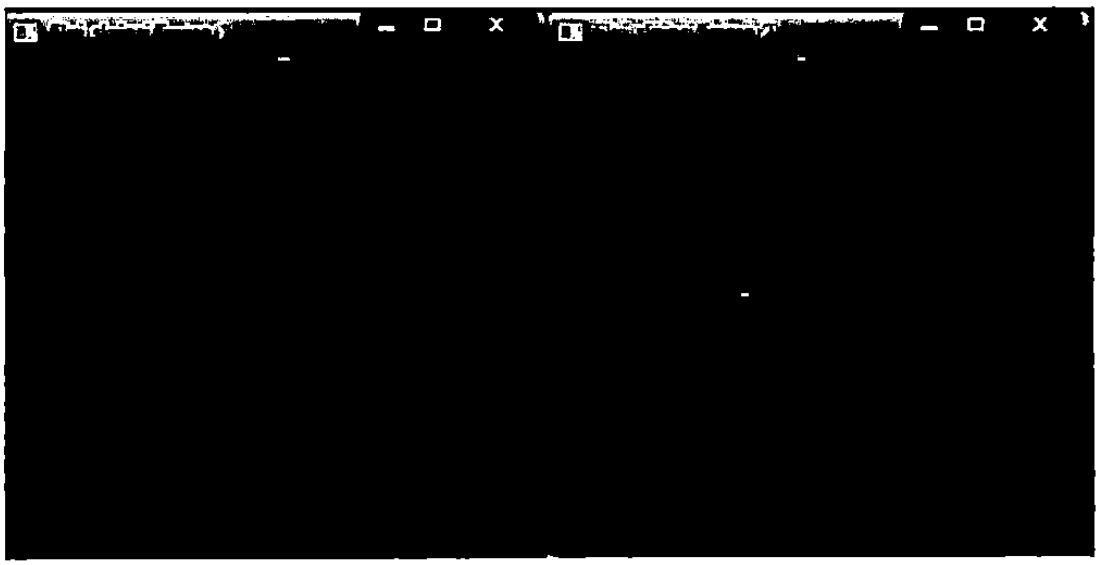
→*file source code 2:*

```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>
Cam::Cam(int camNum) :
rng(12345), isActive_(0), cap_(camNum)
if (cap_.isOpened())  {
isActive_ = 1;
printf("\n Cam %d detected \n", camNum);
cap_.set(CV_CAP_PROP_FRAME_WIDTH, 320);
cap_.set(CV_CAP_PROP_FRAME_HEIGHT, 240);

else{printf("\n Cam %d NOT detected \n", camNum);}
if (isActive_){
cap_ >> frame_;
cvtColor(frame_, frame_gray_, CV_BGR2GRAY);
}
frame_ = Mat::zeros(240, 320, CV_8UC3);
frame_gray_ = Mat::zeros(240, 320, CV_8UC1);
bool Cam::isActive(void){
return isActive_;
void Cam::grab_frame(void){
if (isActive_){
cap_.grab();}}
void Cam::post_grab(void){
cap_.retrieve(frame_, CV_CAP_OPENNI_BGR_IMAGE);
cvtColor(frame_, frame_gray_, CV_BGR2GRAY);
frame_final_ = frame_.clone();}
```

- Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa :

Pada LCD komputer akan ditampilkan hasil *capture* dari webcam berupa gambar video secara *real time*. Gambar di bawah ini merupakan hasil sebelum diintegrasikan dengan fungsi lain.

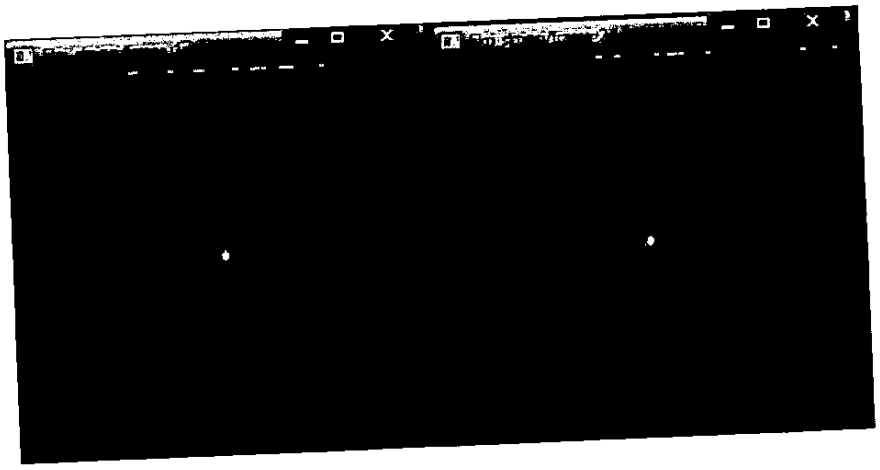


Gambar 4.2 Sebelum Algoritma Stereo Vision

Pada gambar di atas menampilkan hasil tangkapan dari kedua webcam, jendela dengan nama "fikri_skripsi/frame_1" adalah hasil tangkapan dari webcam sebalah kanan, sedangkan jendela dengan nama "fikri_skripsi/frame_1" berasal dari webcam sebelah kiri sebelum kode sumber algoritma stereo vision aktif. Webcam mendeteksi objek berupa tangan dimana antara webcam kanan dan kiri menangkap tangan yang hasilnya berbeda, webcam kanan mampu menangkap gambar tangan hampir keseluruhan, sedangkan webcam satunya hanya setengah tangan yang

dengan webcam sebelah kanan dan juga disebabakan perbedaan tempat antara webcam kanan dan kiri di sumbu Y.

Gambar di bawah ini adalah hasil setelah kode sumber algoritma stereo vision aktif, terdapat perbedaan antara sebelum dan sesudah blok algoritma stereo vision aktif, perbedaan utama adalah adanya titik merah di tengan panel "fikri_skripsi/frame_1" dan "fikri_skripsi/frame_2", dimana titik tersebut titik tengan atau pusat dari lebar sudut webcam mengambil gambar. Pada gambar terdapat titik merah di tengah, titik tersebut adalah nilai *Middle Mass*.



Gambar 4.3 Sesudah Algoritma Stereo Vision Aktif

- **Fungsi Untuk Mendapatkan Gambar Video Gray**

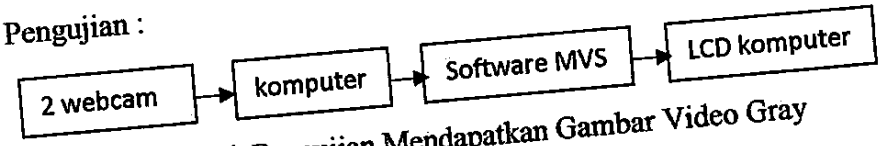
  Pengujian ini untuk menguji apakah kode sumber mampu menghasilkan gambar citra gray. Karena memang citra gray sangat bermanfaat sekali dibidang deometri dan aritmatik.

  Komponen yang terlibat :

  - Komputer Lenovo dan Windows 7
  - Webcam intopic 2 buah

- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9
- Kode Sumber Fungsi "post_grab"

Alur Pengujian :



Gambar 4.4 Blok Pengujian Mendapatkan Gambar Video Gray

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam juga dipasang di atas dudukan aklrilik
- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

→ *file source code 1* :

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;
Mat box0, box1;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);
box0 = fikri_project(Rect(0, 0, 320, 240));
box1 = fikri_project(Rect(320, 0, 320, 240));
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);
```

```
if (cam_idx == 0)

{cvtColor(cams[cam_idx]->getFrameGray(), box0, CV_GRAY2BGR);}

if (cam_idx == 1)

{ cvtColor(cams[cam_idx]->getFrameGray(), box1, CV_GRAY2BGR);}
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);
```

→ *file source code 2* :

```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>

Cam::Cam(int camNum) :
rng(12345), isActive_(0), cap_(camNum)
if (cap_.isOpened())  {
isActive_ = 1;
printf("\n Cam %d detected \n", camNum);
cap_.set(CV_CAP_PROP_FRAME_WIDTH, 320);
cap_.set(CV_CAP_PROP_FRAME_HEIGHT, 240);

else{printf("\n Cam %d NOT detected \n", camNum);}
if (isActive_){
cap_ >> frame_;
cvtColor(frame_, frame_gray_, CV_BGR2GRAY);
}
frame_ = Mat::zeros(240, 320, CV_8UC3);
frame_gray_ = Mat::zeros(240, 320, CV_8UC1);
bool Cam::isActive(void){
return isActive_;
void Cam::grab_frame(void){
if (isActive_){
cap_.grab();}}
void Cam::post_grab(void){
cap_.retrieve(frame_, CV_CAP_OPENNI_BGR_IMAGE);
cvtColor(frame_, frame_gray_, CV_BGR2GRAY);
frame_final_ = frame_.clone();}
```

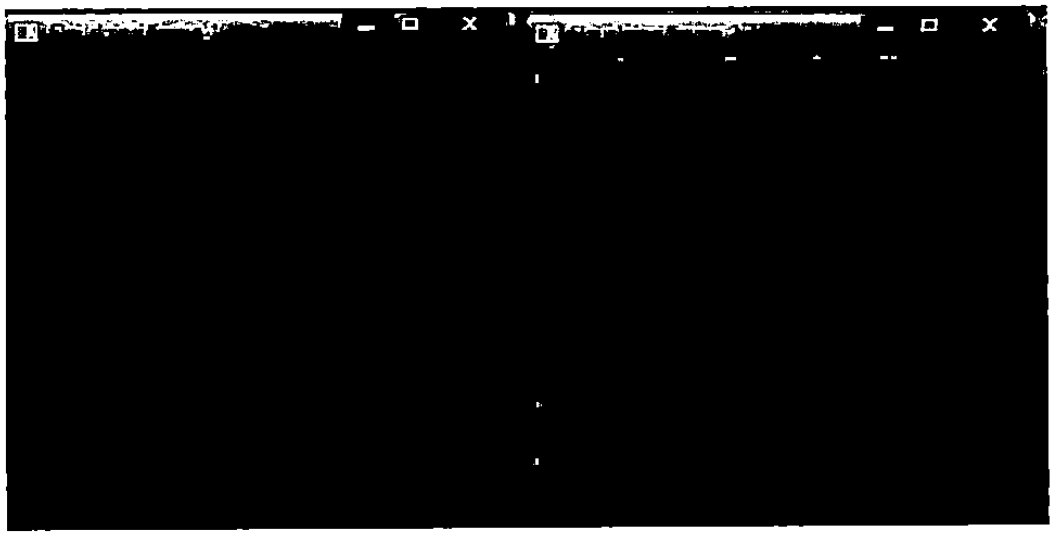
- Jalankan program dengan cara *Debug* di MVS
- Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa :

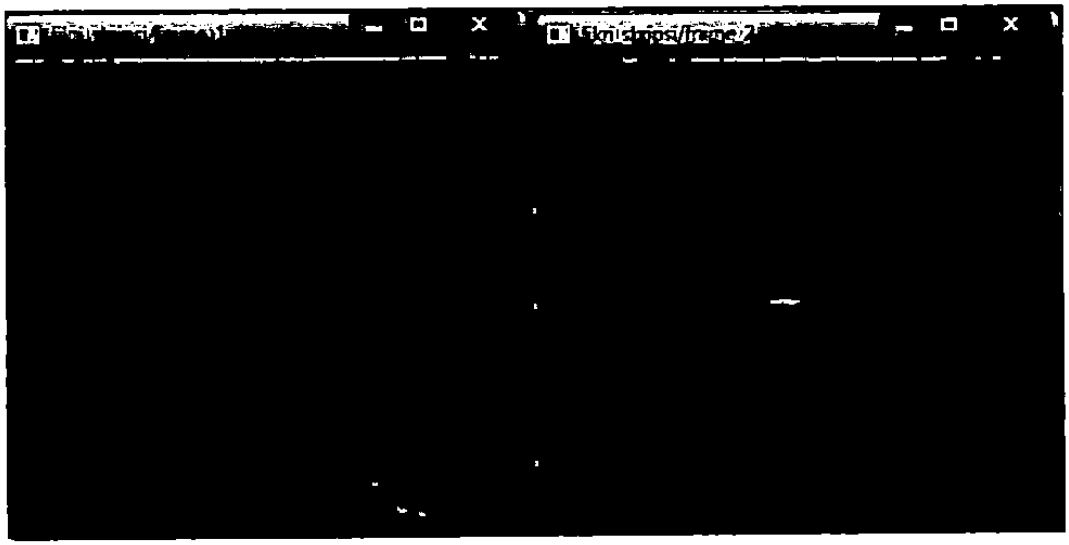
Setelah kode sumber dijalankan maka hasil ditampilkan ke LCD komputer seperti ditunjukkan pada gambar di bawah ini. gambar pertama adalah sebelum kode sumber algoritma stereo vision aktif dimana webcam

"fikri_skripsi/frame_1" dan "fikri_skripsi/frame_2". Sedangkan gambar di bawah adalah setelah kode sumber stereo vision aktif.

Pada fungsi ini kode sumber mampu untuk menangkap gambar video secara *real time*. Namun objek yang ditangkap yang semula berjenis citra RGB dirubah menjadi citra Gray.



Gambar 4.5 Gambar Sebelum Fungsi "post_grab" Aktif



Gambar 4.6 Fugsi "post_grab" Setelah Aktif

Pada gambar di atas, kedua webcam tetap mampu menangkap gambar

mencolok. Sebagaimana perbedaan di istilah stereo vision disebut *displacement* atau *disparity*.

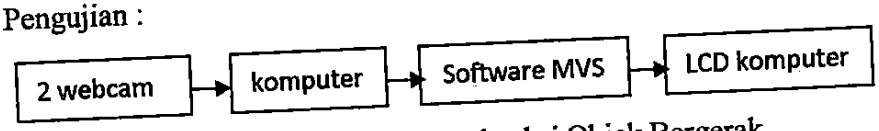
- **Fungsi Untuk Mendeteksi Objek Bergerak**

Pengujian yang ketiga merupakan kelanjutan dari pengujian pertama, yakni pengujian sebuah fungsi kode sumber yang bertugas untuk mendeteksi objek bergerak, dimana objek tersebut akan dijadikan sebagai objek yang akan diukur jaraknya dari webcam ke target tersebut, objek bergerak akan menjadi target persepsi kedalaman *stereo vision*.

Komponen yang terlibat :

- Komputer Lenovo dan Windows 7
- Webcam intopic 2 buah
- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9
- Kode Sumber Fungsi "extractMoving"

Alur Pengujian :



Gambar 4.7 Blok Pengujian Mendeteksi Objek Bergerak

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam juga dipasang di atas dudukan aklrilik

- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

→ *file source code 1* :

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;
Mat box0, box1;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);

box0 = fikri_project(Rect(0, 0, 320, 240));

box1 = fikri_project(Rect(320, 0, 320, 240));
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);

if (cam_idx == 0)

{cvtColor(cams[cam_idx]->getMov(), box0, CV_GRAY2BGR);}

if (cam_idx == 1)

{ cvtColor(cams[cam_idx]->getMov(), box1, CV_GRAY2BGR);}
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);
```

→ *file source code 2* :

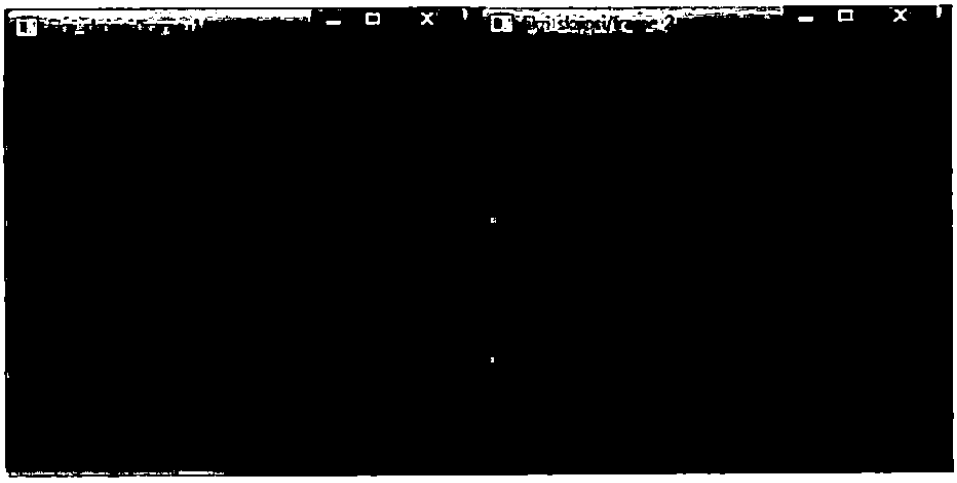
```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>

mov_ = Mat::zeros(240, 320, CV_8UC1);
void Cam::extractMoving(double learningRate){
BGModel(frame_gray_, mov_, learningRate);
medianBlur(mov_, mov_, 5);
threshold(mov_, mov_, 20, 255, THRESH_BINARY);
}
Canny(getMov(), canny_, 0, 30, 3);
dilate_ = canny_.clone();
dilate(dilate_, dilate_, dilation_element_);
```

```
Mat Cam::getMov(void){
return mov_;
}
```

- Jalankan program dengan cara *Debug* di MVS
- Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa

Ini adalah hasil ketika program dijalankan, hasil menunjukkan seolah-olah warna *blank* pada jendela, karena pengujian ini adalah sebelum kode sumber algoritma stereo vision aktif.. sedangkan gambar di bawahnya lagi hasil setelah blok algoritma stereo vision aktif.



Gambar 4.8 Gambar Sebelum Fungsi "extractMoving" Aktif

Pada jendela panel di atas bertuliskan "fikri_skripsi/frame_1" adalah hasil mengambilan data dari webcam sebelah kanan, dan "fikri_skripsi/frame_2" berasal dari webcam sebelah kanan. Pendeteksian benda bergerak dapat metampilkan sebegaimana bentuk aslinya objek yakni tangan, meskipun hasilnya masih cukup kurang jelas. Pada jendela frame_2 gambar yang ditampilkan sedikti berbeda dengan frame_1 hal ini karena perbedaan posisi kedua webcam. Pada pengujian objek yang dalam hal ini tangan cenderung lebih dekat dengan webcam sebelah kanan. Selanjutnya masih akan dilakukan pengujian benda bergerak yang telah disisipi dengan fungsi lain.

- **Fungsi Untuk Mendeteksi Tepi**

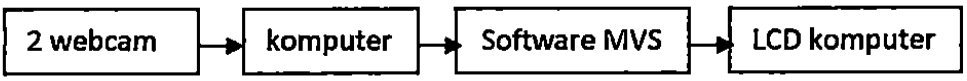
Pengujian ini adalah kelanjutan dari pengujian pendeteksian objek bergerak. Pengujian tepi berfungsi untuk menentukan objek sudut-sudut dari objek bergerak, dimana informasi tersebut berguna untuk mengindentifikasi garis batas (*boudary*) dari suatu objek yang terdapat di citra. Dengan demikian pengujian ini juga bermanfaat di bidang stereo vision agar dapat memustuskan objek mana yang akan diukur kedalamannya.

Komponen yang terlibat :

- Komputer Lenovo dan Windows 7
- Webcam intopic 2 buah
- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9

- Kode Sumber Fungsi "Canny(getMov())"

Alur Pengujian :



Gambar 4.10 Blok Pengujian Mendeteksi Garis Tepi

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam juga dipasang di atas dudukan aklrilik
- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

→ *file source code 1* :

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;
Mat box0, box1;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);

box0 = fikri_project(Rect(0, 0, 320, 240));
box1 = fikri_project(Rect(320, 0, 320, 240));
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);

if (cam_idx == 0)

{cvtColor(cams[cam_idx]->getCanny(), box0, CV_GRAY2BGR);}

if (cam_idx == 1)

{ cvtColor(cams[cam_idx]->getCanny(), box1, CV_GRAY2BGR);}
imshow("fikri_skripsi/frame_1", box0);
```

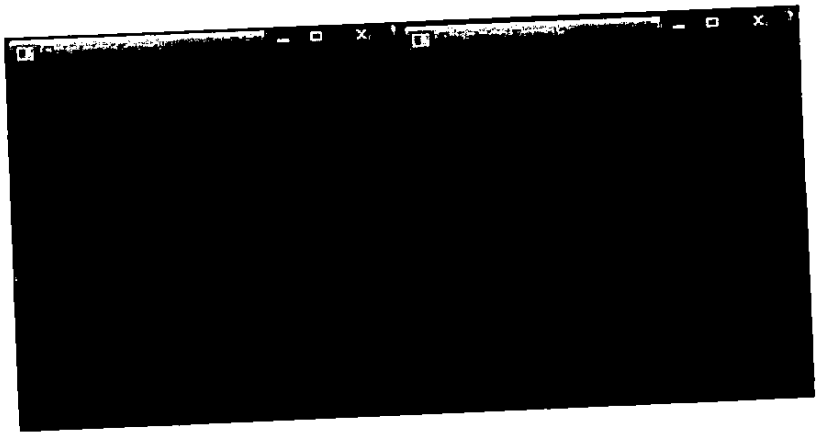
→ *file source code 2* :

```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>

canny_ = Mat::zeros(240, 320, CV_8UC1);
Canny(getMov(), canny_, 0, 30, 3);
dilate_ = canny_.clone();
dilate(dilate_, dilate_, dilation_element_);
```

- Jalankan program dengan cara *Debug* di MVS
- Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa :

Berdasarkan gambar di bawah ini, gambar pertama adalah sebelum fungsi stereo vision jalan, dan gambar satunya lagi setelah fungsi stereo vision jalan atau aktif. Fungsi stereo vision aktif setelah semua rutin fungsi lain selesai dieksekusi. Pada gambar kedua webcam mendeteksi objek berupa tangan dan membentuk garis tepi tangan. Hasil yang dari kedua webcam tidak sama, garis tepi dari webcam sebelah kanan terlihat lebih baik dibandingkan hasil dari webcam sebelah kiri. Webcam sebelah kiri mendeteksi tanganh hanya setengah sedangkan webcam sebelah kanan mendeteksi tangan hampir keseluruhan. Garis tepi yang dihasilkan webcam sebelah kanan sudah lumayan bagus. Meskipun tidak sepenuhnya tiap tepi terdeteksi namun setidaknya hampir semua tepi tangan sudah mampu dideteksi dan hasilnya sudah menyerupai tangan seutuhnya. Pada bagian dalam tangan juga



Gambar 4.11 Gambar Sebelum Fungsi "Canny(getMov())" Aktif

Gambar 4.12 Gambar Garis Bentuk Tangan

➢ **Fungsi Untuk Mendeteksi Bentuk Struktur Elemen Tepi**

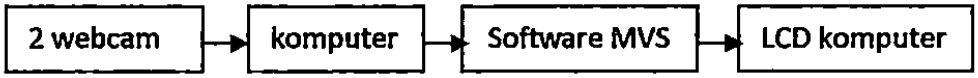
Pengujian ini diharapkan mampu mendeteksi objek bergerak dengan bentuk yang lebih mendekati bentuk aslinya. Pada fungsi ini digabungkannya dua fungsi sebelumnnya, yaitu fungsi pendeteksian objek bergerak dan fungsi pendeteksian tepi. Dengan fungsi ini diharapkan webcam dapat menentukan jarak dari webcam ke target menjadi lebih baik, presisi, dan akurat karena garis bentuk objek bergerak lebih baik dari pada sebelumnya

Komponen yang terlibat :

- Komputer Lenovo dan Windows 7

- Webcam intopic 2 buah
- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9
- Kode Sumber Fungsi "getStructuringElement"

Alur Pengujian :



Gambar 4.13 Blok Pengujian Mendeteksi Bentul Struktur Elemen Tepi

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam juga dipasang di atas dudukan aklrilik
- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

  → *file source code 1* :

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;
Mat box0, box1;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);
```

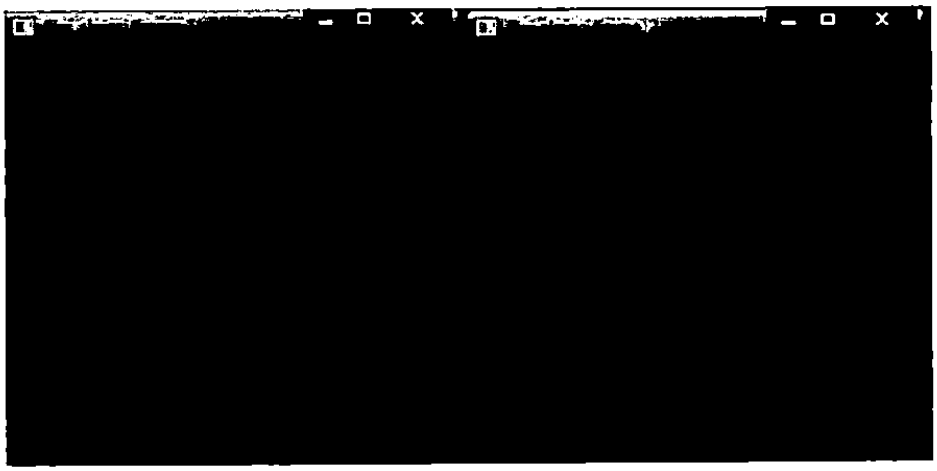
```
box1 = fikri_project(Rect(320, 0, 320, 240));
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);

if (cam_idx == 0)

{cvtColor(cams[cam_idx]->getDilate(), box0, CV_GRAY2BGR);}

if (cam_idx == 1)

{ cvtColor(cams[cam_idx]->getDilate(), box1, CV_GRAY2BGR);}
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);
```

→ *file source code 2* :

```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>

int dilation_size = 8;
dilation_element_ = getStructuringElement(MORPH_RECT,
Size(2 * dilation_size + 1, 2 * dilation_size + 1),
Point(dilation_size, dilation_size));

Canny(getMov(), canny_, 0, 30, 3);
dilate_ = canny_.clone();
dilate(dilate_, dilate_, dilation_element_);

void Cam::Contours(void){
cont_ = dilate_.clone();}
```

- Jalankan program dengan cara *Debug* di MVS
- Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa :



Gambar 4.14 Sebelum Fungsi "getStructuringElement" Aktif

Gambar 4.15 Setelah Fungsi "getStructuringElement" Aktif

Seperti terlihat pada gambar di atas dengan kode sumber ini proses pembentukan bentuk tangan menjadi lebih baik. Pada jendela "fikri_skripsi/frame_1" memperlihatkan bentuk tangan dengan ditandai garis putih-putih berdiameter lebih besar dari pada di gambar pendeteksian objek bergerak. Garis putih tersebut melingkari objek sesuai dengan garis tepi yang dibentuk oleh fungsi pendeteksian tepi. Bukti bahwa dengan fungsi pendeteksian bentuk objke menjadi lebih baik adalah pada citra yang ditampilkan di jendela "fikri_skripsi/frame_2". Jika melihat pada kedua fungsi di atas bahwa jendela kedua atau webcam sebelah kiri selalu menampilkan bentuk objek yang kurang baik, namun dengan fungsi ini terlihat garis bentuk objek menjadi sangat baik meskipun webcam sebelah kiri hanya menangkap objek setengah tangan. Namun karena masih setengah sehingga belum dapat diputuskan apakah objek tersebut tangan atau bukan, akan tetapi setidaknya hasilnya sudah baik.

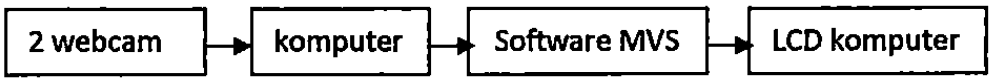
➢ **Fungsi Untuk Mendeteksi Garis Bentuk**

Fungsi ini berguna sebagai pembantu pendeteksian objek bergerak dan tepi objek. Dengan fungsi ini kemampuan mendeteksi objek bergerak menjadi lebih sensitif.

Komponen yang terlibat

- Komputer Lenovo dan Windows 7
- Webcam intopic 2 buah
- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9
- Kode Sumber Fungsi "Contours();"

Alur Pengujian



Gambar 4.16 Blok Pengujian Mendeteksi Garis Bentuk

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam juga dipasang di atas dudukan aklrilik
- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

→ *file source code 1*:

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;
Mat box0, box1;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);

box0 = fikri_project(Rect(0, 0, 320, 240));
box1 = fikri_project(Rect(320, 0, 320, 240));
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);

if (cam_idx == 0)

{cvtColor(cams[cam_idx]->getContours(), box0, CV_GRAY2BGR);}

if (cam_idx == 1)

{ cvtColor(cams[cam_idx]->getContours(), box1, CV_GRAY2BGR);}
imshow("fikri_skripsi/frame_1", box0);
imshow("fikri_skripsi/frame_2", box1);
```

→ *file source code 2*:

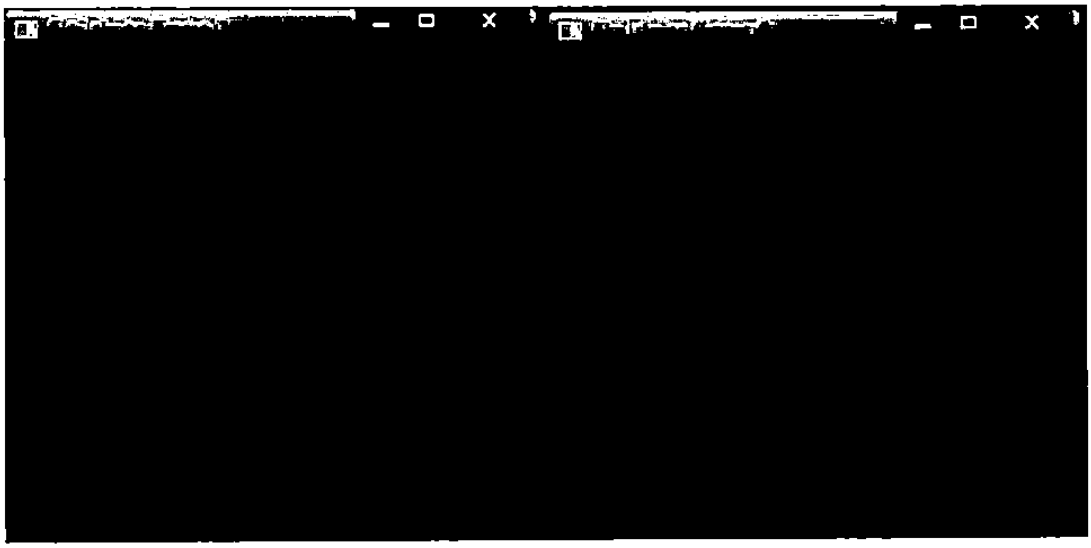
```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>

Contours();
if (0 < contours_.size() && contours_.size() < MAX_CONT){
for (int cont_idx = 0; cont_idx < MAX_CONT; cont_idx
cont_obj_[cont_idx] = -1;}
vector<vector<Point> > cont_poly_(contours_.size());

for (unsigned int i = 0; i < contours_.size();
approxPolyDP(Mat(contours_[i]), cont_poly_[i], 3, true);
minEnclosingCircle(cont_poly_[i], cont_center_[i],
cont_radius_[i]);
}
if (found_obj != -1){
              pairs_[pairs_idx].cont_idx = cont_idx;
              pairs_[pairs_idx].obj_idx = found_obj;
              pairs_[pairs_idx].likeness = max_likeness;
              pairs_idx++;
       }
```

- Jalankan program dengan cara *Debug* di MVS

Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa

Sebagaimana percobaan fungsi-fungsi sebelumnya, jika algoritma stereo vision belum aktif maka jendela panel akan menampilkan warna hitam pekat. Akan tetapi jika sudah aktif maka fungsi akan dijalankkan. Pengujian kali ini adalah pengujian garis bentuk atau dikenal dengan istilah *contour*. Dengan fungsi ini maka kemampuan sistem yang semula hanya mendeteksi garis tepi objek, maka sebagian garis tepi akan dikelilingi garis berbentuk lingkaran, garis tepi yang akan dikelilingin garis lingkaran hanya garis tepi yang benar-benar sudah jelas dan pasti. Dapat dilihat pada gambar yang sebelah bawah disitu terdapat garis-garis yang akan dikelilingin garis lingkaran.



Gambar 4.17 Saklar Fungsi "Contours()" Aktif

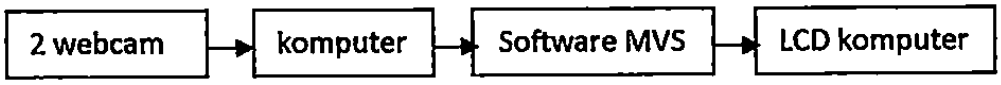
Gambar 4.18 Setelah Fungsi "Contours()" Aktif

➢ **Fungsi Untuk Mengukur Jarak atau Fungsi Algoritma Stereo Vision**

Komponen yang terlibat :

- Komputer Lenovo dan Windows 7
- Webcam intopic 2 buah
- Dudukan webcam
- Kabel perpanjangan USB 2 buah
- *Software* Microsoft Visual Studio 2013
- Pustaka OpenCV 2.4.9
- Kode Sumber Fungsi "Void StereoInfo"

Alur Pengujian :



Gambar 4.19 Blok Pengujian Pengukuran Jarak Metode *Stereo Vision*

- Dua webcam masing-masing dihubungkan dengan kabel perpanjangan USB agar webcam dapat berpindah-pindah, webcam

- Masing-masing kabel perpanjangan dihubungkan ke Komputer, satu kabel perpanjangan USB mendapatkan satu Port.
- Komputer akan mendapatkan inputan dari webcam, diolah oleh *software* MVS, dan hasil ditampilkan ke LCD Komputer
- Kode Sumber program fungsi ini adalah :

file source code 1:

```
#define _CRT_SECURE_NO_WARNINGS
#include "ctime"
#include <opencv/cv.h>
#include "opencv2/cams1.h"
#define CREATEMOVIE 0

using namespace cv;
namedWindow("fikri_skripsi/frame_1", CV_WINDOW_FREERATIO);
namedWindow("fikri_skripsi/frame_2", CV_WINDOW_FREERATIO);
Mat box4, box5;
box4 = fikri_project(Rect(0, 480, 320, 240));
box5 = fikri_project(Rect(320, 480, 320, 240));
cams[cam_idx]->extractMoving(0.007);
if (cam_idx == 0) cams[cam_idx]->getFrame().copyTo(box4);
if (cam_idx == 1) cams[cam_idx]->getFrame().copyTo(box5);
imshow("fikri_skripsi/frame_1", box4);
imshow("fikri_skripsi/frame_2", box5);
if (cam_idx == 0) {
cams[cam_idx]->getFinalFrame().copyTo(box4);}
if (cam_idx == 1) {
cams[cam_idx]->getFinalFrame().copyTo(box5);}
imshow("fikri_skripsi/frame_1", box4);
imshow("fikri_skripsi/frame_2", box5);
```

file source code 2:

```
#include <opencv2/cams1.h>
#include <stdio.h>
#include <stdlib.h>

frame_final_ = Mat::zeros(240, 320, CV_8UC3);

void Cam::post_grab(void){
cap_.retrieve(frame_, CV_CAP_OPENNI_BGR_IMAGE);
cvtColor(frame_, frame_gray_, CV_BGR2GRAY);
frame_final_ = frame_.clone();
}

Object* StereoInfo::getObject(int obj_idx){
return cam0->getObject(obj_idx);}
```

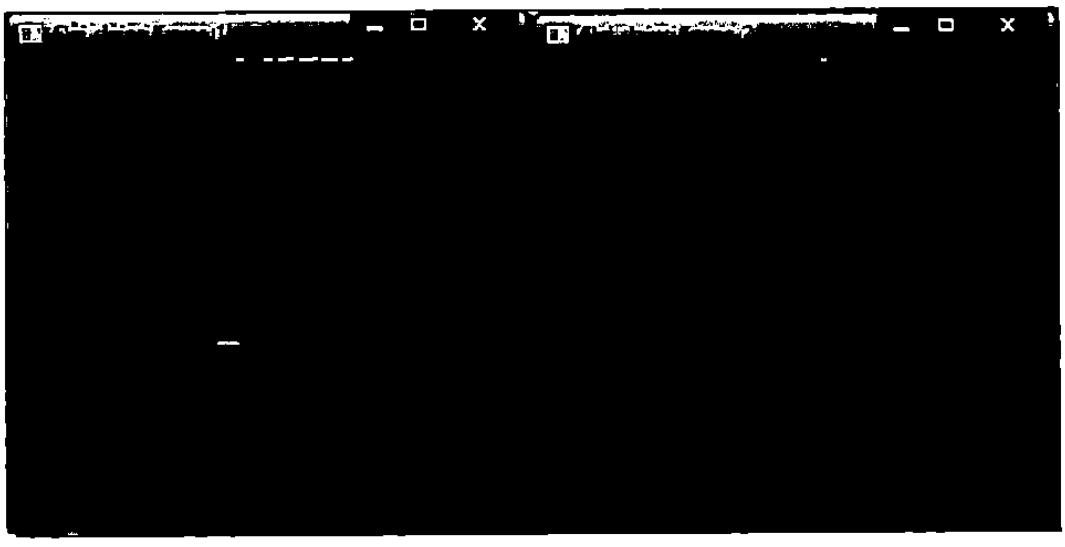
```
return (int)cam0->getObject(obj_idx)->obj_pos_.x - cam1-
>getObject(matchInfo[obj_idx])->obj_pos_.x;}
void StereoInfo::drawInfo(Mat dst0, Mat dst1){
Scalar color_black(0, 0, 0);
Scalar color_white(255, 255, 255);
Object* obj;
Object* match;

int h_displ, distance, distance1;
float c1 = 5000;
float c2 = 1;

Point txt_pos, pt1, pt2;
Point txt_pos1, pt3, pt4;

for (unsigned int obj_idx = 0; obj_idx < cam0->max_objects;
obj_idx++){
obj = cam0->getObject(obj_idx);
if (obj->obj_cont_ != -1 && matchInfo[obj_idx] != -1){
match = cam1->getObject(matchInfo[obj_idx]);
      h_displ = abs(obj->obj_pos_.x - match->obj_pos_.x);
      int avg_displ = 0;
      for (int i = 0; i< 3; i++){
      avg_displ = avg_displ + displ_hist[obj_idx][i];
      }
      avg_displ = avg_displ + h_displ;
      avg_displ = avg_displ / 4;
      for (int i = 2; i> 0; i--){
      displ_hist[obj_idx][i] = displ_hist[obj_idx][i - 1];
      }
      displ_hist[obj_idx][0] = h_displ;
      if (h_displ != 0 && avg_displ != 0){
            distance = c1 / ((float)h_displ + c2); //cm
      distance1 = c1 / ((float)avg_displ + c2); //cm

}
      else{
            distance = -1;
            distance1 = -1;
      }
      circle(dst0, obj->obj_pos_, 4 + (int)obj->obj_radius_ /
      2, obj->obj_color_, 4, 8, 0);
      circle(dst1, match->obj_pos_, 4 + (int)match-
      >obj_radius_ / 2, obj->obj_color_, 4, 8, 0);

      char text[8];
      char text1[8];
      txt_pos = Point(obj->obj_pos_.x + obj->obj_radius_ / 2,
      obj->obj_pos_.y + obj->obj_radius_ / 2);
      txt_pos1 = Point(obj->obj_pos_.x + obj->obj_radius_ /
      2, obj->obj_pos_.y + obj->obj_radius_ / 2 + 15);
      Cam::itoa(distance, text, 10);
      Cam::itoa(distance1, text1, 10);

      pt1 = Point(txt_pos.x, txt_pos.y - 13);
      pt2 = Point(pt1.x + 40, pt1.y + 15);
      pt3 = Point(pt1.x, pt2.y);
      pt4 = Point(pt3.x + 40, pt3.y + 15);
```

```
            rectangle(dst0, pt1, pt2, color_black, -1, 8, 0);
            putText(dst0, text, txt_pos, CV_FONT_ITALIC, 0.5,
            color_white, 2, 8, false);
            rectangle(dst0, pt3, pt4, color_white, -1, 8, 0);
            putText(dst0, text1, txt_pos1, CV_FONT_ITALIC, 0.5,
            color_black, 2, 8, false);
            }
            }
}
```

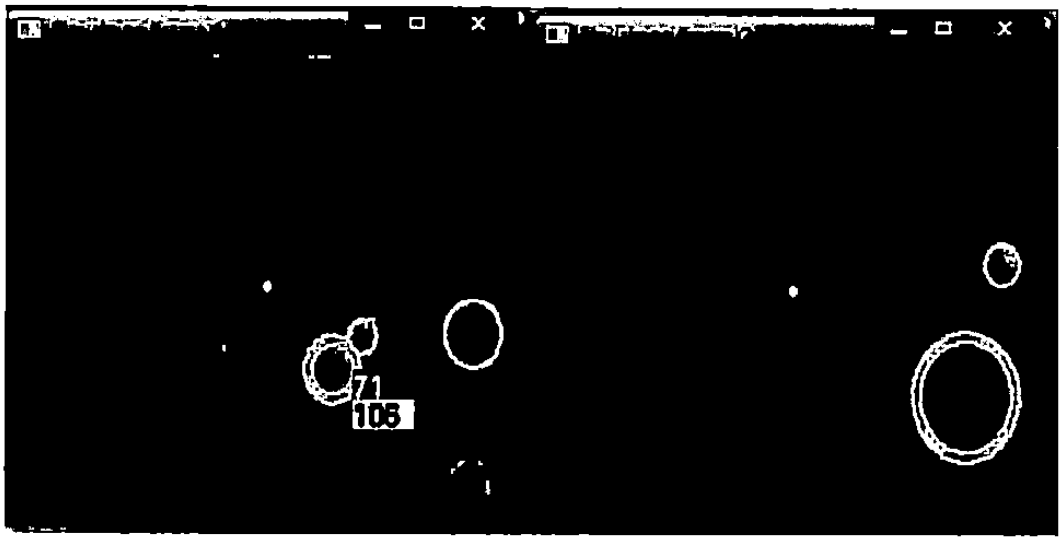
- Jalankan program dengan cara *Debug* di MVS
- Mengamati hasil yang ditampilkan ke LCD komputer

Hasil dan Analisa :

Setelah program dijalankan, selanjutnya program akan mendeteksi pergerakan sebuah objek dan mengukur jarak terhadap objek tersebut dalam hal ini objek berupa tangan. Seperti terlihat pada gambar di bawah. Program sudah mampu mengukur jarak dan menteksi pergerakan objek. Di gambar terdapat dua kolom angka yakni *background* putih dan hitam, dengan warna tulisan putih dan hitam. Tulisan yang berwarna putih adalah nilai jarak, dan yang berwarna hitam adalah nilai rata-rata jarak.



Gambar 4.20 Sebelum Fungsi "Void StereoInfo" Aktif



Gambar 4.21 Setelah Fungsi "Void StereoInfo" Aktif

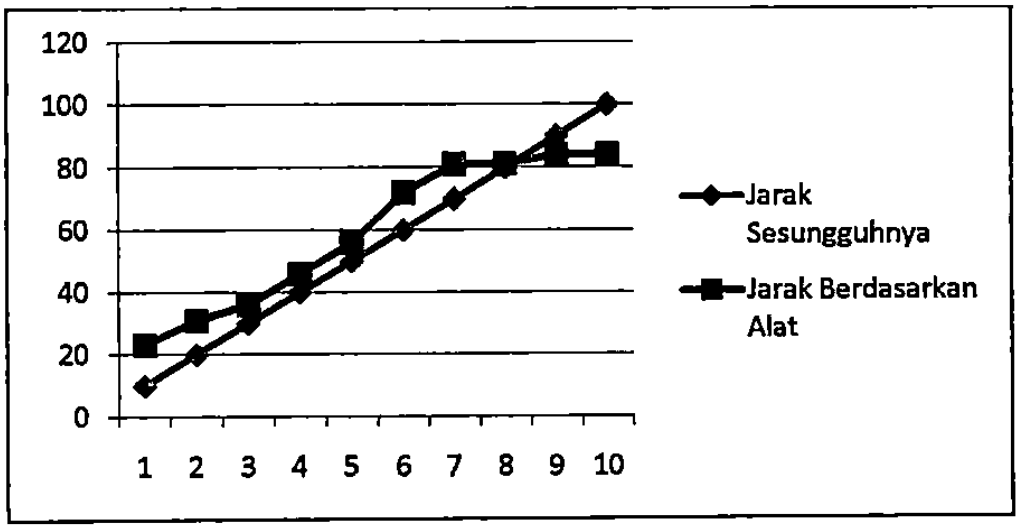
4.3 Pengukuran Jarak Terhadap Lilin

Pada Pengujian ini adalah uji coba pengukuran jarak dari webcam ke lilin. Lilin adalah target utama misi robot pemadam api yang harus dilakukan. Setelah dilakukan pengambilan data, maka di dapat seperti tertera di bawah ini.

Tabel 4.1 Data Pengukuran Jarak Ke Lilin

Sample	Jarak Sesungguhnya (Cm)	Jarak Berdasarkan Alat (Cm)	Selisih Jarak (Cm)	Presentase Selisih Jarak (%)
1	10	23	13	130
2	20	31	11	55
3	30	36	6	20
4	40	46	6	15
5	50	56	6	12
6	60	72	12	20
7	70	81	11	15.71
8	80	81	1	1.25
9	90	84	6	6.67
10	100	84	16	16
Total Selisih Jarak				29.163

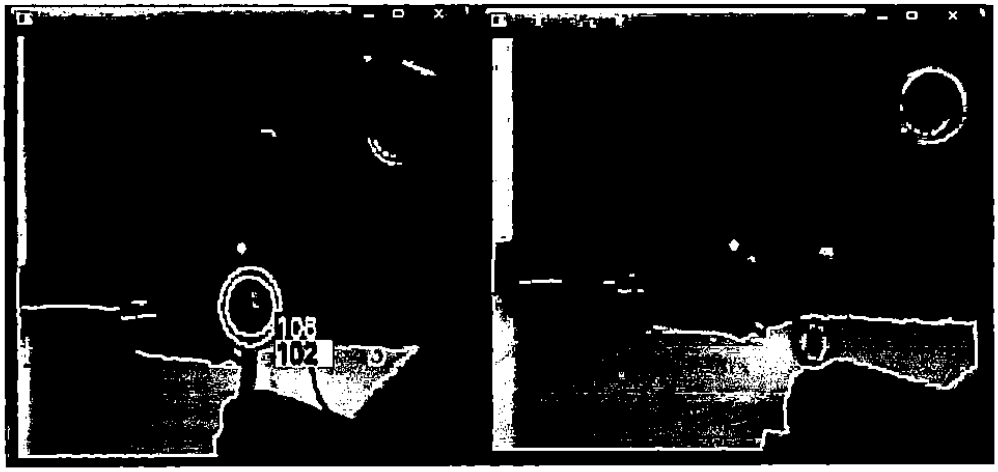
Pada tabel di atas, terdapat 5 kolom, kolom pertama adalah pengujian jarak ke lilin, kolom kedua adalah data jarak sesungguhnya dari webcam ke lilin. Sesangkan kolom ketiga adalah hasil pengukuran yang didapat dari webca, selanjutnya adalah kolom selisih jarak sesungguhnya dengan jarak berdasarkan alat, dan terakhir adalah nilai presentase nilai selisih jarak. Pada pengambilan data ini nilai yang terdapat di kolom 3 adalah nilai yang pernah muncul di LCD komputer. Namun karena metode pengukuran jarak berdasarkan objek bergerak sehingga agar dapat mengukur jarak, objek harus bergerak secara terus-menerus. Nilai yang muncul selalu berubah-rubah dan tidak tetap sehingga sulit sekali untuk menghasilkan nilai pasti yang berdekatan dengan nilai jarak sesungguhnya. Alat ini belum mampu mengukur jarak pada besaran satuan, hanya mampu mengukur di besaran puluhan, sedangkan kemampuan sensor yang dibutuhkan oleh robot pemadam api adalah yang mampu mengukur jarak di skala satuan dan puluhan *centimeter*.



Tabel 4.2 Data Jarak Sensor Ultrasonik

Jarak dengan benda (cm)	Data Ping))) Ultrasonic
1	377
2	377
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20

Tabel di atas adalah hasil pengukuran jarak menggunakan sensor *Ping Ultrsonic*. Berdasarkan nilai yang tertera di tabel dan pengalaman penerapan sensor di robot, robot mampu menelusuri dinding, memadamkan api terbaik di jarak 10 cm akan tetapi robot juga mampu memadamkan api selama dalam rentang 3 cm sampai 20 cm. Jika dibandingkan dengan tabel pengukuran menggunakan webcam, kemampuan sensor ping masih lebih baik dan lebih layak digunakan oleh robot[1].



Gambar 4.23 Pengujian Terhadap Lilin

4.4 Pelajaran Yang Diperoleh

Penelitian yang dilakukan ini memberikan pengetahuan tambahan serta pelajaran bagi penulis. Dalam proses penelitian menuntut perencanaan yang matang dan didukung dengan studi yang memadai sehingga penelitian dapat berjalan dengan lancar. Kesabaran, pantang putus asa dan tanggung jawab adalah hal penting dalam proses penelitian.

Cobaan dan permasalahan-permasalahan yang sempat dialami diantaranya, seringnya terjadi error pada fungsi program karena memang harus menyesuaikan dengan sintaks-sintaks antara bahasa C dan OpenCV, dan juga algoritma *motion detection* masih memerlukan perbaikan dari segi *filtering* objek.

Ilmu pengetahuan yang penulis peroleh pada pembuatan skripsi ini antara lain menambah wawasan penulis dalam hal perancangan perangkat lunak dan keras sebuah sistem, dimana penulis dapat lebih paham dan mengerti dalam menentukan sebuah sistem kerja yang akan dibuat untuk menghasilkan sebuah sistem pengukur jarak berbasis webcam. Menambah wawasan penulis dalam hal algoritma dan pemograman yang menggunakan bahasa C++.