

CHAPTER V

IMPLEMENTATION AND TESTING

5.1. Implementation

5.1.1 Implementation Program

As already described above, program through 2 stages, namely preprocessing and identification process. For preprocessing is grayscale, filtering, resizing, sharpening, tresholding and edge detection while the identification process is calculate the correlation coefficient value. The image used there are two types, reference images and test image. Firstly, each type of image must been done the preprocessing so that the output process can be used in the identification process. This is the display of the program.

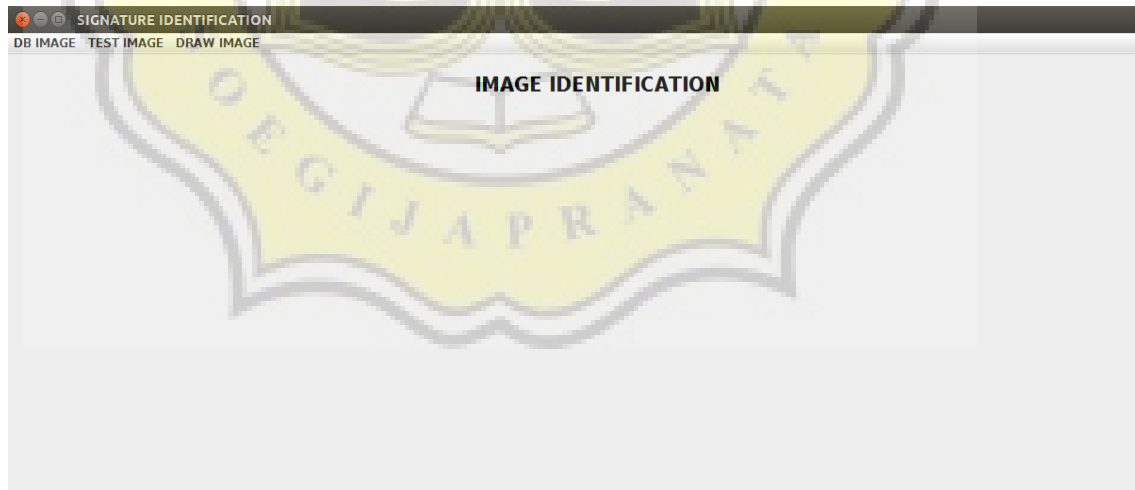


Figure13 : Main program display


The program has three menu i.e. “db image”, “test image” and “draw image”. “Db image” menu has a submenus namely “list db” and “add db

image'. "Test image" menu has a submenus "process" and "identification". Draw image menu has submenu "draw".

5.1.1.1 Preprocessing

To run the preprocessing for each type of image, at the reference image can be done by selecting the "add db image" submenu on "db image" menu, while on the test image can be done by selecting submenu "process" on the "test image" menu. The input image will first through a grayscale process then filtering, sharpening, tresholding and sobel, but before grayscale process program will checked whether size is greater or smaller than 100 x 100 then it will go through resizing process.

As explained above, the program is work by using the output of a process for be used as input on the another process because one process is depending on other process. So on the program by clicking the "process" button then will run a preprocessing until the edge detection process.



```

122 public proses() {
123     tampilan();
124 }
125
126
127 class go implements ActionListener
128 {
129     public void actionPerformed(ActionEvent evt)
130     {
131         tempatgambar1.removeAll();
132         tempatgambar2.removeAll();
133
134         Btn();
135         Lbl();
136         Img();
137
138         grayscale g = new grayscale();
139         g.grayscale(gambarp);
140         filter f = new filter();
141         f.filter(g.getString());
142         sharpening s = new sharpening();
143         s.sharpening(f.getString());
144         tresholding t = new tresholding();
145         t.tresholding(s.getString());
146         sobel ss = new sobel();
147         ss.sobel(t.getString());
148
149         pic2 = new ImageIcon(ss.getString());
150         gambar2 = new JLabel(pic2);
151
152         tempatgambar1.add(gambar1);
153         tempatgambar2.add(gambar2);
154         tempatgambar1.revalidate();
155         tempatgambar2.revalidate();
156     }
157 }
158

```

Figure14 : preprocessing code

5.1.1.2 Resizing

First of all before going to the grayscale process, program will check whether size image is smaller or larger than 100 x 100. If the conditions are meet, then program will run the resizing process. After resizing is finished then the output image will be used in the process of grayscale.

```

w2 = width;
h2 = height;

//-----store rgb to array 2d-----//
for(int i=0;i<h1;i++)
{
    for(int j=0;j<w1;j++)
    {
        Color c = new Color(image.getRGB(j,i));
        inputred[i][j] = c.getRed();
        inputgreen[i][j] = c.getGreen();
        inputblue[i][j] = c.getBlue();
    }
}

x_rasio=w2/(double)w1;
y_rasio=h2/(double)h1;

for(int i=0;i<hasilred.length;i++) {
    for(int j=0;j<hasilred[0].length;j++) {
        posx = (int)Math.floor(i/x_rasio);
        posy = (int)Math.floor(j/y_rasio);

        hasilred[i][j] = inputred[posx][posy];
        hasilgreen[i][j] = inputgreen[posx][posy];
        hasilblue[i][j] = inputblue[posx][posy];
    }
}

BufferedImage gambarbaru = new BufferedImage(w2, h2, image.getType());
for(int i=0;i<hasilred.length;i++) {
    for(int j=0;j<hasilred[0].length;j++) {
        Color newColor = new Color(hasilred[i][j],hasilgreen[i][j],hasilblue[i][j]);
        gambarbaru.setRGB(j, i, newColor.getRGB());
    }
}

```

Figure15 : Resizing code

5.1.1.3 Edge Deteciton

Edge detection methods used in this program is the sobel method. Before the edge detection process, the image will going through the process of grayscale, filtering, sharpening and tresholding. Image output from edge detection process will be used in the identification process, namely to calculate the correlation coefficient value.

```

resize.java x grayscale.java x filter.java x sharpening.java x thresholding.java x sobel.java x
106         for(int i=1;i<width-1;i++)
107         {
108             for(int j=1;j<height-1;j++)
109             {
110                 hred=(htempred[i-1][j-1]*kernel[0][0])+(htempred[i-1][j]*kernel[0][1])+(htempred[i-1][j+1]*kernel[0][2])
+(htempred[i][j-1]*kernel[1][0])+(htempred[i][j]*kernel[1][1])+(htempred[i][j+1]*kernel[1][2])+(htempred[i+1][j-1]*kernel[2][0])
+(htempred[i+1][j]*kernel[2][1])+(htempred[i+1][j+1]*kernel[2][2]);
111                 hbblue=((htempblue[i-1][j-1]*kernel[0][0])+(htempblue[i-1][j]*kernel[0][1])+(htempblue[i-1][j+1]*kernel
[0][2]))+(htempblue[i][j-1]*kernel[1][0])+(htempblue[i][j]*kernel[1][1])+(htempblue[i][j+1]*kernel[1][2]))+(htempblue[i+1
][j-1]*kernel[2][0])+(htempblue[i+1][j]*kernel[2][1])+(htempblue[i+1][j+1]*kernel[2][2]));
112                 hgreen=((htempgreen[i-1][j-1]*kernel[0][0])+(htempgreen[i-1][j]*kernel[0][1])+(htempgreen[i-1][j+1]*kernel
[0][2]))+(htempgreen[i][j-1]*kernel[1][0])+(htempgreen[i][j]*kernel[1][1])+(htempgreen[i][j+1]*kernel[1][2]))+(htempgreen[i+1
][j-1]*kernel[2][0])+(htempgreen[i+1][j]*kernel[2][1])+(htempgreen[i+1][j+1]*kernel[2][2]));
113
114                 if(hred>255){
115                     hred=255;
116                 }
117                 if(hgreen>255){
118                     hgreen=255;
119                 }
120                 if(hbblue>255){
121                     hbblue=255;
122                 }
123                 if(hred<0){
124                     hred=0;
125                 }
126                 if(hgreen<0){
127                     hgreen=0;
128                 }
129                 if(hbblue<0){
130                     hbblue=0;
131                 }
132             }
133         }

```

Figure16 : Edge detection code

5.1.1.4 Correlation Coefficient

The results of the output image from the edge detection process would be the input of the process of calculating the coefficient of correlation. Test image will look for the value of the correlation coefficient with any existing database image.

```

kurang1=kurang1*kurang1;
kurang2=kurang2*kurang2;
totalkuadrat1=totalkuadrat1+kurang1;
totalkuadrat2=totalkuadrat2+kurang2;

kurang1=0; kurang2=0; temp=0;
kurang1k=0; kurang2k=0;
}
}

BigDecimal varsatudua = BigDecimal.valueOf(totalkali);
System.out.println(varsatudua);

BigDecimal kiri = BigDecimal.valueOf(totalkuadrat1);
BigDecimal kanan = BigDecimal.valueOf(totalkuadrat2);
BigDecimal isiakar = kiri.multiply(kanan);
BigDecimal bawah = new BigDecimal(Math.sqrt(isiakar.doubleValue()));

BigDecimal r = varsatudua.divide(bawah,2,BigDecimal.ROUND_HALF_UP);
koefisien = r.floatValue();

System.out.println(koefisien);

ratasatu=0; ratadua=0;
totalkali=0; totalkuadrat1=0; totalkuadrat2=0;
varsatudua = BigDecimal.valueOf(0);
kiri = BigDecimal.valueOf(0);
kanan = BigDecimal.valueOf(0);
isiakar = BigDecimal.valueOf(0);
bawah = BigDecimal.valueOf(0);
r = BigDecimal.valueOf(0);
}
}

```

Figure17 : Correlation coefficient code

5.1.1.5 Tree

```

class berjalan implements ActionListener
{
    public void actionPerformed(ActionEvent evt)
    {
        float r = 0;
        int count=0;
        String strLine;
        ketentuan.removeAll();
        hasil.removeAll();
        Btn();
        Lbl();
        teks();
        try {
            BufferedWriter result = new BufferedWriter(new FileWriter("hasildeteksi.txt"));
            BinaryTree x = new BinaryTree();

            koefisientest k = new koefisientest();
            File directory= new File(System.getProperty("user.dir")+"/dbgambar");
            for (File file2 : directory.listFiles())
            {
                if (file2.getName().endsWith("sobel.png"))
                {
                    //System.out.println("TES = "+namagambar1+" REF = "+file2.getCanonicalPath());
                    System.out.println(file2.getParent());
                    k.koefisientest(namagambar1,file2.getCanonicalPath());
                    float temp = k.getNilai();
                    x.add(temp);
                    result.write("Gambar = "+file2.getName()+" Hasil = "+temp+"\n");
                    count++;
                }
            }
            result.close();

            int v=1:

```

Figure18: calculate correlation coefficient all database image

The process of calculating the value of the correlation coefficient is done with output image from edge detection process which is test image and reference image. The test image will be done the calculation of correlation coefficient with all of the reference image and each value will be inserted into binary tree data structure.

```

}
public float getRoot() {
    return root.getIsi();
}
public void insert(Node parent,Node baru) {
    if(baru.getIsi()<=parent.getIsi()) {
        if(parent.getKi()!=null) {
            insert(parent.getKi(),baru);
        }
        else {
            System.out.println("Tambah "+baru.getIsi()+" di kiri node "+parent.getIsi());
            parent.setKi(baru);
        }
    }
    else if(baru.getIsi(>parent.getIsi()) {
        if(parent.getKa()!=null) {
            insert(parent.getKa(),baru);
        }
        else {
            System.out.println("Tambah "+baru.getIsi()+" di kanan node "+parent.getIsi());
            parent.setKa(baru);
        }
    }
    else {
        System.out.println("NO DATA");
    }
}
}

```

Figure19 : Insert tree code

Each reference image and value of correlation coefficient is appear in the identification process display, so it can be known the reference image which has the similarity with the input image from the maximum value of correlation coefficient.

```

        System.out.println("Tambah "+baru.getIsi()+" di kanan r
        parent.setKa(baru);
    }
}
else {
    System.out.println("NO DATA");
}
}
public float cariterbesar() {
    getMax(root);
    return maks;
}
public void getMax(Node root) {
    if(root.getKa()==null) {
        maks=root.getIsi();
        System.out.println("NILAI TERBESAR = "+maks);
    }
    else {
        maks = root.getIsi();
        temp = root.getKa();
        if(maks<temp.getIsi()) {
            getMax(temp);
        }
    }
}
}

```

Figure20 : maximum value tree code

5.1.2 Program with Scan Image

The program will be tested using the data which is signature image named "tes6.png". Firstly, the input image will go through the grayscale process, filtering, sharpening, tresholding and edge detection. The output image from edge detection process will be used as input in the identification process for calculating the value of the correlation coefficient.

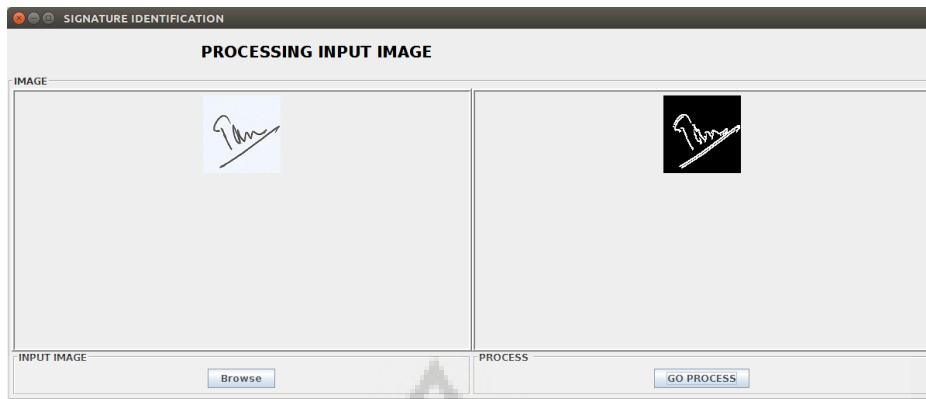


Figure21 : Test image process display

Each process will generate output images and stored in the "test image" folder. In addition each process generate the output image, it is also generates txt file that contains the pixel value.



Figure22 : Result process test image

To find out the validity of the results of each process so it can be done by calculated the process manually. For example, the pixel value is taken from image with the form of 3 x 3 matrix :

$$R = \begin{matrix} 75 & 178 & 231 \\ 254 & 235 & 254 \\ 242 & 249 & 241 \end{matrix}$$

$$G = \begin{matrix} 76 & 179 & 232 \\ 255 & 236 & 255 \\ 245 & 252 & 242 \end{matrix}$$

$$B = \begin{matrix} 78 & 181 & 234 \\ 255 & 240 & 255 \\ 250 & 255 & 247 \end{matrix}$$

First, the image will go through grayscale process i.e. with the formula :

$$S = \frac{r + g + b}{3}$$

The result of the grayscale process is :

$$\begin{array}{ccc} 76 & 179 & 232 \\ 254 & 237 & 254 \\ 245 & 252 & 243 \end{array}$$

For example is taken the first pixel value from every channel that is R = 75 , G = 76 and B = 78, so obtained :

$$\frac{75+76+78}{3} = 76$$

The next process is filtering with the formula :

$$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} / 9$$

The result of the filtering process is :

$$\begin{array}{ccc} 151 & 187 & 197 \\ 209 & 219 & 226 \\ 229 & 235 & 237 \end{array}$$

The pixel value that want to be changed is in the middle of 3x3 matrix, so obtained :

$$\frac{76 + 179 + 232 + 254 + 237 + 254 + 245 + 252 + 243}{9} = 219$$

The next process is sharpening with the kernel of 3x3 matrix :

$$G(x,y) : \begin{array}{ccc} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{array}$$

The result of the sharpening process is :

$$\begin{array}{ccc} 119 & 242 & 246 \\ 255 & 238 & 253 \\ 244 & 250 & 245 \end{array}$$

The pixel value that want to be changed is in the middle of 3x3 matrix, so obtained :

$$(-1.187) + (-1.209) + (5.219) + (-1.226) + (-1.235) = 238$$

The next process is tresholding with the provisions of :

$$F_0(x,y) = 0, f_1(x,y) < 128$$

$$F_0(x,y) = 255, f_1(x,y) \geq 128$$

The result of the tresholding process is :

$$\begin{array}{ccc} 0 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{array}$$

The next process is edge detection process with sobel method which has horizontal kernel and vertical kernel :

$$G_x : \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad G_y : \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

$$M = \sqrt{G_x^2 + G_y^2}$$

The result of the edge detection process is :

$$\begin{matrix} 255 & 255 & 0 \\ 255 & 255 & 0 \\ 0 & 0 & 0 \end{matrix}$$

The calculation for each kernel and the result :

$$\text{Horizontal} : 0+0+(1.255)+(-2.255)+0+(2.255)+(-1.255)+0+(1.255) \\ = 255$$

$$\text{Vertical} : 0+(2.255)+(1.255)+0+0+0+(-1.255)+(-2.255)+(-1.255) \\ = -255$$

$$S : \sqrt{255^2 + (-255)^2} = \sqrt{65025 + 65025} = \sqrt{130050} = 361,$$

$$361 > 255 = 255$$

After going through the process above, then can be performed to calculate the correlation coefficient value using the output image from edge detection process.

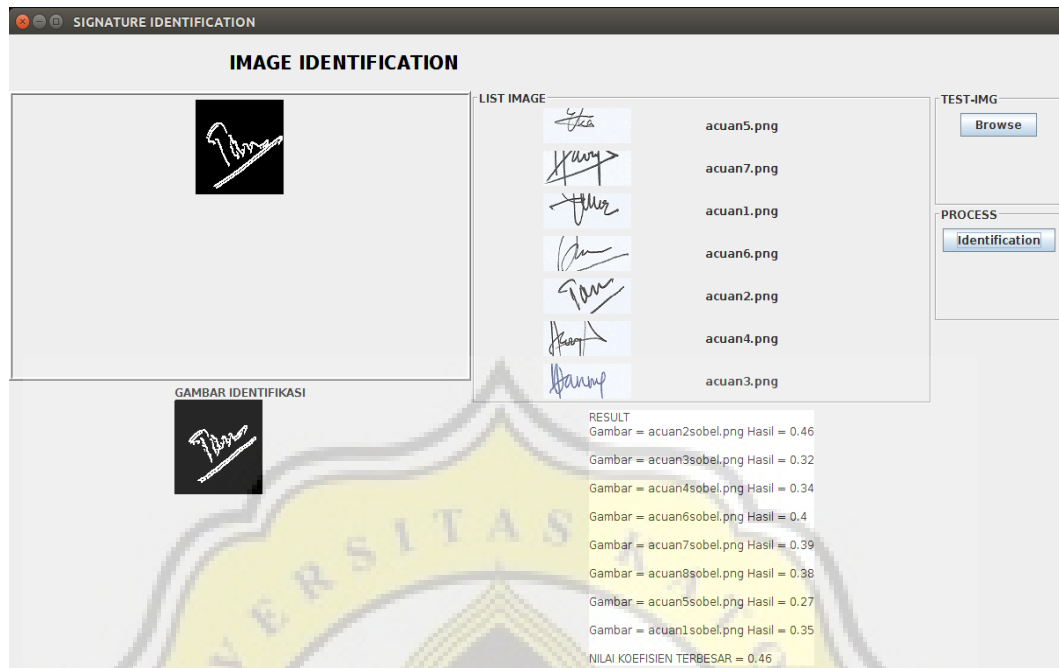


Figure23 : Identification test image display

The program will calculate the value of the correlation coefficient against each database image and display it in the program. At the same time the value of the correlation coefficient will be inserted in the binary tree data structure for later search the maximum value.

```

0.46
ROOT = 0.46
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.33
Tambah 0.33 di kiri node 0.46
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.34
Tambah 0.34 di kanan node 0.33
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.4
Tambah 0.4 di kanan node 0.34
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.39
Tambah 0.39 di kiri node 0.4
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.27
Tambah 0.27 di kiri node 0.33
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.35
Tambah 0.35 di kiri node 0.39
NILAI TERBESAR = 0.46

```

Figure24 : Insert tree test image

It can be known that the maximum value of the correlation coefficient is 0.46 with “acuan2sobel.png” image therefore the database image “acuan2.png” has similarity with the input image.

5.1.3 Program with Draw Image

Another testing is done by program using the image that is created by using the mouse in the program, the images that tested with a 100 x 100 size. The image is tes96.png.

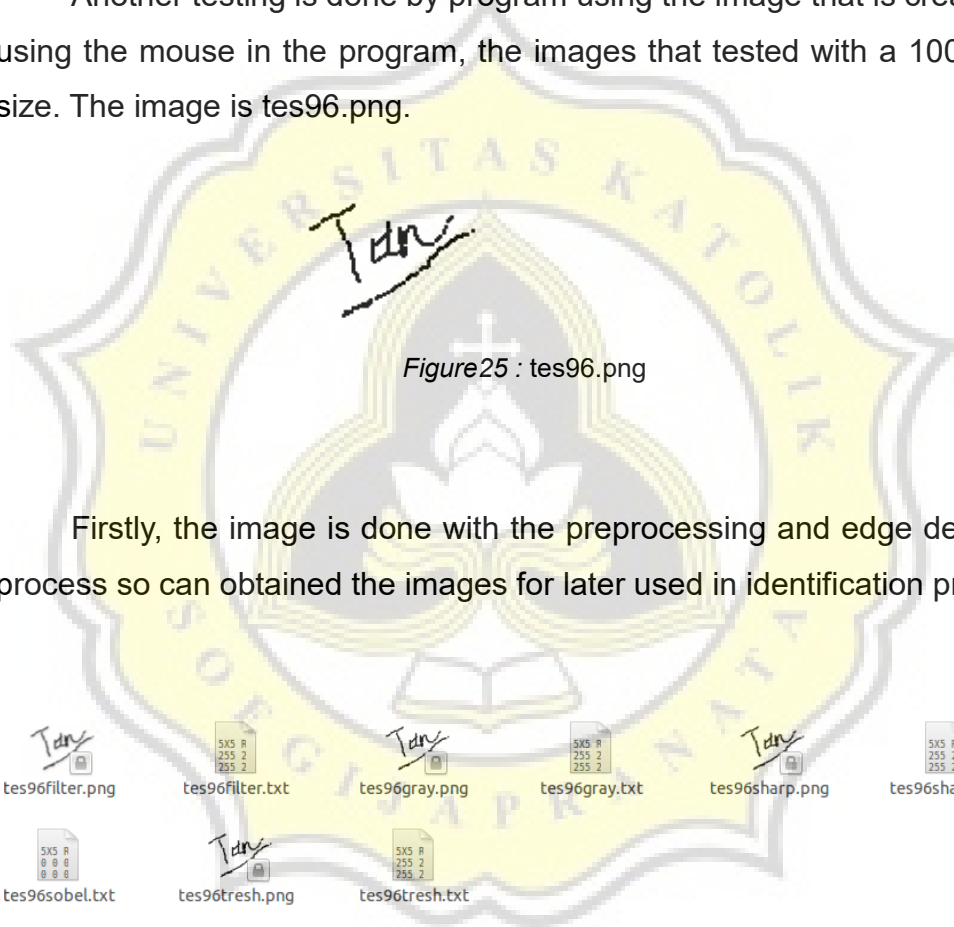


Figure25 : tes96.png

Firstly, the image is done with the preprocessing and edge detection process so can obtained the images for later used in identification process.

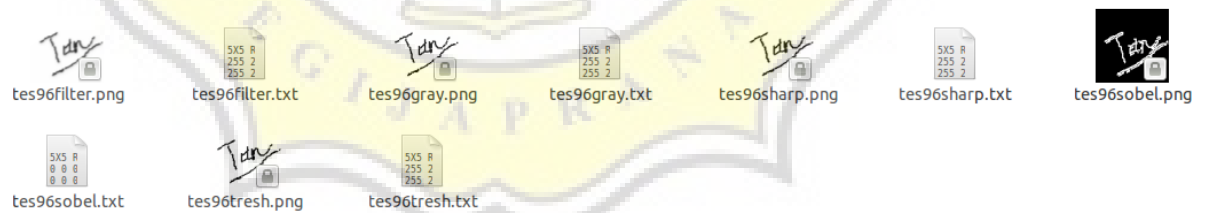


Figure26 : Tes96.png process

Each process generates output images and txt file contains the pixels value. The next step is inserted the output image from edge detection process to the identification process.

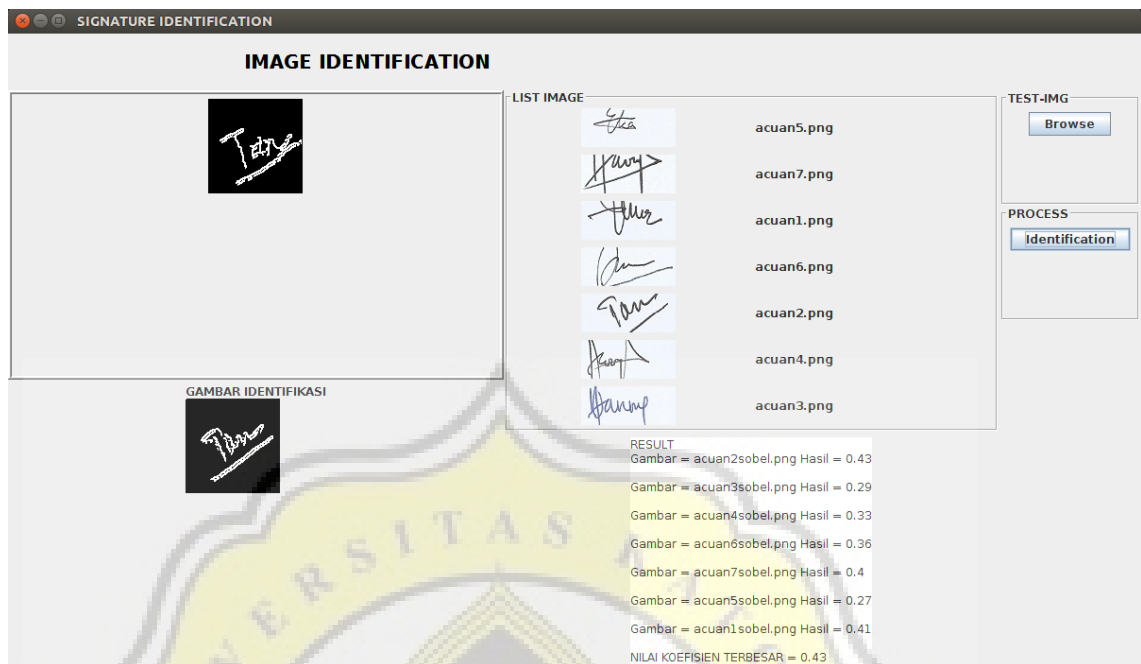


Figure27 : Tes96.png result

On the terminal can be known each value of the correlation coefficient that is inserted into the binary tree data structure.

```
0.43
ROOT = 0.43
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.31
Tambah 0.31 di kiri node 0.43
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.33
Tambah 0.33 di kanan node 0.31
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.36
Tambah 0.36 di kanan node 0.33
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.4
Tambah 0.4 di kanan node 0.36
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.27
Tambah 0.27 di kiri node 0.31
/home/harry/IMAGE/PROGRAMTA/dbgambar
0.41
Tambah 0.41 di kanan node 0.4
NILAI TERBESAR = 0.43
```

Figure28 : tes95.png tree

The results show the program can recognize properly the similarities between the reference image against test image namely tes96.png. On the tes96.png image the largest correlation coefficient value was 0.43 on the acuan2.png image.

5.2. Testing

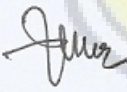
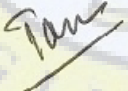
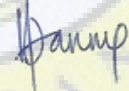
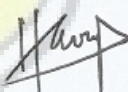

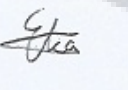

5.2.1 Testing Scan Image

Another testing was doing with 5 different images where each image is given 4 different conditions, namely:

1. normal image(size 100x100),
2. image given the scratches (100x100),
3. image size 200x200,
4. image size 200x200 and given the scratches.

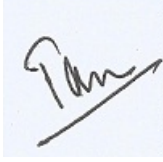

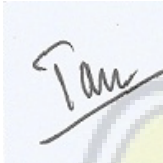
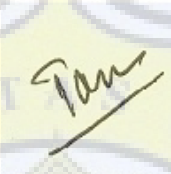
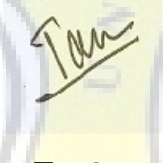
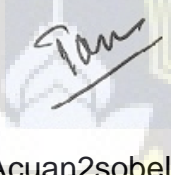
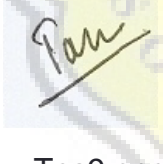
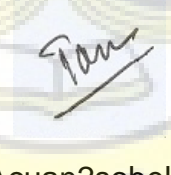
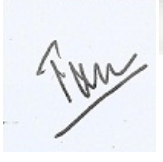
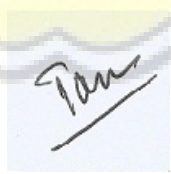
The list of database image used is :

Table1 : list database image

 Acuan1.png	 Acuan2.png	 Acuan3.png	 Acuan7.png
 Acuan4.png	 Acuan5.png	 Acuan6.png	

The images that used as database is 7 with each different signatures. These images will be used for knowing similarities with the test images.

Table2 : Normal image (100x100)

Input Image	Identification result	Koefisien Max
 Tes6.png	 Acuan2sobel.png	0.46
 Tes7.png	 Acuan2sobel.png	0.51
 Tes8.png	 Acuan2sobel.png	0.47
 Tes9.png	 Acuan2sobel.png	0.5
 Tes10.png	 Acuan2sobel.png	0.47

The results showed the program can recognize the signature image as much as 5 of 5 test images with the value of the correlation coefficient of varied between 0.46 until 0.51.

Table3 : Image given the scratches (100x100)

Input Image	Identification result	Koefisien Max
 Tes11.png	 Acuan2sobel.png	0.48
 Tes12.png	 Acuan2sobel.png	0.46
 Tes13.png	 Acuan2sobel.png	0.48
 Tes14.png	 Acuan2sobel.png	0.53
 Tes15.png	 Acuan2sobel.png	0.81

The results showed the program can recognize the signature image as much as 5 of 5 test images with the value of the correlation coefficient of varied between 0.46 until 0.81.

Table4 : Image size 200x200

Input Image	Identification result	Koefisien Max
 Tes16.png	 Acuan2sobel.png	0.46
 Tes17.png	 Acuan2sobel.png	0.53
 Tes18.png	 Acuan2sobel.png	0.45
 Tes19.png	 Acuan2sobel.png	0.54
 Tes20.png	 Acuan2sobel.png	0.83

The results showed the program can recognize the signature image as much as 5 of 5 test images with the value of the correlation coefficient of varied between 0.45 until 0.83.

Table5 : Image size 200x200 and given the scratches

Input Image	Identification result	Koefisien Max
 Tes21.png	 Acuan2sobel.png	0.53
 Tes22.png	 Acuan2sobel.png	0.44
 Tes23.png	 Acuan2sobel.png	0.46
 Tes24.png	 Acuan2sobel.png	0.52
 Tes25.png	 Acuan2sobel.png	0.76

The results showed the program can recognize the signature image as much as 5 of 5 test images with the value of the correlation coefficient of varied between 0.44 until 0.76.

From the testing results which uses 20 data of signatures images , 15 signatures images can be recognized while the other 5 are wrong with the details:

1. normal signature 5 sample, 5 correct,
2. normal signature is given a scratches 5 sample, 5 correct,
3. signature images size 200x200 5 sample, 5 correct,
4. signature images size 200x200 were given a scratches 5 sample, 5 correct

Therefore, the percentage of success is

$$\frac{20}{20} \times 100\% = 100\%$$

20

After that program tested with the rotation image to find out the value of the correlation coefficient and the result :

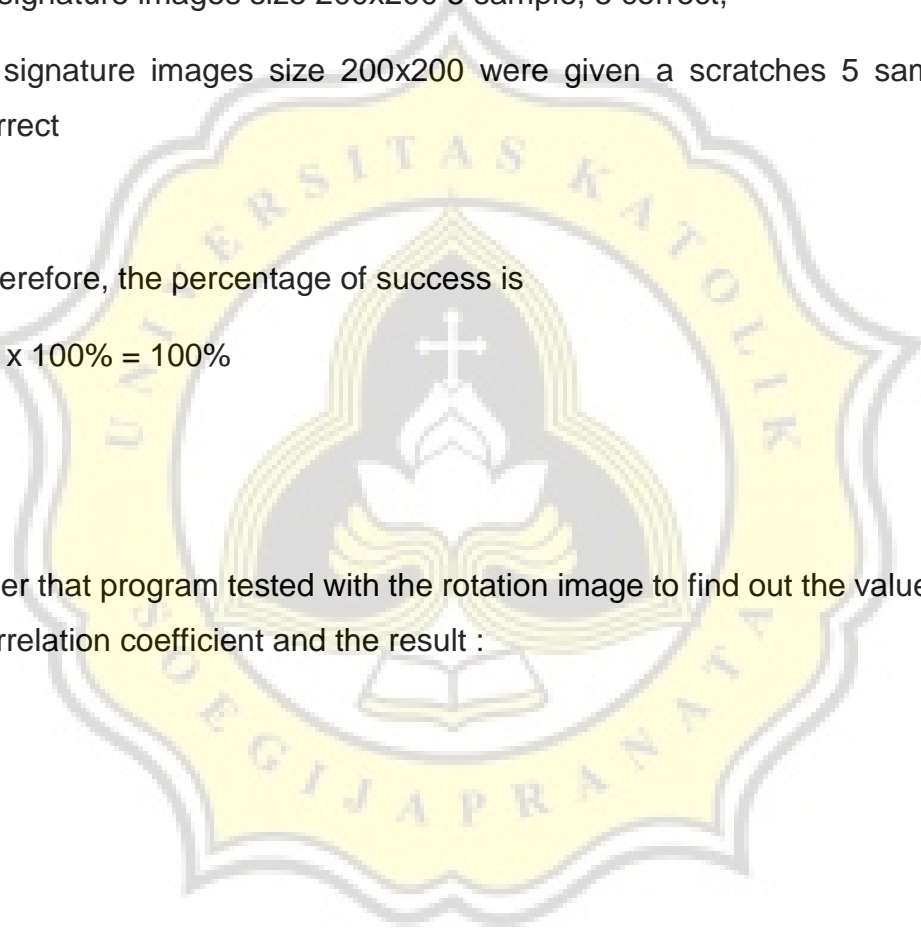



Table6 : rotation image 1

Image	Identification	Correlation Coefficient
 10 degree	 Acuan2sobel.png	0.6
 20 degree	 Acuan2sobel.png	0.5
 30 degree	 Acuan2sobel.png	0.46
 40 degree	 Acuan6sobel.png	0.41
 50 degree	 Acuan6sobel.png	0.39

The results showed the program can recognize the signature image as much as 3 of 5 test images with the value of the correlation coefficient of varied between 0.39 until 0.6.

Table7 : rotation image 2

Image	Identification	Correlation Coefficient
 10 degree	 Acuan7sobel.png	0.55
 20 degree	 Acuan7sobel.png	0.49
 30 degree	 Acuan7sobel.png	0.44
 40 degree	 Acuan7sobel.png	0.44
 50 degree	 Acuan7sobel.png	0.45

The results showed the program can recognize the signature image as much as 5 of 5 test images with the value of the correlation coefficient of varied between 0.45 until 0.55.

Table8 : rotation image 3

Image	Identification	Correlation Coefficient
 10 degree	 Acuan9sobel.png	0.62
 20 degree	 Acuan9sobel.png	0.53
 30 degree	 Acuan9sobel.png	0.5
 40 degree	 Acuan9sobel.png	0.48
 50 degree	 Acuan9sobel.png	0.45

The results showed the program can recognize the signature image as much as 5 of 5 test images with the value of the correlation coefficient of varied between 0.45 until 0.62.

Table9 : rotation image 4

Image	Identification	Correlation Coefficient
 10 degree	 Acuan10sobel.png	0.49
 20 degree	 Acuan10sobel.png	0.4
 30 degree	 Acuan9sobel.png	0.38
 40 degree	 Acuan4sobel.png	0.37
 50 degree	 Acuan4sobel.png	0.36

The results showed the program can recognize the signature image as much as 2 of 5 test images with the value of the correlation coefficient of varied between 0.36 until 0.49.

Test images are same with the database image but given a slope were rotated of 10-50 degrees and from the result signature with more scratches is more known and not affected by slope, meanwhile signature with less scratches still known with a slope of 10-30 degrees. The program can still recognize the image at the slope between 10-30 degrees because of the position of the signature is not moved too far from the original image. With a greater slope then the location of the scratches corresponding to the original image will be shifted and the program more difficult to recognize.

5.2.1 Testing Draw Image

Testing is also done for the draw image which is using the mouse to find out the similarities with the reference image. The images were tested as much as 5 and the here's the result :

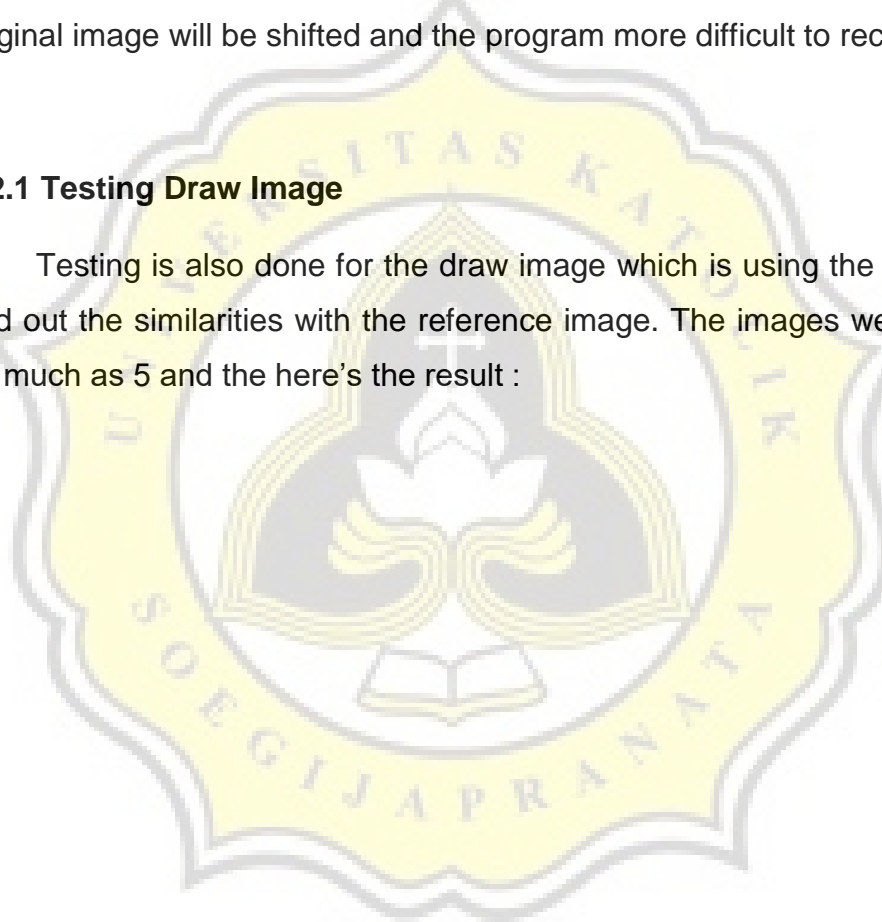

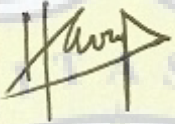


Table10: draw image

Image	Identification	Correlation Coefficient
 Tes94.png	 Acuan6sobel.png	0.43
 Tes95.png	 Acuan7sobel.png	0.45
 Tes96.png	 Acuan2sobel.png	0.43
 Tes97.png	 Acuan2sobel.png	0.49
 Tes98.png	 Acuan2sobel.png	0.4

From the results above obtained four images can be identified correctly while 1 image cannot identified correctly. At the time of drawing with the mouse, position of signature objects and the number of scratches made is very affects the results of the identification.

