CHAPTER V

IMPLEMENTATION AND TEST

5.1. Implementation

5.1.1. Implementation Program

The program have three display, i.e. main display, encryption display, and decryption display.

Figure 7: Main Program Display

The main display has two menu i.e. Encryption and Decryption. Then if the encryption button is pressed, encryption menu will appear. And also for decryption button. Before the process of encryption and decryption are performed, the input images must be processed first to grayscale, filtering, edge detection. After that the user can enter a message to be hidden into the image for the encryption process and for the decryption process could be to take the message in the image.

5.1.1.1. Initial Process

Because each process requires the output of another process, the process must run sequence i.e. grayscale, filtering, edge detection.

TampilanEnkrip.java × | TampilanDekrip.java × | stegaedge.java × | testdekripedge.java ×

```
164     class EdgeDetec implements ActionListener
165     {
166             public void actionPerformed(ActionEvent evt)
167             {
168                     //proses Grayscale
169                     grayscale eka = new grayscale();
170                     eka.ProsesGrayscale(namagambar);
171
172                     //proses Filter
173                     filter eka1 = new filter();
174                     eka1.ProsesFilter(eka.getNamaGrayscale());
175
176                     //proses Edgedetection
177                     edgedetection eka2 = new edgedetection();
178                     eka2.ProsesEdge(eka1.getNamaFilter());
179
180                     //mengambil nama gambar edge untuk dimasukkan ke label
181                     String tampungtepi = eka2.getNamaEdge();
182                     String c = eka2.getJumlah();
183                     jumlahpixel = eka2.getHitung();
184
185                     gambar2 = new ImageIcon(tampungtepi);
186                     img2.setIcon(gambar2);
187
188                     JOptionPane.showMessageDialog(null,c,"Notification",JOptionPane.INFORMATION_MESSAGE);
189             }
190     }
```

Figure 8: Initial Process

5.1.1.2. Grayscale

First step from this program is change the input image into grayscale image. Grayscale process is done by taking the average value of each channel red, green, blue at each pixel.

TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×

```java

        public void ProsesGrayscale(String opo)
        {
                String namagambar = opo;
                int a = namagambar.length();
                String pisah = namagambar.substring(0,a-4);
                System.out.println("Proses GrayScale");
                try {
                        File input = new File(namagambar);
                        image = ImageIO.read(input);
                        width = image.getWidth();
                        height = image.getHeight();

                        for(int i=0;i<height;i++)
                        {
                                for(int j=0 ; j<width; j++)
                                {
                                        Color c = new Color(image.getRGB(j,i));
                                        int total = (c.getRed()+c.getBlue()+c.getGreen())/3;

                                        Color newColor = new Color(total,total,total);
                                        image.setRGB(j,i,newColor.getRGB());
                                }
                        }
                        File output = new File(pisah+"gray.png");
                        ImageIO.write(image,"png",output);

                        hasilgambar = pisah+"gray.png";
                } catch(Exception e) {}
        }

        public String getNamaGrayscale()
        {
                return hasilgambar;
        }
}
```

Java ▾ Tab Width: 8 ▾ Ln 9, Col 8 INS

Figure 9: Grayscale code

5.1.1.3. Filtering

Next step after grayscale is filtering process. The filtering process requires the output of grayscale process. The filter used is mean filter by adding the pixel with neighbors around the pixel to be calculated and then divided by the sum of all the neighbors and the pixel itself.

```
for(int i=1; i<height-1 ; i++)
{
        for(int j=1 ; j<width-1 ; j++)
        {
                temp[0] = (pixel[i-1][j-1] * filter[0][0]);
                temp[1] = (pixel[i-1][j] * filter[0][1]);
                temp[2] = (pixel[i-1][j+1] * filter[0][2]);

                temp[3] = (pixel[i][j-1] * filter[1][0]);
                temp[4] = (pixel[i][j] * filter[1][1]);
                temp[5] = (pixel[i][j+1] * filter[1][2]);

                temp[6] = (pixel[i+1][j-1] * filter[2][0]);
                temp[7] = (pixel[i+1][j] * filter[2][1]);
                temp[8] = (pixel[i+1][j+1] * filter[2][2]);


                int total = (temp[0]+temp[1]+temp[2]+temp[3]+temp[4]+temp[5]+temp[6]+temp[7]+temp[8])/9;

                if(total > 255)
                {
                        total = 255;
                }
                pixelhasil[i][j] = total;
        }
}
```

Figure 10: Filter Code

5.1.1.4. Edge Detection

After the filter process is completed, the next step is edge detection process. Edge detection process requires the output of filtering process. This program uses sobel edge detection method. Pixel to be calculated is taken as well as with its neighbors form a 3x3 matrix multiplied by kernel Gx and Gy.

TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×

```
                int total = (temp[0]+temp[1]+temp[2]+temp[3]+temp[4]+temp[5]+temp[6]+temp[7]+temp[8]);
                if(total < 0 )
                {
                        total = total * -1;
                }

                temp2[0] = (pixel[i-1][j-1] * gy[0][0]);
                temp2[1] = (pixel[i-1][j] * gy[0][1]);
                temp2[2] = (pixel[i-1][j+1] * gy[0][2]);

                temp2[3] = (pixel[i][j-1] * gy[1][0]);
                temp2[4] = (pixel[i][j] * gy[1][1]);
                temp2[5] = (pixel[i][j+1] * gy[1][2]);

                temp2[6] = (pixel[i+1][j-1] * gy[2][0]);
                temp2[7] = (pixel[i+1][j] * gy[2][1]);
                temp2[8] = (pixel[i+1][j+1] * gy[2][2]);

                int total2 = (temp2[0]+temp2[1]+temp2[2]+temp2[3]+temp2[4]+temp2[5]+temp2[6]+temp2[7]+temp2[8]);
                if(total2 < 0 )
                {
                        total2 = total2 * -1;
                }

                int baru = total + total2;
                if(baru > 255)
                {
                        baru = 255 ;
                        //hitung++;
                }

                pixelhasil[i][j] = baru;
            }
        }
```

Java ▾ Tab Width: 8 ▾ Ln 177, Col 31 INS

Figure 11: Edge Detection code

5.1.1.5. Threshold

After getting a new pixel value of the edge detection process, the final step is to do a threshold process. If the pixel value is less than 128 then changed to 0, if more than 128 then changed to 255.

TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×

```
123                     }
124             }
125
126             //proses threshold pixelhasil
127             for(int i = 1 ; i < height-1 ; i++)
128             {
129                     for(int j = 1 ; j < width-1 ; j++)
130                     {
131                             if(pixelhasil[i][j] >= 128)
132                             {
133                                     pixelhasil[i][j] = 255;
134                                     hitung++;
135                             }
136                             else
137                             {
138                                     pixelhasil[i][j] = 0;
139                             }
140                     }
141             }
142
143             for(int i = 0 ; i < height ; i++)
144             {
145                     for(int j = 0 ; j < width ; j++)
146                     {
147                             Color newColor = new Color(pixelhasil[i][j],pixelhasil[i][j],pixelhasil[i][j]);
148                             image.setRGB(j,i,newColor.getRGB());
149                     }
150             }
151
152                             File output = new File(pisah+"sobel.png");
153                             ImageIO.write(image,"png",output);
154
155                             System.out.println("Jumlah pixel tepi(pixel 255) : "+hitung);
156                             System.out.println("Jumlah huruf yang bisa ditampung "+hitung/8);
157
158                             jumlahpixel = "Jumlah pixel tepi(pixel 255) : "+hitung+"\n"+"Jumlah huruf yang bisa ditampung
```

Java ▾ Tab Width: 8 ▾ Ln 177, Col 31 INS

Figure 12: Threshold code

5.1.1.6. Steganography

After all the process is completed the next step is steganography. The user must enter the message to hide into the picture. After that changed the message into binary form.

```
char[] huruf = pesan.toCharArray();
int[][] binerpesan = new int[pesan.length()][8];

int[][] pixeltepi = new int[jumlahpixel][2];

for(int i = 0 ; i < huruf.length ; i++)
{
        desimal = (int) huruf[i];
        binerpesan[i][0] = 0;
        for(int j = 7 ; j >= 1 ; j--)
        {
                a = desimal % 2 ;
                binerpesan[i][j] = a;
                desimal = desimal / 2 ;
        }
}
pixel = pesan.length()*8;
//mencetak matriks pesan dalam biner
for(int i = 0 ; i < huruf.length ; i++)
{
        for(int j = 0 ; j <8 ; j++)
        {
                System.out.print(binerpesan[i][j]+" ");
        }
        System.out.println();
}
```

Figure 13: Change Message into binary code

After it, determines the starting point on the edge pixel in a random way as a starting point to hide a message. After getting the starting point, hide the first bit of pixel message

```
TampilanEnkrip.java × | TampilanDekrip.java × | stegaedge.java × | testdekripedge.java × | grayscale.java × | edgedetection.java × | filter.java ×

168
169                     //random posisi dr matrix
170                     Random randeka = new Random();
171                     int r = randeka.nextInt(jumlahpixel);
172
173                     System.out.println("Pixel start : "+pixeltepi[r][0]+" "+pixeltepi[r][1]);
174
175                     //proses menyisipkan pesan kedalam gambar
176                     int loop = r;
177                     BufferedWriter tuliss = new BufferedWriter(new FileWriter("filetxt/prosesEnkripsi"+pisah+".txt",true));
178                     tuliss.append("\nPixel Red Awal(Enkripsi) "+namagambar+" : \n");
179
180                     //proses penyisipan pixel pertama
181                     Color d = new Color(asli.getRGB(pixeltepi[loop][1],pixeltepi[loop][0]));
182
183                     tampungasli = d.getRed();
184                     tampungblue = d.getBlue();
185                     System.out.print(tampungblue+",");
186
187                     //proses masuk ke txt
188                     tuliss.append("("+pixeltepi[loop][0]+","+pixeltepi[loop][1]+")="+tampungasli);
189
190                     //untuk mengubah pixel menjadi biner
191                     for(int b = 7 ; b >=0 ; b--)
192                     {
193                             isi = tampungasli % 2;
194                             binerpixel[b] = isi;
195                             tampungasli = tampungasli / 2;
196
197                             isibiru = tampungblue % 2;
198                             binerbiru[b] = isibiru;
199                             tampungblue = tampungblue / 2;
200                     }
201
202
203                     //proses menyisipi pesan

Java ▾   Tab Width: 8 ▾   Ln 169, Col 51   INS
```

Figure 14: Random Pixel Code

The process will go on until the last bit of the message is hidden. Messages can be stored is also limited depending on the number of pixel edges of the image, because one letter is stored in 8 pixel. Binary messages are stored in a red layer of the pixel, to signal that the pixel contains the message or not, then the blue layer also converted into binary form. If on red layer there is a message the last bit of the blue layer was changed to 0 if there is no message then change last bit of the blue layer to 1. After all the pixels are processed and then change into the image again.

Here the illustration hiding the message in the edge pixel of image :

Original Image Image Threshold

Figure 15: Illustration Hiding the message

The original image will be processed until threshold process. After the threshold image is generated, the messages that have been converted into binary inserted at the pixel edge start from random pixel. As shown above, the message will be inserted on the red line.

TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×

```
245                     if(count < pixel)
246                     {
247                             tuliss.append("("+pixeltepi[loop][0]+","+pixeltepi[loop][1]+")="+tampungasli);
248
249                             //untuk mengubah pixel menjadi biner
250                             for(int b = 7 ; b >=0 ; b--)
251                             {
252                                     isi = tampungasli % 2;
253                                     binerpixel[b] = isi;
254                                     tampungasli = tampungasli / 2;
255
256                                     isibiru = tampungblue % 2;
257                                     binerbiru[b] = isibiru;
258                                     tampungblue = tampungblue / 2;
259                             }
260
261                             //proses menyisipi pesan
262                             binerpixel[7] = binerpesan[x][y];
263                             binerbiru[7] = 0;
264
265                             y++;
266                             if(y == 8)
267                             {
268                                     y=0;
269                                     x++;
270                             }
271
272                             //proses pengembalian biner pixel ke angka desimal
273                             jumlah = 0;
274                             jumlahbiru = 0;
275                             pangkat = 7;
276                             for(int k = 0 ; k < 8 ; k++)
277                             {
278                                     total = (binerpixel[k]*(Math.pow(2,pangkat))) ;
279                                     jumlah = jumlah + total;
280
281                                     totalbiru = (binerbiru[k]*(Math.pow(2,pangkat))) ;
```

Saving file '/home/eka/ProjectTA/stegaedge.java'... Java ▾ Tab Width: 8 ▾ Ln 262, Col 41 INS

Figure 16: Transform pixels into binary and Hiding Message code

5.1.1.7. Decryption

In the decryption process, the first step is like encryption process that is changing the input image into grayscale, filter, edge detection. After

getting the edge detection image, count the number of edges pixel and insert the coordinate into the array.

```
TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×
75              double jumlah ;
76              double total ;
77
78              //hitung pixel tepi
79              for(int i = 1 ; i < height-1 ; i++)
80              {
81                      for(int j = 1 ; j < width-1 ; j++)
82                      {
83                              Color asd = new Color(edge.getRGB(j,i));
84                              if(asd.getRed() == 255)
85                              {
86                                      hitungjumlah++;
87                              }
88                      }
89              }
90
91              //buat array matriks koordinat pixel tepi
92              int[][] pixeltepi = new int[hitungjumlah][2];
93              int ee=0;
94              for(int i = 1 ; i < height-1 ; i++)
95              {
96                      for(int j = 1 ; j < width-1 ; j++)
97                      {
98                              Color def = new Color(edge.getRGB(j,i));
99                              if(def.getRed() == 255)
100                             {
101                                     pixeltepi[ee][0] = i;
102                                     pixeltepi[ee][1] = j;
103                                     ee++;
104                             }
105                     }
106             }
107
108             BufferedWriter tulis = new BufferedWriter(new FileWriter("filetxt/prosesDekripsi"+namatok+".txt",true));
109             tulis.append("\nPixel Red Stegano(Dekripsi) "+namagambar+" : \n");
110
Java ▾   Tab Width: 8 ▾   Ln 48, Col 1   INS
```

Figure 17: Count the edge pixel code (Decryption)

After that the program will find the coordinates pixel of the beginning message by means of random. Then checked whether the coordinates of the starting point or not by the way: if the coordinates before on green pixel has last binary value of one, it means the starting point of a pixel message, if not then the program will search again by random until you find it.

```
TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×
116                 while(status!=1)
117                 {
118                         //random posisi dr matrix
119                         Random randeka = new Random();
120                         int r = randeka.nextInt(hitungjumlah);
121                         looping = r ;
122
123                         Color c = new Color(image.getRGB(pixeltepi[looping][1],pixeltepi[looping][0]));
124
125                         tampungbiru = c.getBlue();
126
127                         for(int x = 7 ; x >=0 ; x--)
128                         {
129                                 a = tampungbiru % 2;
130                                 tampungbiru = tampungbiru / 2 ;
131                                 binerbiru[x] = a;
132                         }
133
134                         if(binerbiru[7] == 0)
135                         {
136                                 System.out.print("Pixel Pesan,");
137                                 Color cek = new Color(image.getRGB(pixeltepi[looping-1][1],pixeltepi[looping-1][0]));
138                                 tampungcek = cek.getBlue();
139                                 for(int abc = 7 ; abc >=0 ; abc--)
140                                 {
141                                         a = tampungcek % 2;
142                                         tampungcek = tampungcek / 2 ;
143                                         binercek[abc] = a;
144                                 }
145                                 if(binercek[7] == 1)
146                                 {
147                                         System.out.println("Yes Pixel Start Pesan");
148                                         startpixel = looping;
149                                         status = 1 ;
150                                 }
151                                 else
                                                                                Java ▾  Tab Width: 8 ▾   Ln 48, Col 1   INS
```

Figure 18: Search coordinate code

Then begin the process of taking the message from the image.

```
TampilanEnkrip.java × TampilanDekrip.java × stegaedge.java × testdekripedge.java × grayscale.java × edgedetection.java × filter.java ×
166                         Color cari = new Color(image.getRGB(pixeltepi[startpixel][1],pixeltepi[startpixel][0]));
167
168                         tampung = cari.getRed(); //pixel red gambar stegano
169
170                         //proses masuk ke txt
171                         tulis.append("("+pixeltepi[startpixel][0]+","+pixeltepi[startpixel][1]+")="+tampung+"    ");
172
173                         //mengubah pixel dari gambar stega kedalam biner dan mengambil bit terakhir
174                         for(int x = 7 ; x >=0 ; x--)
175                         {
176                                 a = tampung % 2;
177                                 tampung = tampung / 2 ;
178                                 biner[x] = a;
179                         }
180                         temp[z] = biner[7];
181                         z++;
182
183                         if(z==8)
184                         {
185                                 //proses pengembalian pesan
186                                 pangkat = 7;
187                                 jumlah = 0;
188                                 total = 0 ;
189
190                                 for(int p = 0 ; p < 8 ; p++)
191                                 {
192                                         total = (temp[p]*(Math.pow(2,pangkat)));
193                                         jumlah = jumlah + total;
194                                         pangkat--;
195                                 }
196                                 curr = (char) jumlah;
197
198                                 kata = kata + curr;
199                                 z=0;
200                         }
201                         startpixel++;
                                                                                Java ▾  Tab Width: 8 ▾   Ln 48, Col 1   INS
```

Figure 19: Decryption Code

5.1.2. Program using Image

Encryption :

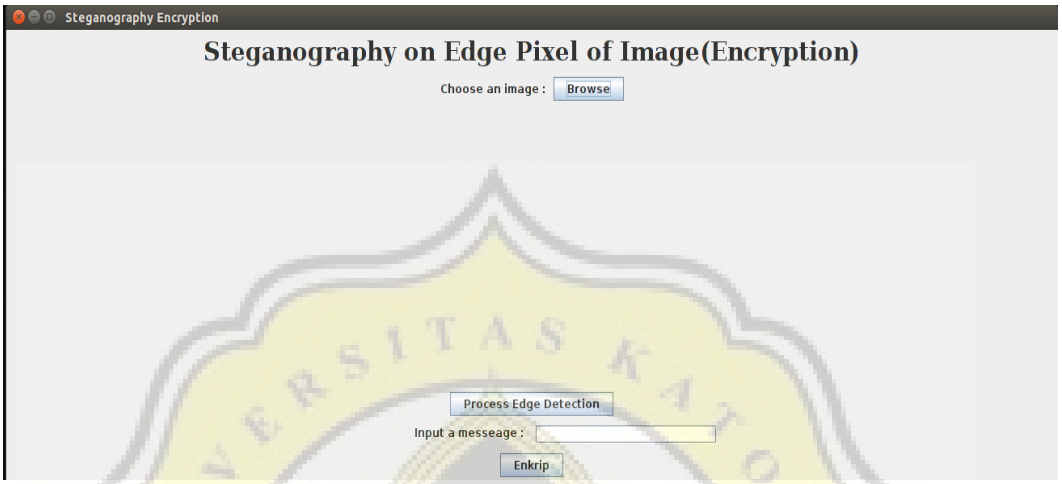


Figure 20: Main Display Encryption

The user must insert the image as a place to hide messages.

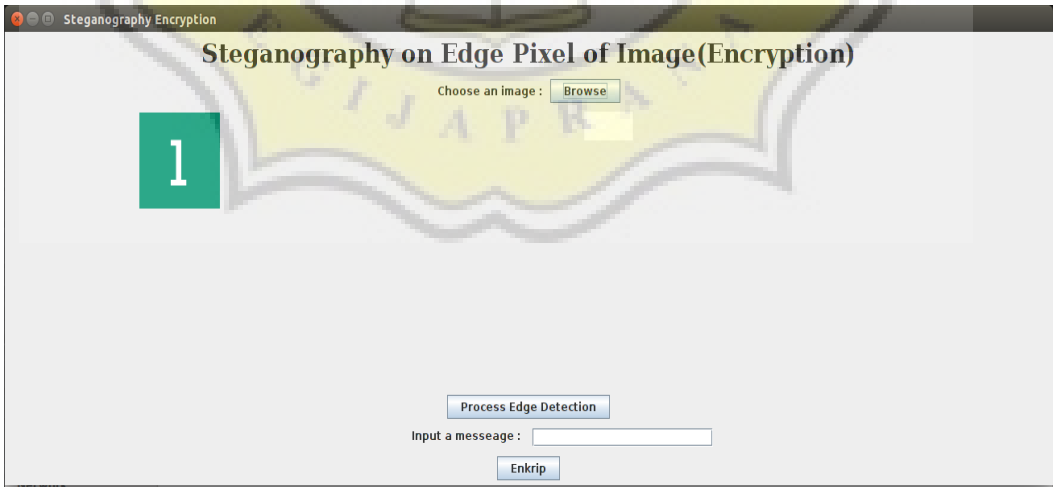


Figure 21: Image input Encryption

The image will be processed to grayscale, filter, and edge detection. Output from the image edge detection will be used as a reference for hiding messages.

First Process is grayscale. The process of summing the three kernels(red,green,blue) of one pixel and then divided by three to get grayscale value.

filter.java × grayscale.java × testdekripedge.java × pixelangka1.txt ×

1	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
2	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
3	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
4	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
5	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
6	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
7	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
8	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
9	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
10	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
11	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
12	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
13	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
14	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
15	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117
16	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117	8,153,117

Figure 22: Initial Pixel

This is the data for beginning pixel, then it will be processed to grayscale. Grayscale pixel calculation results will be :

1	92	92	92	92	92	92	92	92
2	92	92	92	92	92	92	92	92
3	92	92	92	92	92	92	92	92
4	92	92	92	92	92	92	92	92
5	92	92	92	92	92	92	92	92
6	92	92	92	92	92	92	92	92
7	92	92	92	92	92	92	92	92
8	92	92	92	92	92	92	92	92
9	92	92	92	92	92	92	92	92
10	92	92	92	92	92	92	92	92
11	92	92	92	92	92	92	92	92
12	92	92	92	92	92	92	92	92
13	92	92	92	92	92	92	92	92
14	92	92	92	92	92	92	92	92
15	92	92	92	92	92	92	92	92
16	92	92	92	92	92	92	92	92

Figure 23: Grayscale Pixel

Image Result of Grayscale :

After grayscale process is done next step is filtering. Filtering need the output from grayscale process. The filter used is mean filter 3x3 matrix. The process is by adding the pixel with neighbors around the pixel to be calculated and then divided by the sum of all the neighbors and the pixel itself. So the input pixel for filtering process is grayscale pixel above. The result of filter process is :

1	92	92	92	92	92	92	92	92
2	92	92	92	92	92	92	92	92
3	92	92	92	92	92	92	92	92
4	92	92	92	92	92	92	92	92
5	92	92	92	92	92	92	92	92
6	92	92	92	92	92	92	92	92
7	92	92	92	92	92	92	92	92
8	92	92	92	92	92	92	92	92
9	92	92	92	92	92	92	92	92
10	92	92	92	92	92	92	92	92
11	92	92	92	92	92	92	92	92
12	92	92	92	92	92	92	92	92
13	92	92	92	92	92	92	92	92
14	92	92	92	92	92	92	92	92
15	92	92	92	92	92	92	92	92
16	92	92	92	92	92	92	92	92

Figure 24: Filter Pixel

Image Result of Filtering :

The next process is edge detection with sobel method. Sobel method has two kernel I.e Gx and Gy.

$$Gx : \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad Gy : \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

M = |Gx| + |Gy|

The last process is threshold with range :

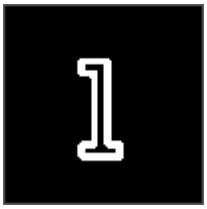
If pixel >= 128 then change to 255

if pixel < 128 then change to 0

1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0

Figure 25: Threshold pixel

Image result of threshold process :



After completing the process until the threshold, then go to the process of steganography.

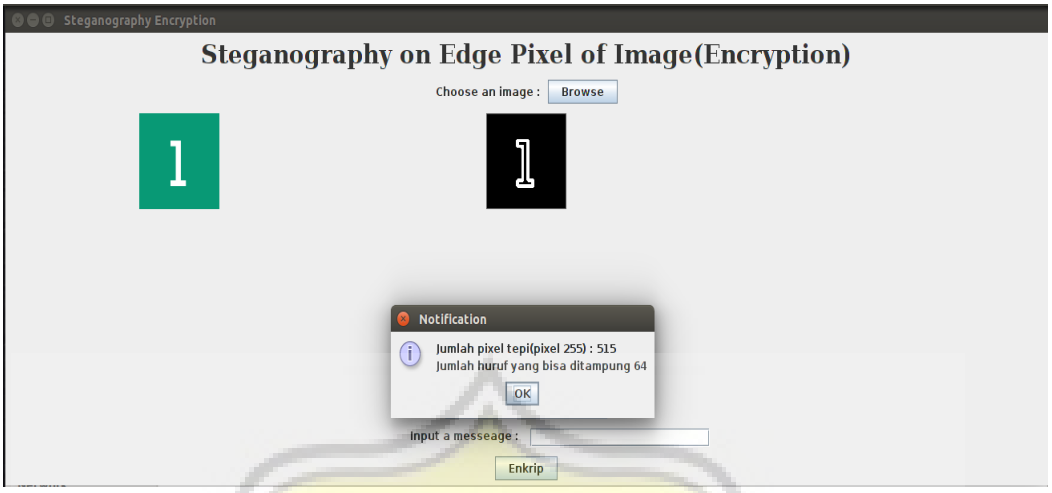


Figure 26: Initial Process in Program

After the image has done processing until threshold, user must input the message to be hidden. For examples message to be hidden is : “eka prayogo thedy”

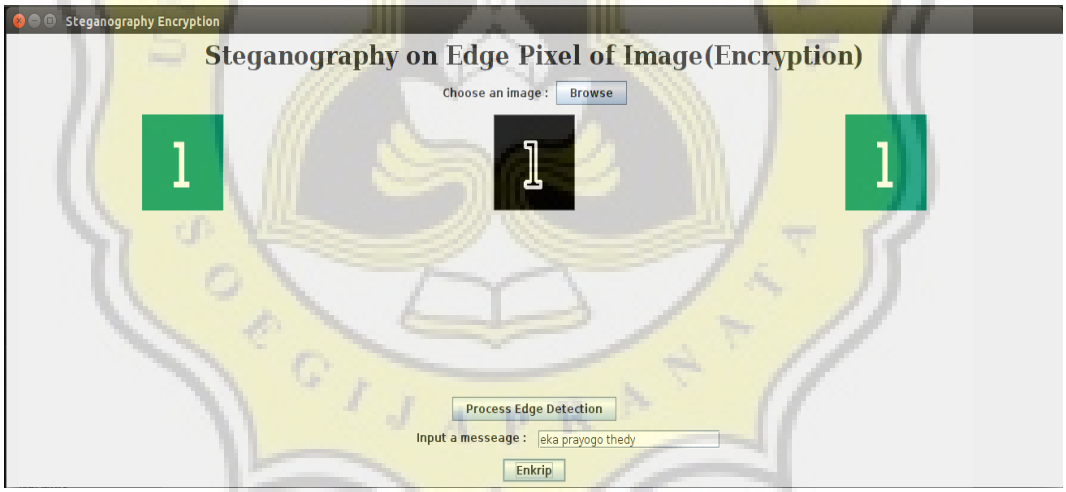


Figure 27: Steganography Process in Program

The program will hide the message based on edge pixel from threshold image output. After the message hidden in the picture, the program will create image that similar to original image but contain the messages.

And now for the decryption process is the same as the encryption process. User must input the image that contains messages.

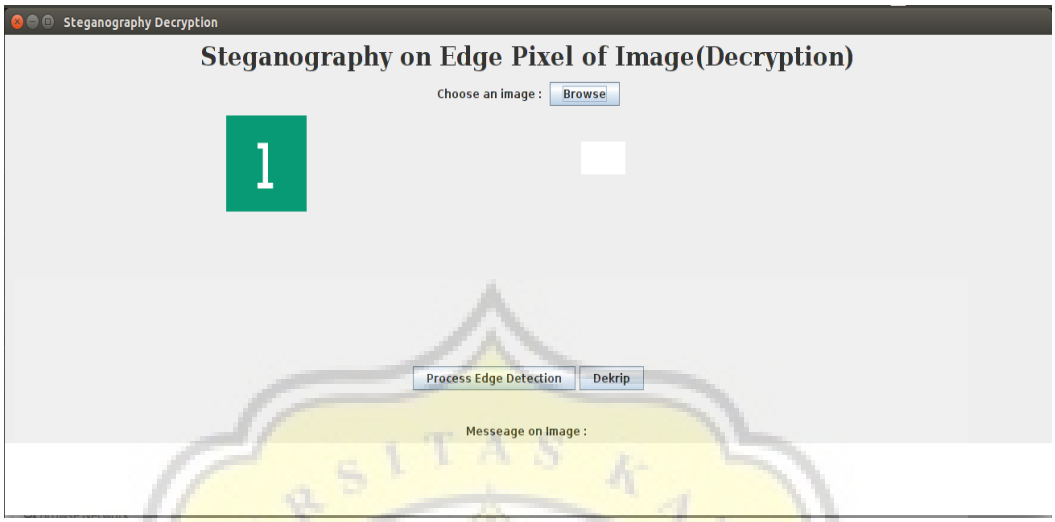


Figure 28: Image input Decryption

After that the image will go through grayscale, filtering, edge detection, and threshold process.

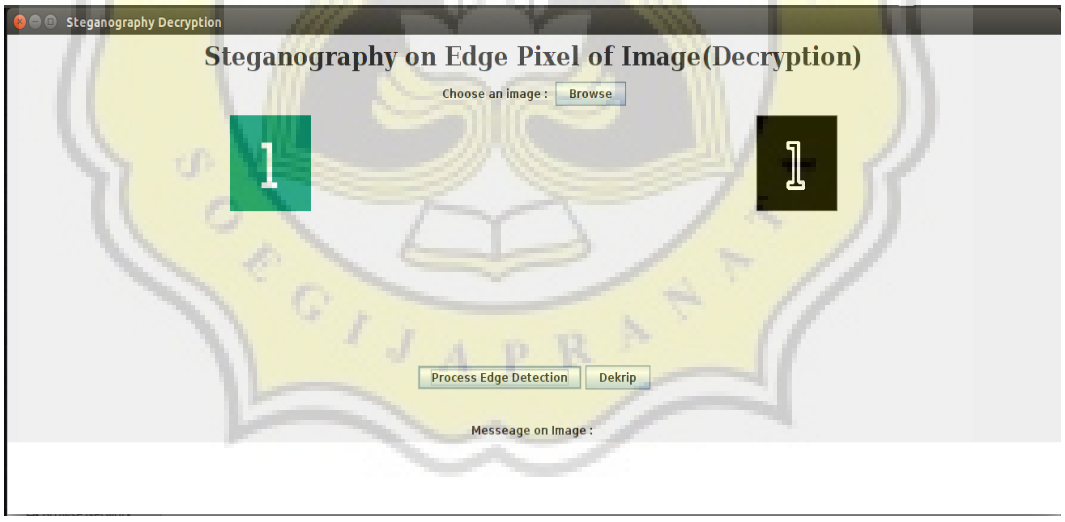


Figure 29: Initial Process (Decryption)

After the initial process is completed, the program will find the message hidden in the pixel based on edge pixel from the threshold image output. After getting the message on the image, the program will display the results.

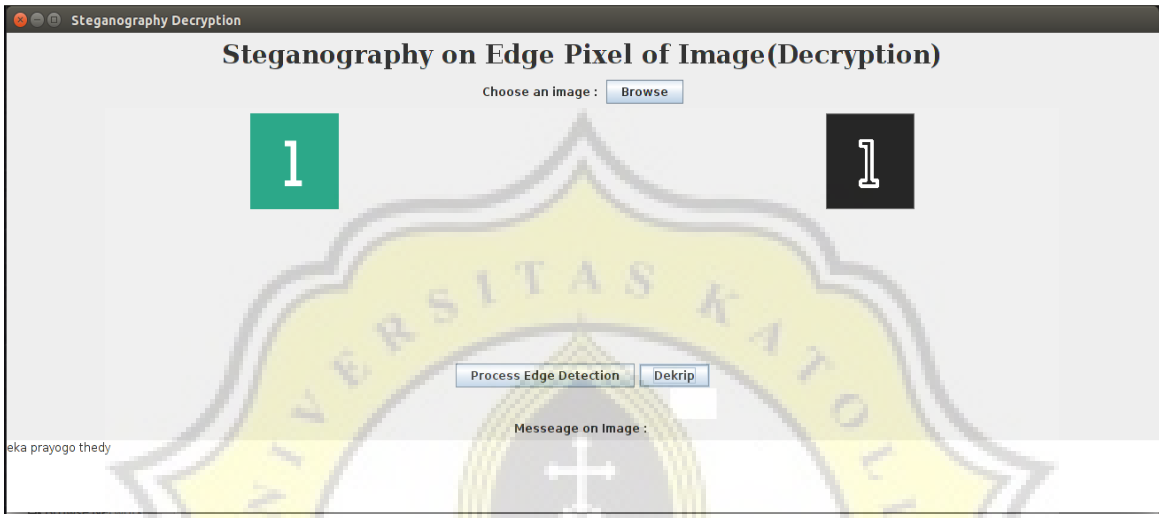
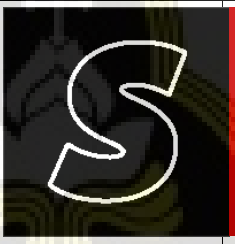


Figure 30: Decryption Process

5.2. Testing Program

5.2.1. Testing using image that have background red with 2 colors

Table 1 : Testing using gambars.png

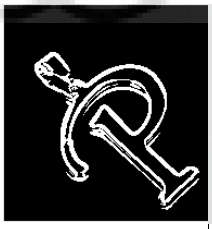
Image	Messages	Image Edge Detection	Image Steganography	Decode Messages
	ikom			ikom
	ikom unika			ikom unika
	ilmu komputer unika soegijaprana ta			ilmu komputer unika soeghÔÂà äÂÜÂè
	eka prayogo thedy teknik informatika 2013			eka prayogo thedy teknik informatikb @d

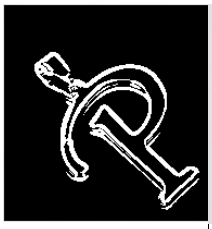
	balonku ada lima rupa rupa warnanya hijau kuning kelabu merah muda dan biru meletus balon hijau			balonku ada lima rupa rupa warnanya hihÂê@Öê ÜÒÜÎ@Ö ÊØÂÄê@ ÚÊäÂÐ@ ÚêÈÂ@È ÂÜ@ÄÒä ê@ÚÊØÊ èêæ@ÄÂ ØÞÜ@ÐÒ ÔÂ

From the results of the test above using pictures gambars.png with 5 different message, 2 messages successfully decoded while the other 3 failed to decoded. Shorter message more effectively to be hidden in the gambars.png, because the difference between pixel original with pixel results are not much different.

5.2.2. Testing using image that have background gray with 2 colors

Table 2: Testing using gambarp.png

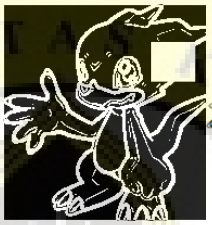
Image	Messages	Image Edge Detection	Image Steganography	Decode Messages
	teknik informatika			teknik inLÞäÚÂèÒÖ

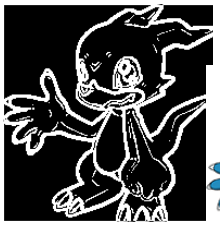
	eka prayogo thedy			eka prayogo thedy
	balonku ada lima rupa rupa warnanya hijau kuning kelabu merah muda dan biru			balonku@ÂÈÂ@Ø¥Û„• ÉÕÁ„□ÉÕÁ„□Ý…ÉÜÂÜòÂ@ÐÒÔÂê@ÖêÜÒÜÎ@ÖÊØ
	ilmu komputer teknik informatika unika			ilmu kompê
	semarang memiliki bangunan tugu muda untuk memperingati pertempuran 5 hari di semarang			semarang memiliki bangunan tugu

From the results above 1 message successfully decoded while 4 others failed. This is because the edge image results from encode and decode processes have different amounts. For example edge image from the encode process has a 100 pixel edge while the edge image from pixel edge decode process have more than 100 and even can be less than 100.

5.2.3. Testing using image that have background white with several colors

Table 3: Testing using Veemon.png

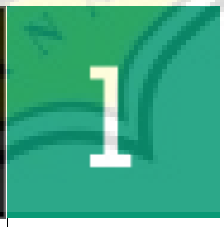
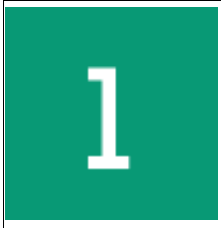
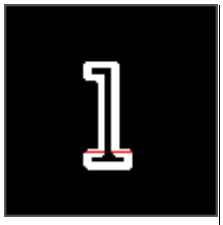
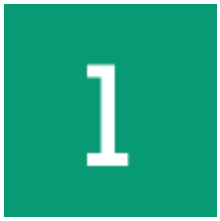
Image	Messages	Image Edge Detection	Image Steganography	Decode Messages
	eka prayogo thedy			eka prayogo tĐÊÈ
	unika soegijapranata semarang			unika soegijapranata semarang
	ilmu komputer teknik informatika unika soegijapranata semarang			ilmu komputer teknik informatika unika soegijapranata semarang
	besok jangan lupa meeting dengan klien penting jam 8 di kantor			besok jangan lupa meeting dengan klien@àÊÜè ÒÜÎ@ÔÂÚ@ p@ÈÒ@ÖÂ ÜèÞ

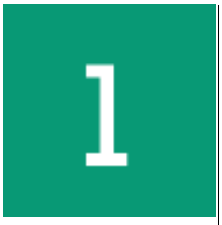
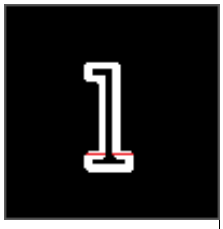
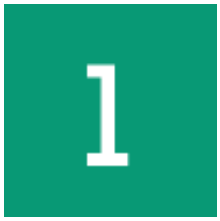
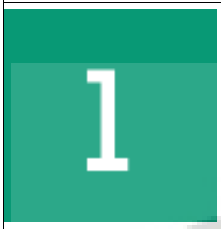
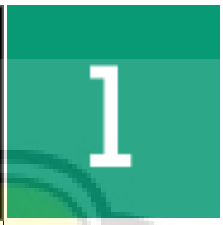
	you've been struggling to make things right That's how a superhero learns to fly			yotNìÊ@ÄÊÊÜ@æèäêÎÎØÒÜÎ@èÞ@ÚÂÖÊ@èÐå¹•Ì• É¥• ¡Ð• Q¡…Ð• Ì• ¡½Ü•

From testing using image Veemon.png, 2 messages successfully decoded while 3 messages failed to decoded, because there are differences in the number of edge pixels generated from the encoding process and decoding process so that the order took pixel which accommodate messages become incompatible.

5.2.4. Testing using image that have background green with 2 color

Table 4: Testing using angka1.png

Image	Messages	Image Edge Detection	Image Steganography	Decode Messages
	eka prayogo thedy			eka prayogo thedy
	semarang ibu kota jawa tengah			semarang ibu kota jawa tengah

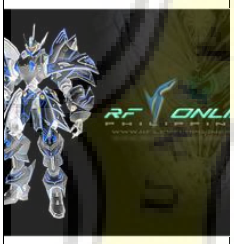
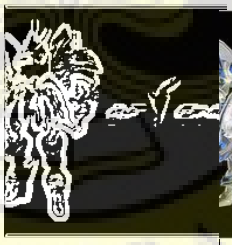
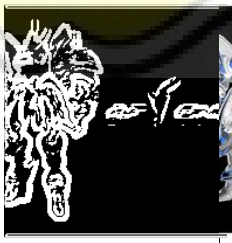
	libur natal dan tahun baru sebentar lagi tiba			liDêä@ÜÂè ÂØ@ÈÂÜ@ èÂÐêÜA‰ …ÉÔ□Í•‰•¹ Ñ…È□±…□ ¤□Ñ¥‰
	teknik informatika unika soegijapranata semarang			teknik informatika unika soegijapranata æÊÚÂäÂÜ
	Garuda pancasila akulah pendukung mu patriot proklamasi			Garuda pancasila akulah pendukung mu patriot prnÖØÂÚÂ æ

From testing using image gambar1.png, two messages successfully decoded while three others failed to decoded. This is because the edge image results from encode and decode processes have different amounts. For example edge image from the encode process has a 100 pixel edge while the edge image from pixel edge decode process have more than 100 and even can be less than 100.

5.2.5. Testing using image that have background dark with many colors

Table 5: Testing using acc.png

Image	Messages	Image Edge Detection	Image Steganography	Decode Messages

	eka prayogo thedy			eka prazÞÎÞ@thedy
	ikom unika soegijapranata			ikom unika soegijapranata
	bangun pemuda pemudi indonesia			bangun pemuda pemudi indonesia
	pesan ini sudah tersembunyi didalm gambar robot			pesan ini sudah tersembunyi didalm gambar robot
	tuju belas agustus tahun empat lima itulah hari kemerdekaan kita			tuju belas agustus tahun@ÊÚà

From testing using image acc.png, three messages successfully decoded while two others failed to decoded. This is because the edge image results from encode and decode processes have different amounts. For

example edge image from the encode process has a 100 pixel edge while the edge image from pixel edge decode process have more than 100 and even can be less than 100.

From all tests carried out, the percentage success of the decoding process is :

$\frac{10}{25}$ x 100% = 40%.