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.

![Main Program 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.

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.
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.
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.
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.
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[][] pixeltepl = 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.
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**  
![Original Image](image1)

**Image Threshold**  
![Image Threshold](image2)

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.

![Code](image3)

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.

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.
Figure 18: Search coordinate code

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

Figure 19: Decryption Code
5.1.2. Program using Image Encryption:

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

![Initial Pixel](image)

Figure 22: Initial Pixel

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

![Grayscale Pixel](image)

Figure 23: Grayscale Pixel
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:

<table>
<thead>
<tr>
<th>1</th>
<th>92</th>
<th>92</th>
<th>92</th>
<th>92</th>
<th>92</th>
<th>92</th>
<th>92</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>3</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>6</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>7</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>8</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>9</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>10</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>11</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>12</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>13</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>14</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>15</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>16</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
</tbody>
</table>

Figure 24: Filter Pixel

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:

Image Result of Filtering:
The next process is edge detection with sobel method. Sobel method has two kernel i.e Gx and Gy.

\[
\begin{align*}
G_x : & \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} & \quad G_y : & \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}
\end{align*}
\]

\[M = |G_x| + |G_y|\]

The last process is threshold with range:

If pixel $\geq 128$ then change to 255

if pixel $< 128$ then change to 0

![Figure 25: Threshold pixel](image)

Image result of threshold process:

![Threshold Image](image)

After completing the process until the threshold, then go to the process of steganography.
After the image has done processing until threshold, user must input the message to be hidden. For example, the message to be hidden is: “eka prayogo thedy”.

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.
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

<table>
<thead>
<tr>
<th>Image</th>
<th>Messages</th>
<th>Image Edge Detection</th>
<th>Image Steganography</th>
<th>Decode Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Ikom" /></td>
<td>ikom</td>
<td><img src="" alt="Ikom" /></td>
<td>ikom</td>
<td><img src="" alt="Ikom" /></td>
</tr>
<tr>
<td><img src="" alt="Ikom Unika" /></td>
<td>ikom unika</td>
<td><img src="" alt="Ikom Unika" /></td>
<td>ikom unika</td>
<td><img src="" alt="Ikom Unika" /></td>
</tr>
<tr>
<td><img src="" alt="Ilmu Komputer Unika Soegijapranata" /></td>
<td>ilmu komputer unika soegijapranata</td>
<td><img src="" alt="Ilmu Komputer Unika Soegijapranata" /></td>
<td>ilmu komputer unika soegijapranata</td>
<td><img src="" alt="Ilmu Komputer Unika Soegijapranata" /></td>
</tr>
<tr>
<td><img src="" alt="Eka Prayogo Thedy Teknik Informatika 2013" /></td>
<td>eka prayogo thedy teknik informatika 2013</td>
<td><img src="" alt="Eka Prayogo Thedy Teknik Informatika 2013" /></td>
<td>eka prayogo thedy teknik informatika 2013</td>
<td><img src="" alt="Eka Prayogo Thedy Teknik Informatika 2013" /></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Image</th>
<th>Messages</th>
<th>Image Edge Detection</th>
<th>Image Steganography</th>
<th>Decode Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td>teknik informatika</td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
<td>teknik inLPäÜÄëØÖ</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eka prayogo thedy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| balonku ada lima rupa rupa warnanya hijau kuning kelabu merah muda dan biru | balonku@ÃÆÆ@ØvÜ• É ÖÁ„ÉÔÁ„¹ Y…ÉÚÚÜoÁ @DOÖâé@ ŒêÚÜÜí@Ö ÈO
| ilmu komputer teknik informatika unika |                                                                                   |
| 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

<table>
<thead>
<tr>
<th>Image</th>
<th>Messages</th>
<th>Image Edge Detection</th>
<th>Image Steganography</th>
<th>Decode Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image 1]</td>
<td>eka prayogo thedy</td>
<td>![Image Edge Detection 1]</td>
<td>![Image Steganography 1]</td>
<td>eka prayogo tDEE</td>
</tr>
<tr>
<td>![Image 2]</td>
<td>unika soegijapranat a semarang</td>
<td>![Image Edge Detection 2]</td>
<td>![Image Steganography 2]</td>
<td>unika soegijapranat a semarang</td>
</tr>
<tr>
<td>![Image 3]</td>
<td>ilmu komputer teknik informatika unika soegijapranat a semarang</td>
<td>![Image Edge Detection 3]</td>
<td>![Image Steganography 3]</td>
<td>ilmu komputer teknik informatika unika soegijapranat a semarang</td>
</tr>
<tr>
<td>![Image 4]</td>
<td>besok jangan lupa meeting dengan klien penting jam 8 di kantor</td>
<td>![Image Edge Detection 4]</td>
<td>![Image Steganography 4]</td>
<td>besok jangan lupa meeting dengan klien@àÊÜè ÖÜl@ÖÂÜ@ p@ÈÖ@ÖA ÜèP</td>
</tr>
</tbody>
</table>
you've been struggling to make things right. That's how a superhero learns to fly.

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

<table>
<thead>
<tr>
<th>Image</th>
<th>Messages</th>
<th>Image Edge Detection</th>
<th>Image Steganography</th>
<th>Decode Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>eka prayogo thedy</td>
<td>1</td>
<td>1</td>
<td>eka prayogo thedy</td>
</tr>
<tr>
<td>1</td>
<td>semarang ibu kota jawa tengah</td>
<td>1</td>
<td>1</td>
<td>semarang ibu kota jawa tengah</td>
</tr>
</tbody>
</table>
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
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:

\[
\frac{10}{25} \times 100\% = 40\%.
\]