APPENDIX

1. Grayscale :

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.*;
import javax.imageio.ImageIO;
public class Grayscale {
  BufferedImage  img;
  int width;
  int height;
  public Grayscale() {
    try {
  File input = new File("/home/yohan/Pictures/Project/Images/daun40.jpg");
    img = ImageIO.read(input);
    width = img.getWidth();
    height = img.getHeight();
    for(int i=0; i<height; i++){
      for(int j=0; j<width; j++){
        Color c = new Color(img.getRGB(j, i));
    Color c = new Color(img.getRGB(j, i));
    int red = (int)(c.getRed()*0.299);
    int green = (int) (c.getGreen()*0.587);
    int blue = (int) (c.getBlue()*0.114);
    int gray = (red + green + blue);
        Color newcolor = new Color(gray, gray, gray);
        img.setRGB(j, i, newcolor.getRGB());
      }
    }
  File output = new File("/home/yohan/Pictures/Project/Outputs/Grayscale/grayscale40.jpg");
    ImageIO.write(img, "jpg", output);
    } catch (Exception e) {}
    }
    static public void main(String args[]) throws Exception
    {
      Grayscale obj = new Grayscale();
    }
  }
```

2. Thresholding

```
import javax.imageio.ImageIO;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
public class Program {
  private int width;
  private int height;
```

```
9.    public static int RGBVal(BufferedImage img, int width, int
   height, int i, int j) {
10.     i = Math.max(0, Math.min(width - 1, i));
11.     j = Math.max(0, Math.min(height - 1, j));
12.     return img.getRGB(i, j);
13.   }
14.    public static int[] histo(BufferedImage img,int width, int
       height) {
15.     int interval[] = new int[256];
16.     for (int i = 0; i < img.getWidth(); i++) {
17.       for (int j = 0; j < img.getHeight(); j++) {
18.         int p = RGBVal( img, width, height, i, j);
19.         int r = (p >> 16) & 255;
20.         interval[r]++;
21.       }
22.     }
23.     return interval;
24.   }
25.   public static int calculate(BufferedImage img, int width, int
      height) {
26.     int histogram[] = histo(img,width, height );
27.     int total = width * height;
28.     float sum = 0;
29.     for (int i = 0; i < 256; i++) {
30.       sum += i * histogram[i];
31.     }
32.     float sum_back = 0;
33.     int wg_back = 0, wg_fore = 0;
34.     float varMax = 0;
35.     int threshold = 0;
36.     for (int i = 0; i < 256; i++) {
37.       wg_back += histogram[i];
38.       if (wg_back == 0) {
39.         continue;
40.       }
41.       wg_fore = total - wg_back;
42.       if (wg_fore == 0) {
43.         break;
44.     }
45.     sum_back += (float) (i * histogram[i]);
46.     float mean_back = sum_back / wg_back;
47.     float mean_fore = (sum - sum_back) / wg_fore;
48.     float varBetween = (float) wg_back * (float) wg_fore *
       (mean_back - mean_fore) * (mean_back - mean_fore);
49.     if (varBetween > varMax) {
50.       varMax = varBetween;
51.       threshold = i;
52.     }
53.   }
54.   return threshold;
55.   }
56.   public BufferedImage apply(BufferedImage img) {
57.     int th, val;
58.     this.width = img.getWidth();
```

```
59.     this.height = img.getHeight();
60.     double thresValue = calculate(img, width, height);
61.     BufferedImage Output = new
        BufferedImage(this.width,this.height,
        BufferedImage.TYPE_3BYTE_BGR);
62.     for (int i = 0; i < img.getWidth(); i++) {
63.       for (int j = 0; j < img.getHeight(); j++) {
64.       val = RGBVal(img,width,height, i, j);
65.       val = ((val >> 16) & 0xff);
66.       if (val > thresValue) {
67.         th = 255;
68.       } else {
69.         th = 0;
70.       }
71.       th = (th << 16) | (th << 8) | (th);
72.       Output.setRGB(i, j, th);
73.       }
74.     }
75.     System.out.println("Threshold = "+(int)(thresValue));
76.     return Output;
77.   }
78.   public void run(){
79.     BufferedImage img = null;
80.     try {
81.  File file = new
    File("/home/yohan/Pictures/Project/Outputs/Grayscale/grayscale
    30.jpg");
82.       img = ImageIO.read(file);
83.      } catch (Exception e) {
84.        System.out.println("File couldn't be loaded");
85.      }
86.      System.out.println("Image started processing");
87.      Program threshold = new Program();
88.      img = threshold.apply(img);
89.  File outfile = new
    File("/home/yohan/Pictures/Project/Outputs/Threshold/Thres30.j
    pg");
90.     try {
91.       ImageIO.write(img,"jpg",outfile);
92.     } catch (IOException e) {
93.        e.printStackTrace();
94.     }
95.     Desktop dt = Desktop.getDesktop();
96.     try {
97.       dt.open(outfile);
98.     } catch (IOException e) {
99.        e.printStackTrace();
100.     }
101.    }
102.  }
```

3. Morphological Closing

Erosi

```
1. for (int x = 0; x <= filWidth; x++) {
2.   for (int y = 0; y <= filHeight; y++) {
3.    window = (byte[]) oldData.getDataElements(x, y, sSize, sSize,
   null);
4. newValue = min(window);
5. newData.setSample(x + shpSize, y + shpSize, 0, newValue);
6. }
7. }
8. for (int x = 0; x < shpSize; x++) {
9.   for (int y = 0; y <= filHeight; y++) {
10.    window = (byte[]) oldData.getDataElements(0, y, sSize,
     sSize,null);
11.    newValue = min(window);
12.    newData.setSample(x, y + shpSize, 0, newValue);
13.    }
14.  }
15.  newData.setSamples(0, LoSide, shpSize, shpSize,
     0,fillArray(shpSize * shpSize, newValue));
16.  window = (byte[]) oldData.getDataElements(0, 0,
     sSize,sSize,null);
17.  newValue = min(window);
18.  newData.setSamples(0, 0, shpSize, shpSize,0,
     fillArray(shpSize*shpSize,newValue));
19.
20.  for (int x = RSide; x < imgWidth; x++) {
21.    for (int y = 0; y <= filHeight; y++) {
22.     window = (byte[]) oldData.getDataElements(filWidth, y,
     sSize, sSize, null);
23.       newValue = min(window);
24.       newData.setSample(x, y + shpSize, 0, newValue);
25.   }
26.  }
27.  newData.setSamples(RSide, LoSide, shpSize, shpSize, 0,
     fillArray(shpSize * shpSize, newValue));
28.
29.  for (int y = LoSide - 1; y < imgHeight; y++) {
30.    for (int x = 0; x <= filWidth; x++) {
31.     window = (byte[]) oldData.getDataElements(x, filHeight,
     sSize, sSize, null);
32.      newValue = min(window);
33.      newData.setSample(x + shpSize, y, 0, newValue);
34.   }
35.  }
36.
37.  for (int y = 0; y < shpSize; y++) {
38.    for (int x = 0; x <= filWidth; x++) {
39.     window = (byte[]) oldData.getDataElements(x, 0, sSize,
     sSize, null);
40.       newValue = min(window);
41.       newData.setSample(x + shpSize, y, 0, newValue);
```

```
42.    }
43.   }
44.  newData.setSamples(RSide,    0,    shpSize,    shpSize,    0,
     fillArray(shpSize * shpSize, newValue));
```

Dilation

```
1.   for (int x = 0; x <= filWidth; x++) {
2.     for (int y = 0; y <= filHeight; y++) {
3.      window  =  (byte[])  oldData.getDataElements(x,  y,  sSize,
     sSize, null);
4.        newValue = max(window);
5.      newData.setSample(x + shpSize, y + shpSize, 0, newValue);
6.      }
7.    }
8.
9.   for (int x = 0; x < shapeSize; x++) {
10.    for (int y = 0; y <= filHeight; y++) {
11.     window  =  (byte[])  oldData.getDataElements(0,  y,  sSize,
     sSize, null);
12.       newValue = max(window);
13.       newData.setSample(x, y + shpSize, 0, newValue);
14.     }
15.  }
16.
17.  newData.setSamples(0,    LoSide,    shpSize,    shpSize,    0,
     fillArray(shpSize * shpSize, newValue));
18.  window = (byte[]) oldData.getDataElements(0, 0, sSize, sSize,
     null);
19.  newValue = max(window);
20.  newData.setSamples(0,    0,    shpSize,    shpSize,    0,
     fillArray(shpSize * shpSize, newValue));
21.  for (int x = RSide; x < imgWidth; x++) {
22.    for (int y = 0; y <= filHeight; y++) {
23.     window  =  (byte[])  oldData.getDataElements(filWidth,  y,
     sSize, sSize, null);
24.      newValue = max(window);
25.      newData.setSample(x, y + shpSize, 0, newValue);
26.    }
27.  }
28.  newData.setSamples(RSide,   LoSide,   shpSize,   shpSize,   0,
     fillArray(shpSize * shpSize, newValue));
29.  for (int y = LoSide - 1; y < imgHeight; y++) {
30.      for (int x = 0; x <= filWidth; x++) {
31.       window = (byte[]) oldData.getDataElements(x, filHeight,
     sSize, sSize, null);
32.        newValue = max(window);
33.        newData.setSample(x + shpSize, y, 0, newValue);
34.     }
35.  }
36.
37.  for (int y = 0; y < shpSize; y++) {
38.     for (int x = 0; x <= filWidth; x++) {
```

```
39.     window = (byte[]) oldData.getDataElements(x, 0, sSize,
    sSize, null);
40.       newValue = max(window);
41.       newData.setSample(x + shpSize, y, 0, newValue);
42.     }
43. }
44. newData.setSamples(RSide, 0, shpSize, shpSize, 0,
  fillArray(shpSize * shpSize, newValue));
```

Closing

```
1.  BufferedImage dilatedImg, closedImg;
2.  Erosion erosion = new Erosion(shape, shapeSize);
3.  Dilation dilation = new Dilation(shape, shapeSize);
4.  dilatedImg = dilation.execute(img);
5.  closedImg = erosion.execute(dilatedImg);
```

4. PSNR dan MSE

```
1.  File image1 = new
    File("/home/yohan/Pictures/Project/Images/daun40.jpg");
2.  File image2 = new
    File("/home/yohan/Pictures/Project/Outputs/Closing2/close40.j
    pg");
3.  Img1 = ImageIO.read(image1);
4.  Img2 = ImageIO.read(image2);
5.  width = Img1.getWidth();
6.  height = Img1.getHeight();
7.  for(int i=0;i<height;i++) {
8.    for(int j=0;j<width;j++) {
9.      Color P1 = new Color(Img1.getRGB(j,i));
10.     Color P2 = new Color(Img2.getRGB(j,i));
11.     double R_P1 = (double)(P1.getRed());
12.     double R_P2 = (double)(P2.getRed());
13.     double G_P1 = (double)(P1.getGreen());
14.     double G_P2 = (double)(P2.getGreen());
15.     double B_P1 = (double)(P1.getBlue());
16.     double B_P2 = (double)(P2.getBlue());
17.     if(R_P1>R_img1) {
18.       R_img1=R_P1;
19.     }
20.     if(G_P1>G_img1) {
21.       G_img1=G_P1;
22.     }
23.     if(B_P1>B_img1) {
24.       B_img1=B_P1;
25.     }
26.     if(R_P2>R_img2) {
27.       R_img2=R_P2;
28.     }
29.     if(G_P2>G_img2) {
30.       R_img2=R_P2;
31.     }
32.     if(B_P2>B_img2) {
```

```
33.         R_img2=R_P2;
34.       }
35.       R_MSE=R_MSE+Math.pow((R_P2-R_P1),2);
36.       G_MSE=G_MSE+Math.pow((G_P2-G_P1),2);
37.       B_MSE=B_MSE+Math.pow((B_P2-B_P1),2);
38.       }
39.     }
40.     R_MSE = R_MSE/(width*height);
41.     G_MSE = G_MSE/(width*height);
42.     B_MSE = B_MSE/(width*height);
43.     MSE = (R_MSE+G_MSE+B_MSE)/3;
44.     R_PSNR = 10.0 * logbase10(Math.pow(Math.max(R_img1,R_img2),
    2) / R_MSE);
45.     G_PSNR = 10.0 * logbase10(Math.pow(Math.max(G_img1,G_img2),
    2) / G_MSE);
46.     B_PSNR = 10.0 * logbase10(Math.pow(Math.max(B_img1,B_img2),
    2) / B_MSE);
47.     PSNR = (R_PSNR+G_PSNR+B_PSNR)/3;
48.     System.out.println("MSE = "+dfmse.format(MSE));
49.     System.out.println("PSNR = "+dfpsnr.format(PSNR)+" dB");
50.   }
51.   catch(IOException e)
52.     {}
53.    }
54.   public static double logbase10(double x) {
55.      return Math.log(x) / Math.log(10);
56.   }
```

PLAGIARISM CHECK.ORG

1.08% PLAGIARISM APPROXIMATELY

Report #12261041

Introduction Background Digital imagery is an image or similarity of an object recorded by an electronic device.Studying digital image processing is very useful in improving imagery or wanting to make the image look interesting to see. One of the things learned in digital processing is segmenting or separating objects with the background.In segmenting there are many examples of algorithms that can segment, for example Thresholding algorithms and Morphological algorithms. In segmentation, of course, must determine the algorithm that has a low error rate and maximum image level. [1] [2] [3] To find out this, PSNR (Peak Signal to Noise Ratio) and MSE (Mean Square Error) tests are provided. PSNR and MSE are parameters commonly used as indicators to measure the similarity of two images. The better the image, the value in MSE is close to 0 while for PSNR it is said to be good if the value is above 30 dB. To know the final result, a table will be created so that the value of the 2 algorithms can be known. Scope The scope of this research : Can separate objects with background using Thresholding and Morphological (Closing) algorithms. Can compare Thresholding algorithm with Morphological Closing with PSNR and MSE parameters. Objective The objectives that will be achieved from this project are: Get algorithms that match MSE conditions close to 0 and