APPENDIX

CannyEdgeDetector.java

```
package skripsi.sidik.jari;

import java.awt.image.BufferedImage;
import java.util.Arrays;

public class CannyEdgeDetector {

    private final static float GAUSSIAN_CUT_OFF = 0.005f;
    private final static float MAGNITUDE_SCALE = 100F;
    private final static float MAGNITUDE_LIMIT = 1000F;
    private final static int MAGNITUDE_MAX = (int) (MAGNITUDE_SCALE *
MAGNITUDE_LIMIT);

    private int height;
    private int width;
    private int picsize;
    private int[] data;
    private int[] magnitude;
    private BufferedImage sourceImage;
    private BufferedImage edgesImage;

    private float gaussianKernelRadius;
    private float lowThreshold;
    private float highThreshold;
    private int gaussianKernelWidth;
    private boolean contrastNormalized;

    private float[] xConv;
    private float[] yConv;
    private float[] xGradient;
    private float[] yGradient;

    public CannyEdgeDetector() {
        lowThreshold = 0.5f; // untuk proses histereris
        highThreshold = 1f; // untuk proses histeresis
        gaussianKernelRadius = 2f;
        gaussianKernelWidth = 16;
        contrastNormalized = false;
    }

    // untuk mendapatkan image input
    public BufferedImage getSourceImage() {
        return sourceImage;
    }

    // mensetting image input untuk di proses oleh canny
    public void setSourceImage(BufferedImage image) {
        sourceImage = image;
```

```
    }

    // untuk output hasil deteksi tepi, output berupa image
    public BufferedImage getEdgesImage() {
        return edgesImage;
    }

    //
//    public void setEdgesImage(BufferedImage edgesImage) {
//        this.edgesImage = edgesImage;
//    }
    public float getLowThreshold() {
        return lowThreshold;
    }

    public void setLowThreshold(float threshold) {
        if (threshold < 0) {
            throw new IllegalArgumentException();
        }
        lowThreshold = threshold;
    }

    public float getHighThreshold() {
        return highThreshold;
    }

    public void setHighThreshold(float threshold) {
        if (threshold < 0) {
            throw new IllegalArgumentException();
        }
        highThreshold = threshold;
    }

    public int getGaussianKernelWidth() {
        return gaussianKernelWidth;
    }

    public void setGaussianKernelWidth(int gaussianKernelWidth) {
        if (gaussianKernelWidth < 2) {
            throw new IllegalArgumentException();
        }
        this.gaussianKernelWidth = gaussianKernelWidth;
    }

    public float getGaussianKernelRadius() {
        return gaussianKernelRadius;
    }

    public void setGaussianKernelRadius(float gaussianKernelRadius) {
        if (gaussianKernelRadius < 0.1f) {
            throw new IllegalArgumentException();
        }
        this.gaussianKernelRadius = gaussianKernelRadius;
```

```
    }

    public boolean isContrastNormalized() {
        return contrastNormalized;
    }

    public void setContrastNormalized(boolean contrastNormalized) {
        this.contrastNormalized = contrastNormalized;
    }

    // [3 4 5]
    // [5 4 5]
    // [3 3 3]
    // 33 / 9  = 3
    // [3 3 3]
    // [3 3 3]
    // [3 3 3]
    // methods
    public void process() {
        width = sourceImage.getWidth();
        height = sourceImage.getHeight();
        picsize = width * height;
        initArrays();
        readLuminance();
//        if (contrastNormalized) {
//            normalizeContrast();
//        }
        computeGradients(gaussianKernelRadius, gaussianKernelWidth);
        int low = Math.round(lowThreshold * MAGNITUDE_SCALE);
        int high = Math.round(highThreshold * MAGNITUDE_SCALE);
        performHysteresis(low, high);
        thresholdEdges();
        writeEdges(data);
    }

    private void initArrays() {
        if (data == null || picsize != data.length) {
            data = new int[picsize];
            magnitude = new int[picsize];

            xConv = new float[picsize];
            yConv = new float[picsize];
            xGradient = new float[picsize];
            yGradient = new float[picsize];
        }
    }

    private void computeGradients(float kernelRadius, int kernelWidth) {

        float kernel[] = new float[kernelWidth];
        float diffKernel[] = new float[kernelWidth];
        int kwidth;
        for (kwidth = 0; kwidth < kernelWidth; kwidth++) {
```

```
    float g1 = gaussian(kwidth, kernelRadius);
    if (g1 <= GAUSSIAN_CUT_OFF && kwidth >= 2) {
        break;
    }
    float g2 = gaussian(kwidth - 0.5f, kernelRadius);
    float g3 = gaussian(kwidth + 0.5f, kernelRadius);
    kernel[kwidth] = (g1 + g2 + g3) / 3f / (2f * (float) Math.PI * kernelRadius * kernelRadius);
    diffKernel[kwidth] = g3 - g2;
}

int initX = kwidth - 1;
int maxX = width - (kwidth - 1);
int initY = width * (kwidth - 1);
int maxY = width * (height - (kwidth - 1));

for (int x = initX; x < maxX; x++) {
    for (int y = initY; y < maxY; y += width) {
        int index = x + y;
        float sumX = data[index] * kernel[0];
        float sumY = sumX;
        int xOffset = 1;
        int yOffset = width;
        for (; xOffset < kwidth;) {
            sumY += kernel[xOffset] * (data[index - yOffset] + data[index + yOffset]);
            sumX += kernel[xOffset] * (data[index - xOffset] + data[index + xOffset]);
            yOffset += width;
            xOffset++;
        }

        yConv[index] = sumY;
        xConv[index] = sumX;
    }

}

for (int x = initX; x < maxX; x++) {
    for (int y = initY; y < maxY; y += width) {
        float sum = 0f;
        int index = x + y;
        for (int i = 1; i < kwidth; i++) {
            sum += diffKernel[i] * (yConv[index - i] - yConv[index + i]);
        }

        xGradient[index] = sum;
    }

}

for (int x = kwidth; x < width - kwidth; x++) {
    for (int y = initY; y < maxY; y += width) {
        float sum = 0.0f;
        int index = x + y;
        int yOffset = width;
```

```
            for (int i = 1; i < kwidth; i++) {
               sum += diffKernel[i] * (xConv[index - yOffset] - xConv[index + yOffset]);
               yOffset += width;
            }

            yGradient[index] = sum;
         }

      }

      initX = kwidth;
      maxX = width - kwidth;
      initY = width * kwidth;
      maxY = width * (height - kwidth);
      for (int x = initX; x < maxX; x++) {
         for (int y = initY; y < maxY; y += width) {
            int index = x + y;
            int indexN = index - width;
            int indexS = index + width;
            int indexW = index - 1;
            int indexE = index + 1;
            int indexNW = indexN - 1;
            int indexNE = indexN + 1;
            int indexSW = indexS - 1;
            int indexSE = indexS + 1;

            float xGrad = xGradient[index];
            float yGrad = yGradient[index];
            float gradMag = hypot(xGrad, yGrad);

            //perform non-maximal supression
            float nMag = hypot(xGradient[indexN], yGradient[indexN]);
            float sMag = hypot(xGradient[indexS], yGradient[indexS]);
            float wMag = hypot(xGradient[indexW], yGradient[indexW]);
            float eMag = hypot(xGradient[indexE], yGradient[indexE]);
            float neMag = hypot(xGradient[indexNE], yGradient[indexNE]);
            float seMag = hypot(xGradient[indexSE], yGradient[indexSE]);
            float swMag = hypot(xGradient[indexSW], yGradient[indexSW]);
            float nwMag = hypot(xGradient[indexNW], yGradient[indexNW]);
            float tmp;

            if (xGrad * yGrad <= (float) 0 /*(1)*/
                  ? Math.abs(xGrad) >= Math.abs(yGrad) /*(2)*/
                  ? (tmp = Math.abs(xGrad * gradMag)) >= Math.abs(yGrad * neMag - (xGrad +
yGrad) * eMag) /*(3)*/
                  && tmp > Math.abs(yGrad * swMag - (xGrad + yGrad) * wMag) /*(4)*/
                  : (tmp = Math.abs(yGrad * gradMag)) >= Math.abs(xGrad * neMag - (yGrad +
xGrad) * nMag) /*(3)*/
                  && tmp > Math.abs(xGrad * swMag - (yGrad + xGrad) * sMag) /*(4)*/
                  : Math.abs(xGrad) >= Math.abs(yGrad) /*(2)*/
                  ? (tmp = Math.abs(xGrad * gradMag)) >= Math.abs(yGrad * seMag + (xGrad -
yGrad) * eMag) /*(3)*/
                  && tmp > Math.abs(yGrad * nwMag + (xGrad - yGrad) * wMag) /*(4)*/
```

```
                    : (tmp = Math.abs(yGrad * gradMag)) >= Math.abs(xGrad * seMag + (yGrad -
xGrad) * sMag) /*(3)*/
                        && tmp > Math.abs(xGrad * nwMag + (yGrad - xGrad) * nMag) /*(4)*/) {
                    magnitude[index] = gradMag >= MAGNITUDE_LIMIT ? MAGNITUDE_MAX :
(int) (MAGNITUDE_SCALE * gradMag);

                } else {
                    magnitude[index] = 0;
                }
            }
        }
    }

    private float hypot(float x, float y) {
        return (float) Math.hypot(x, y);
    }

    private float gaussian(float x, float sigma) {
        return (float) Math.exp(-(x * x) / (2f * sigma * sigma));
    }

    private void performHysteresis(int low, int high) {

        Arrays.fill(data, 0);

        int offset = 0;
        for (int y = 0; y < height; y++) {
            for (int x = 0; x < width; x++) {
                if (data[offset] == 0 && magnitude[offset] >= high) {
                    follow(x, y, offset, low);
                }
                offset++;
            }
        }
    }

    private void follow(int x1, int y1, int i1, int threshold) {
        int x0 = x1 == 0 ? x1 : x1 - 1;
        int x2 = x1 == width - 1 ? x1 : x1 + 1;
        int y0 = y1 == 0 ? y1 : y1 - 1;
        int y2 = y1 == height - 1 ? y1 : y1 + 1;

        data[i1] = magnitude[i1];
        for (int x = x0; x <= x2; x++) {
            for (int y = y0; y <= y2; y++) {
                int i2 = x + y * width;
                if ((y != y1 || x != x1)
                        && data[i2] == 0
                        && magnitude[i2] >= threshold) {
                    follow(x, y, i2, threshold);
                    return;
                }
            }
```

```
        }
    }

    private void thresholdEdges() {
        for (int i = 0; i < picsize; i++) {
            data[i] = data[i] > 0 ? -1 : 0xff000000;
        }
    }

    //Pencahayaan pada citra
    private int luminance(float r, float g, float b) {
        return Math.round(0.299f * r + 0.587f * g + 0.114f * b);
    }

    private void readLuminance() {
        int type = sourceImage.getType();
//        if (type == BufferedImage.TYPE_INT_RGB || type == BufferedImage.TYPE_INT_ARGB)
{
//            int[] pixels = (int[]) sourceImage.getData().getDataElements(0, 0, width, height, null);
//            for (int i = 0; i < picsize; i++) {
//                int p = pixels[i];
//                int r = (p & 0xff0000) >> 16;
//                int g = (p & 0xff00) >> 8;
//                int b = p & 0xff;
//                data[i] = luminance(r, g, b);
//            }
//        } else if (type == BufferedImage.TYPE_BYTE_GRAY) {
//            byte[] pixels = (byte[]) sourceImage.getData().getDataElements(0, 0, width, height, null);
//            for (int i = 0; i < picsize; i++) {
//                data[i] = (pixels[i] & 0xff);
//            }
//        } else if (type == BufferedImage.TYPE_USHORT_GRAY) {
//            short[] pixels = (short[]) sourceImage.getData().getDataElements(0, 0, width, height,
null);
//            for (int i = 0; i < picsize; i++) {
//                data[i] = (pixels[i] & 0xffff) / 256;
//            }
//        } else
        if (type == BufferedImage.TYPE_3BYTE_BGR) {
            byte[] pixels = (byte[]) sourceImage.getData().getDataElements(0, 0, width, height, null);
            int offset = 0;
            for (int i = 0; i < picsize; i++) {
                int b = pixels[offset++] & 0xff;
                int g = pixels[offset++] & 0xff;
                int r = pixels[offset++] & 0xff;
                data[i] = luminance(r, g, b);
            }
//        } else {
//            throw new IllegalArgumentException("Unsupported image type: " + type);
        }
    }

//    private void normalizeContrast() {
```

```
//        int[] histogram = new int[256];
//        for (int i = 0; i < data.length; i++) {
//            histogram[data[i]]++;
//        }
//        int[] remap = new int[256];
//        int sum = 0;
//        int j = 0;
//        for (int i = 0; i < histogram.length; i++) {
//            sum += histogram[i];
//            int target = sum * 255 / picsize;
//            for (int k = j + 1; k <= target; k++) {
//                remap[k] = i;
//            }
//            j = target;
//        }
//
//        for (int i = 0; i < data.length; i++) {
//            data[i] = remap[data[i]];
//        }
//    }
    private void writeEdges(int pixels[]) {
        if (edgesImage == null) {
            edgesImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
        }
        edgesImage.getWritableTile(0, 0).setDataElements(0, 0, width, height, pixels);
    }
}
```

EuclideanDistance.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package skripsi.sidik.jari;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;

/**
 *
 *
 */
public class Euclidean {

   public static double jarakEuclidean(BufferedImage img1, BufferedImage img2) {

        int gray1, gray2 = 0;
          double jarakEuclid = 0;
       int barisImg1 = img1.getWidth();
       int barisImg2 = img2.getWidth();

       int kolomImg1 = img1.getHeight();
       int kolomImg2 = img2.getHeight();

       if ((barisImg1 == barisImg2) && (kolomImg1 == kolomImg2)) {

          for (int baris = 0; baris < barisImg1; baris++) {
             for (int kolom = 0; kolom < kolomImg1; kolom++) {
                int rgb1 = img1.getRGB(baris, kolom);
                int r1 = (rgb1 >> 16) & 0xFF;
                int g1 = (rgb1 >> 8) & 0xFF;
                int b1 = (rgb1 & 0xFF);
                gray1 = (r1 + g1 + b1) / 3;

                int rgb2 = img2.getRGB(baris, kolom);
                int r2 = (rgb2 >> 16) & 0xFF;
                int g2 = (rgb2 >> 8) & 0xFF;
                int b2 = (rgb2 & 0xFF);
                gray2 = (r2 + g2 + b2) / 3;

                jarakEuclid = jarakEuclid + Math.sqrt(Math.pow(gray1 - gray2, 2));

             }
          }
       }
```

```
        return jarakEuclid;
    }

    public static void main(String[] args) throws IOException {
//        File coba = new File("D://Sidikjari/dataset/1_canny.jpg");
//        BufferedImage img1 = ImageIO.read(coba);
//
//
//         File input1 = new File("D://Sidikjari/dataset/1.jpg");
//
//        BufferedImage inputCitra = null;
//        try {
//            inputCitra = ImageIO.read(input1);
//        } catch (IOException ex) {
//            Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);
//        }
//
//         CannyEdgeDetector detector = new CannyEdgeDetector();
//
//        detector.setLowThreshold(0.5f);
//        detector.setHighThreshold(1f);
//        detector.setSourceImage(inputCitra);
//        detector.process();
//        BufferedImage HasilCanny = detector.getEdgesImage();
//
//
//        File coba1 = new File("D://Sidikjari/dataset/2_canny.jpg");
//        BufferedImage img2 = ImageIO.read(coba1);
//
//
//        System.out.println(jarakEuclidean(HasilCanny, HasilCanny));
//
//
//        System.out.println(jarakEuclidean(img1, img1));
    }

}
```

MainGUI.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package skripsi.sidik.jari;

import java.awt.Dimension;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;

/**
 *
 *
 */
public class MainGUI extends javax.swing.JFrame {

    /**
     * Creates new form MainGUI
     */
    public MainGUI() throws IOException {
        initComponents();
        this.setTitle("Edge Detection");
        this.setLocationRelativeTo(this);




    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
```

```
        jLabel2 = new javax.swing.JLabel();
        jPanel1 = new javax.swing.JPanel();
        jButton2 = new javax.swing.JButton();
        inputImage = new javax.swing.JLabel();
        path = new javax.swing.JTextField();
        jPanel2 = new javax.swing.JPanel();
        outputCanny = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jPanel3 = new javax.swing.JPanel();
        outputEulid = new javax.swing.JLabel();
        jButton3 = new javax.swing.JButton();
        hasilEuclidean = new javax.swing.JTextField();
        jPanel4 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        identitas = new javax.swing.JTextField();
        jLabel4 = new javax.swing.JLabel();
        jenis = new javax.swing.JTextField();
        jPanel5 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        outputTeks = new javax.swing.JTextArea();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel1.setText("CLASSIFYING FINGERPRINT IMAGES ACCORDING TO
FINGERPRINT PATTERN");

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        jLabel2.setText("USING CANNY EGDE DETECTION AND EUCLIDEAN DISTANCE
METHOD");

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Image Input"));

        jButton2.setText("Upload");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        inputImage.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING)
                    .addComponent(inputImage, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(path, javax.swing.GroupLayout.DEFAULT_SIZE, 139,
Short.MAX_VALUE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                    .addComponent(jButton2)))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(inputImage, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
                    .addComponent(jButton2)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(1, 1, 1)
                        .addComponent(path)))
                .addContainerGap())
        );

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder("Canny Edge Detection"));

        outputCanny.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jButton1.setText("Get Result");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                        .addGap(0, 130, Short.MAX_VALUE)
                        .addComponent(jButton1))
                    .addComponent(outputCanny, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(outputCanny, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jButton1)
                .addContainerGap())
        );

        jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder("Eulidean Distance"));

        outputEulid.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jButton3.setText("Get Result");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addComponent(outputEulid, javax.swing.GroupLayout.PREFERRED_SIZE, 213,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 0, Short.MAX_VALUE))
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addComponent(hasilEuclidean)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButton3)))
                .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(outputEulid, javax.swing.GroupLayout.PREFERRED_SIZE, 223,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(jButton3)
                    .addComponent(hasilEuclidean, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(12, Short.MAX_VALUE))
        );
```

```
        jPanel4.setBorder(javax.swing.BorderFactory.createTitledBorder("Classification"));

        jLabel3.setText("Fingerprint Identity");

        jLabel4.setText("Type of Fingerprint");

        javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
        jPanel4.setLayout(jPanel4Layout);
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addGap(21, 21, 21)
                .addComponent(jLabel3)
                .addGap(18, 18, 18)
                .addComponent(identitas, javax.swing.GroupLayout.PREFERRED_SIZE, 200,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(87, 87, 87)
                .addComponent(jLabel4)
                .addGap(18, 18, 18)
                .addComponent(jenis)
                .addContainerGap())
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
                    .addComponent(jLabel3)
                    .addComponent(identitas, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel4)
                    .addComponent(jenis, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(16, Short.MAX_VALUE))
        );

        jPanel5.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        outputTeks.setColumns(20);
        outputTeks.setRows(5);
        jScrollPane1.setViewportView(outputTeks);

        javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);
        jPanel5.setLayout(jPanel5Layout);
        jPanel5Layout.setHorizontalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane1)
                .addContainerGap())
        );
```

```
        jPanel5Layout.setVerticalGroup(
            jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel5Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 199,
Short.MAX_VALUE)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addGap(0, 0, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(109, 109, 109))
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(142, 142, 142)
                        .addComponent(jLabel2)))
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jPanel4, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jPanel5, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(17, 17, 17)
                .addComponent(jLabel1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jLabel2)
```

```
            .addGap(23, 23, 23)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

//        BufferedImage image = null;
//        try {
//            File img = new File("D://COBA.jpg");
//            image = ImageIO.read(img);
//            System.out.println(image);
//        } catch (IOException e) {
//            e.printStackTrace();
//        }
        BufferedImage img = null;
        try {
            img = ImageIO.read(new File(filenya.getPath()));
        } catch (IOException e) {
            e.printStackTrace();
        }

        CannyEdgeDetector detector = new CannyEdgeDetector();

        detector.setLowThreshold(0.5f);
        detector.setHighThreshold(1f);

        detector.setSourceImage(img);
        detector.process();
        BufferedImage HasilCanny = detector.getEdgesImage();
        System.out.println(HasilCanny);
        //  Image image=GenerateImage.toImage(true);  //this generates an image file
        ImageIcon icon = new ImageIcon(HasilCanny);

        Image dimg = HasilCanny.getScaledInstance(outputCanny.getWidth(),
```

```
outputCanny.getHeight(),
            Image.SCALE_SMOOTH);
        ImageIcon imageIcon = new ImageIcon(dimg);
        // inputImage.setIcon(imageIcon);
        outputCanny.setIcon(imageIcon);


    }
    JFileChooser chooser = new JFileChooser("");
    File filenya;

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        int buka_dialog = chooser.showOpenDialog(MainGUI.this);
        if (buka_dialog == JFileChooser.APPROVE_OPTION) {
            filenya = chooser.getSelectedFile();
            //  file =  chooser.getSelectedFile();

            BufferedImage img = null;
            try {
                img = ImageIO.read(new File(filenya.getPath()));
                path.setText(filenya.getPath());
            } catch (IOException e) {
                e.printStackTrace();
            }
            Image dimg = img.getScaledInstance(inputImage.getWidth(), inputImage.getHeight(),
                    Image.SCALE_SMOOTH);
            ImageIcon imageIcon = new ImageIcon(dimg);
            inputImage.setIcon(imageIcon);
        }
    }


    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        ToCanny data = new ToCanny();
        long startTime = System.currentTimeMillis();
        Euclidean e = new Euclidean();
        File input1 = new File(filenya.getPath());
        //   File input1 = new File("D://Sidikjari/dataset/1.jpg");

        BufferedImage inputCitra = null;
        try {
            inputCitra = ImageIO.read(input1);
        } catch (IOException ex) {
            Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);
        }

        CannyEdgeDetector detector = new CannyEdgeDetector();

        detector.setLowThreshold(0.5f);
        detector.setHighThreshold(1f);
```

```
        detector.setSourceImage(inputCitra);
        detector.process();
        //  BufferedImage HasilCanny = detector.getEdgesImage();
        File direktori = null;

        BufferedImage direktoriCitra;
        double[] hasilEuclidean = new double[72];
        double minValue = 10000000;
        try {
           //JPEGImageDecoder decoder;
           // BufferedImage sourceImg =
           for (int i = 0; i < 72; i++) {
              //  direktori = new File("dataset/" + i + "_canny.jpg");
              //      decoder = JPEGCodec.createJPEGDecoder(new FileInputStream(new
File("dataset/" + i + "_canny.jpg")));
              direktori = new File("/home/marco/Marco/SKRIPSI/EUCLIDEAN/Progress Akhir
Euclidean/Sidik Jari Euclidean/Sidikjari/dataset/" + i + ".jpg");
              direktoriCitra = ImageIO.read(direktori);
              //   System.out.println("Hasil euclidean dengan citra " + i + " = " +
+e.jarakEuclidean(HasilCanny, direktoriCitra));
              System.out.println((int) e.jarakEuclidean(inputCitra, direktoriCitra));
              outputTeks.append("Result of euclidean using Image " + i + " Comparation = " +
+e.jarakEuclidean(inputCitra, direktoriCitra) + "\n");

              hasilEuclidean[i] = e.jarakEuclidean(inputCitra, direktoriCitra);

              if (e.jarakEuclidean(inputCitra, direktoriCitra) < minValue) {
                 minValue = e.jarakEuclidean(inputCitra, direktoriCitra);
                 Image dimg = direktoriCitra.getScaledInstance(outputEulid.getWidth(),
outputEulid.getHeight(),
                       Image.SCALE_SMOOTH);
                 ImageIcon imageIcon = new ImageIcon(dimg);
                 outputEulid.setIcon(imageIcon);
                 this.hasilEuclidean.setText(direktori.getPath());


                 //File nama = new File("dataset/" + direktori.getName() + ".jpg");
                 identitas.setText(data.nama[i]);
                 jenis.setText(data.jenis[i]);
              }

           }

           System.out.println("");
           long endTime = System.currentTimeMillis();
           System.out.println((endTime - startTime) + " milliseconds");

        } catch (IOException ex) {
           Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);
        }

     }
```

```
//    public static int getMin(int[] inputArray) {
//        int minValue = inputArray[0];
//        for (int i = 1; i < inputArray.length; i++) {
//            if (inputArray[i] < minValue) {
//                minValue = inputArray[i];
//            }
//        }
//        return minValue;
//    }
    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Windows".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainGUI.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainGUI.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainGUI.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainGUI.class.getName()).log(java.util.logging.Level.SEVER
E, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    new MainGUI().setVisible(true);
                } catch (IOException ex) {
                    Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);
                }
```

```
        }
    });

}

// Variables declaration - do not modify
private javax.swing.JTextField hasilEuclidean;
private javax.swing.JTextField identitas;
private javax.swing.JLabel inputImage;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JPanel jPanel5;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextField jenis;
private javax.swing.JLabel outputCanny;
private javax.swing.JLabel outputEulid;
private javax.swing.JTextArea outputTeks;
private javax.swing.JTextField path;
// End of variables declaration
}
```

PLAGIARISM CHECK.ORG

1.14% PLAGIARISM APPROXIMATELY

Report #12352099

CHAPTER 1 Introduction Background Nowadays, fingerprints were used a lot in several things as an identity due to identity in the form of property does not rule out the possibility of imitation or loss. With the advance technologies that is available nowadays, fingerprints become one of the way to verify identity. Every human being have unique pattern of fingerprints so every person on this earth have different fingerprint patterns. Those difference lead to the existence of many types of fingerprints. So, the aim of this research is to groups those types of fingerprints based on the patterns. Currently, Euclidean distance is used as the basis of the various kinds of processing on the data. In this research, this Euclidean distance is used to groups the fingerprints image. If the results from the test image is close to the reference image then the result is correct, however if the results from the test image is far from the reference image then the result is false. The results from this project is that the fingerprints can be put into grouping based on their pattern. Fingerprint image based on the type of the data / type of the pattern in general are grouped into 4 groups and the resul t from those groupings get an accuracy value through 2 ways which are scanner and stamps. Problem Formulation Are the fingerprints can be classified / groupe d based on their patterns? Which fingerprint image can be identified better? The