APPENDIX

CODING BILINEAR INTERPOLATION

```
import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.text.DecimalFormat;
import java.text.NumberFormat;

class Pixel {
    public int[] ARGB = new int[4];

    Pixel(int ARGB) {
        for (int i = 0; i < this.ARGB.length; i++) {
            this.ARGB[i] = (ARGB >> (8 * i)) & 0xFF;
        }
    }

    public int getRGB() {
        int RGB = 0;
        for (int i = 0; i < this.ARGB.length; i++) {
            RGB |= this.ARGB[i] << (8 * i);
        }
        return RGB;
    }
}

public class Bilinear {
    private static double lerp(double a, double b, double t) {
        return (1 - t) * a + t * b;
    }

     private static BufferedImage scale(BufferedImage img, double
scale) {
        long start = System.currentTimeMillis();
        int newWidth = (int)(img.getWidth() * scale);
        int newHeight = (int)(img.getHeight() * scale);

            BufferedImage newImage = new BufferedImage(newWidth,
newHeight, img.getType());

        for (int y = 0; y < newHeight; y++) {
            for (int x = 0; x < newWidth; x++) {
                    double yImgReal = (double)y / newHeight *
(img.getHeight() - 1);
                     double xImgReal = (double)x / newWidth *
(img.getWidth() - 1);

                int yImgInt = (int)yImgReal;
                int xImgInt = (int)xImgReal;
```

```
                double xt = xImgReal - xImgInt;
                double yt = yImgReal - yImgInt;

                    Pixel q12 = new Pixel(img.getRGB(xImgInt,
yImgInt));
                   Pixel q22 = new Pixel(img.getRGB(xImgInt + 1,
yImgInt));
                 Pixel q11 = new Pixel(img.getRGB(xImgInt, yImgInt
+ 1));
                   Pixel q21 = new Pixel(img.getRGB(xImgInt + 1,
yImgInt + 1));

                Pixel p = new Pixel(0);

                for (int i = 0; i < 4; i++) {
                    int i12 = q12.ARGB[i];
                    int i22 = q22.ARGB[i];
                    int i11 = q11.ARGB[i];
                    int i21 = q21.ARGB[i];

                    double r1 = lerp(i11, i21, xt);
                    double r2 = lerp(i12, i22, xt);

                    p.ARGB[i] = (int)lerp(r2, r1, yt);

                }

                newImage.setRGB(x, y, p.getRGB());
            }
        }
        long end = System.currentTimeMillis();
        NumberFormat formatter = new DecimalFormat("#0.00000");
                System.out.println("Execution time is " +
formatter.format((end - start) / 1000d) + " seconds");
        return newImage;

    }

    public static void main(String[] args) {
        if (args.length != 3) {
            System.err.println("Tidak ada data");
            System.exit(1);
        }

        String imgPath = args[0];
        String scaleArg = args[1];
        String newPath = args[2];

        double scale = 1.0;
        try {
            scale = Double.parseDouble(scaleArg);
        } catch (Exception e) {
```

```
            System.err.println("Error: tidak bisa membaca skala: "
+ scaleArg);
            System.exit(1);
        }

        BufferedImage img = null;
        try {
            img = ImageIO.read(new File(imgPath));
        } catch (Exception e) {
              System.err.println("Error: tidak bisa membaca input
file: " + imgPath);
            System.exit(1);
        }

        BufferedImage newImage = scale(img, scale);

        try {
            ImageIO.write(newImage, "jpg", new File(newPath));

        } catch (Exception e) {
              System.err.println("Error: tidak bisa membaca output
file: " + newPath);
        }
    }
}
```

CODE BICUBIC INTERPOLATION

```
import java.util.Iterator;
import java.util.Scanner;
import javax.imageio.IIOImage;
import javax.imageio.ImageIO;
import javax.imageio.ImageWriteParam;
import javax.imageio.ImageWriter;
import javax.imageio.stream.FileImageOutputStream;
import java.awt.Graphics2D;
import java.awt.RenderingHints;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;


public class ImageRescale
{

      public static void writeImage(File outputFile, BufferedImage
image) throws FileNotFoundException, IOException
      {

            String name = outputFile.getName().toLowerCase();
```

```
            String suffix = name.substring(name.lastIndexOf('.') +
1);

            boolean isJPG = suffix.endsWith("jpg");

            Iterator<ImageWriter>            writers            =
ImageIO.getImageWritersBySuffix(suffix);
            if (!writers.hasNext())
                  System.err.println("Unrecognized    image    file
extension " + suffix);

            outputFile.createNewFile();
            ImageWriter writer = writers.next();
            writer.setOutput(new
FileImageOutputStream(outputFile));

            ImageWriteParam param = writer.getDefaultWriteParam();

            if (isJPG)
            {

      param.setCompressionMode(ImageWriteParam.MODE_EXPLICIT);
                  param.setCompressionQuality(0.5f);
            }

            IIOImage iioImage = new IIOImage(image, null, null);
            writer.write(null, iioImage, param);
      }

      public   static   void   imageRescale(File   inputFile,   File
outputFile, double scaleFactor)
      {
            System.out.println("Rescaling process started for " +
inputFile);

            try
            {
                  long start = System.currentTimeMillis();
                  BufferedImage source = ImageIO.read(inputFile);

                              boolean         hasAlpha         =
source.getColorModel().hasAlpha();

                  int sourceW = source.getWidth();
                  int sourceH = source.getHeight();

                  int w = (int) (sourceW * scaleFactor + 0.5);
                  int h = (int) (sourceH * scaleFactor + 0.5);

                  int       format       =       hasAlpha       ?
BufferedImage.TYPE_INT_ARGB : BufferedImage.TYPE_INT_RGB;

                  BufferedImage  output  =  new  BufferedImage(w,  h,
format);
```

```
                Graphics2D g = output.createGraphics();

      g.setRenderingHint(RenderingHints.KEY_INTERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BICUBIC);
                g.drawImage(source, 0, 0, w, h, null);

                writeImage(outputFile, output);
                long end = System.currentTimeMillis();
                NumberFormat       formatter       =       new
DecimalFormat("#0.00000");
                System.out.println("Execution   time   is   "   +
formatter.format((end - start) / 1000d) + " seconds");
            }
            catch (IOException e)
            {
                System.err.println(e.getMessage());
                return;
            }
      }

      public static void main(String args[])
      {

            System.out.print("Enter the input file : ");
            Scanner inFileDir = new Scanner(System.in);
            String inputFileDirectory = inFileDir.nextLine();
            File inputFile = new File(inputFileDirectory);

            System.out.print("Enter the output file name : ");
            Scanner outFileDir = new Scanner(System.in);
            String outputFileDirectory = outFileDir.nextLine();

            System.out.print("Enter the scale factor : ");
            Scanner scaleVal = new Scanner(System.in);
            double                scaleFactor                =
Double.parseDouble(scaleVal.nextLine());


                        imageRescale(inputFile,    outputFile,
scaleFactor);

      }
}
```

CODE PSNR

```
import java.util.Scanner;
import java.awt.Color;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import java.text.DecimalFormat;

class MSE_PSNR
{
    public static void main(String[] args)
    {
      String Encrypt;
      BufferedImage GambarAsli,InterpolasiGambar;
      int width;
      int height;
      double mse = 0,mse_r = 0,mse_g = 0,mse_b = 0;
      double psnr = 0,psnr_r = 0,psnr_g = 0,psnr_b = 0;
      double max_rc = 0,max_gc = 0,max_bc = 0;
      double max_rs = 0,max_gs = 0,max_bs = 0;
      DecimalFormat dfmse = new DecimalFormat("#.##############");
      DecimalFormat dfpsnr = new DecimalFormat("#.##");
      try
      {

          File image1 = new File("santuy.jpg");
          File image2 = new File("santuy_bicubic.jpg");
          GambarAsli = ImageIO.read(image1);
          InterpolasiGambar = ImageIO.read(image2);
          width = GambarAsli.getWidth();
          height = GambarAsli.getHeight();
          for(int i=0;i<height;i++)
          {
              for(int j=0;j<width;j++)
              {
                  Color c = new Color(GambarAsli.getRGB(j,i));
                                                  Color   s   =   new
Color(InterpolasiGambar.getRGB(j,i));
                  double redc = (double)(c.getRed());
                  double reds = (double)(s.getRed());
                  double greenc = (double)(c.getGreen());
                  double greens = (double)(s.getGreen());
                  double bluec = (double)(c.getBlue());
                  double blues = (double)(s.getBlue());
                  if(redc>max_rc)
                  {
                    max_rc=redc;
                  }
                  if(greenc>max_gc)
                  {
                    max_gc=greenc;
```

```
                    }
                    if(bluec>max_bc)
                    {
                      max_bc=bluec;
                    }
                    if(reds>max_rs)
                    {
                      max_rs=reds;
                    }
                    if(greens>max_gs)
                    {
                      max_rs=reds;
                    }
                    if(blues>max_bs)
                    {
                      max_rs=reds;
                    }
                    mse_r=mse_r+Math.pow((reds-redc),2);
                    mse_g=mse_g+Math.pow((greens-greenc),2);
                    mse_b=mse_b+Math.pow((blues-bluec),2);

                  }
                }
                mse_r = mse_r/(width*height);
                mse_g = mse_g/(width*height);
                mse_b = mse_b/(width*height);
                mse = (mse_r+mse_g+mse_b)/3;
                                                psnr_r    =    10.0    *
logbase10(Math.pow(Math.max(max_rc,max_rs), 2) / mse_r);
                                                psnr_g    =    10.0    *
logbase10(Math.pow(Math.max(max_gc,max_gs), 2) / mse_g);
                                                psnr_b    =    10.0    *
logbase10(Math.pow(Math.max(max_bc,max_bs), 2) / mse_b);
                psnr = (psnr_r+psnr_g+psnr_b)/3;

                System.out.println("MSE R : "+dfmse.format(mse_r)+" MSE
G : "+dfmse.format(mse_g)+" MSE B : "+dfmse.format(mse_b)+" MSE =
"+dfmse.format(mse));
                System.out.println("Max R : "+Math.max(max_rc,max_rs)+"
Max    G    :    "+Math.max(max_gc,max_gs)+"    Max    B    :
"+Math.max(max_bc,max_bs));
                System.out.println("PSNR R : "+dfpsnr.format(psnr_r)+"
dB   PSNR   G   :   "+dfpsnr.format(psnr_g)+"   dB   PSNR   B   :
"+dfpsnr.format(psnr_b)+" dB PSNR = "+dfpsnr.format(psnr)+" dB");

            }
            catch(IOException e)
            {}
        }
```

```
 public static double logbase10(double x)
    {
        return Math.log(x) / Math.log(10);
    }

}
```

UNICHECK

Submission author:
15k10049 FEBRI BADAI DIRGANTARA

Check ID:
16014579

Check date:
17.01.2020 23:48:40 GMT+0

Check type:
Doc vs Internet + Library

Report date:
18.01.2020 00:20:49 GMT+0

User ID:
29118

Unika SOEGIJAPRANATA
Talenta pro patria et humanitate

File name: Project Report scan.docx

File ID: 20317459 Page count: 13 Word count: 2932 Character count: 17745 File size: 38.57 KB

3.21% Matches

Highest match: 2.8% with library source. File ID: 13401556

No Internet Sources Found

3.21% Library matches 48 ..... Page 15

0% Quotes

No quotes found

0% Exclusions

No exclusions found

Replacement

No replaced characters found