Appendix F

Frame Difference/ Fixation Maps

Pixels have a red, green and blue value that lies between 0 (black) and 255 (white). By comparing the different between the pixel values of two frame images (640x480) we can create images to: i) determine areas of movement in video frames (the distance between two images), ii) measure the difference between frames as a result of video manipulation (frame rate variation, error, delay, RoI manipulation); iii) analyse the difference between eye-based and content-based RoI information; and iv) produce RoI fixation maps. The following code accordingly creates and manipulates Jpeg images via manipulation of pixel values.

Package: definingroi

pixelData.java – object contains single pixel value

```
/* pixelData is class that facilitates the easy manipulation
of pixel information. Each pixel defines the content of a
pixelData object, therefore many hundred-thousand
objects are created for each image. */

package definingroi;
public class pixelData {
/* pixelData objects contain x, y coordinate values and
red, green and blue pixel values. */
   private int x;
   private int y;
   private int red;
   private int green;
   private int blue;

  public pixelData(int x_value, int y_value, int red_value,
int green_value, int blue_value)
//assign values from pixel at object creation
  {
    x = x_value;
    y = y_value;
    red = red_value;
    green = green_value;
    blue = blue_value;
  }

  private void setXValue(int x_value)
  {
    x = x_value;
// allows the x coordinate to be manipulated
  }

  public int getXValue()
  {
    return x;
// allows the x coordinate to be monitored

  }

  private void setYValue(int y_value)
  {
    y = y_value;
// allows the y coordinate to be manipulated
  }

  public int getYValue()
  {
    return y;
// allows the x coordinate to be monitored

  }

  private void setRedValue(int red_value)
  {
   red = red_value;
//allows the red value to be manipulated
  }

 public int getRedValue()
  {
   return red;
// allows the red value to be monitored
  }

 private void setGreenValue(int green_value)
  {
   green = green_value;
//allows the green value to be manipulated
  }

  public int getGreenValue()
  {
    return green;
// allows the green value to be monitored
  }

  private void setBlueValue(int blue_value)
   {
    blue = blue_value;
// allows the blue value to manipulated
   }

   public int getBlueValue()
   {
    return blue;
// allows the blue value to be monitored
   }
  public void description()
  {
    System.out.println("X:"+x+" Y:"+y+" red:"+red+"
green:"+ green +" blue:"+blue);
// allows the object content to be displayed
  }

}
```

createImage.java – allows creation of a JPEG image from pixel values

```
package definingroi;

/* createImage creates an image (prefix.jpg) that is
defined by an array list of pixelData objects. The image
size is set to 640*480 (which is the size of the extracted
frame images). The location of the output image is
determined as being C:\ temp\ output\ + prefix + .jpg –
where prefix is the image name, yet may be changed as
needed. */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.util.*;
import java.net.URL;
import java.io.*;
import com.sun.image.codec.jpeg.*;

public class createImage extends JFrame {
  Image newImage;
  class ImagePanel extends JPanel {

    public ImagePanel() {
    }
    public void paintComponent(Graphics g) {
      super.paintComponent(g) ;
      g.drawImage(newImage, 10, 10, 650, 490, this);
      }
    }
  }
// We use a JFrame to facilitate manipulation, however
the frame is not made visible and forced to close to
memory problems were identified. */

  public createImage(ArrayList inputdata, int prefix) {
    setTitle("Input Image");
    setSize(700, 500);

    Container contentPane = getContentPane();
    try {

      int[] pixels = new int[640 * 480];
// pixels is an array that contains pixelData objects

      for (int i = 0; i < (640 * 480); i++) {
// for all pixels in the input image – get pixel values
        int red = ((pixelData)
inputdata.get(i)).getRedValue();
        int blue = ((pixelData)
inputdata.get(i)).getBlueValue();
        int green = ((pixelData)
inputdata.get(i)).getGreenValue();
        int alpha = 255;
// Combine the RGB and alpha values
//place value in pixels array
        pixels[i++] = (alpha << 24) | (red << 16)
          | (green << 8) | blue;
      }

      newImage = createImage(new
MemoryImageSource(640, 480, pixels, 0, 640));
repaint();
//create and display the new image
/* The following code creates an output image from the
new image.*/
      BufferedImage outImage = new BufferedImage(640,
480, BufferedImage.TYPE_INT_RGB);
      Graphics2D graphics2D =
outImage.createGraphics();
graphics2D.setRenderingHint(RenderingHints.KEY_IN
TERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BILINE
AR);
      graphics2D.drawImage(newImage, 0, 0, 640, 480,
null);
      BufferedOutputStream out = new
BufferedOutputStream(new
FileOutputStream("C:\\temp\\output\\"+prefix+".jpg
"));
// define output location
      JPEGImageEncoder encoder =
JPEGCodec.createJPEGEncoder(out);
      JPEGEncodeParam param = encoder.
        getDefaultJPEGEncodeParam(outImage);
      int quality = 50;
// 50 out of 100 quality
      quality = Math.max(0, Math.min(quality, 100));
      param.setQuality( (float) quality / 100.0f, false);
      encoder.setJPEGEncodeParam(param);
      encoder.encode(outImage);
      System.out.println("Done.");
      contentPane.add(new ImagePanel());
    }
/* compress image as JPEG – use of BMP would be
practically unrealistic */
    catch (Exception e) {
      System.err.println(e.getMessage());
    }

    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
// close the Window or problems occur
      }
    });
  }
}
```

frameDifference.java – compares the pixel difference between to JPEG images

Difference between consecutive frames

```
package definingroi;
/* Defining the difference between two frames is essential to our research. Accordingly, frameDifference compares the pixel values of two pictures and creates a image from the pixel difference. */
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.util.*;
import java.net.URL;
import java.io.*;
import com.sun.image.codec.jpeg.*;

public class frameDifference
    extends JFrame {
  Image compImage1;
  Image compImage2;
// two images defined
  Image DifferenceImage;
// difference image defined
  ArrayList FirstImage = new ArrayList();
  ArrayList SecondImage = new ArrayList();
//Two arrayLists created to store the information from the two pictures.

  class ImagePanel extends JPanel {
    public ImagePanel() {
    }

    public void paintComponent(Graphics g) {
      super.paintComponent(g) ;
      g.drawImage(compImage1, 10, 10, 160, 120, this);
      g.drawImage(compImage2, 10, 170, 160, 120, this);
      g.drawImage(DifferenceImage, 180, 10, this);
      /* defines the layout of the images */
    }
    }
  }

  public void handlesinglepixel(int x, int y, int pixel, int
  image) {
 /* takes a single pixel value and transfers the details to a pixelData object */
    int alpha = (pixel >> 24) & 0xff;
    int red = (pixel >> 16) & 0xff;
    int green = (pixel >> 8) & 0xff;
    int blue = (pixel) & 0xff;
    pixelData pg = new pixelData(x, y, red, green, blue);
    if (image == 1){FirstImage.add(pg);}
// if image value is 1 then add to FirstImage arraylist
    else if (image ==2) {SecondImage.add(pg);}
// if image value is 2 then add to SecondImage arraylist
  }

  public void handlepixels(Image img, int x, int y, int w,
int h, int image) {
/* provides pixels information for an image */
    int[] pixels = new int[w * h];
/* an array that contains information for about all pixels in an image */
    PixelGrabber pg = new PixelGrabber(img, x, y, w, h,
pixels, 0, w);
//pg grabs the pixel values for an image
    try {
      pg.grabPixels();
    }
    catch (InterruptedException e) {
      System.err.println("interrupted waiting for pixels!");
      return;
    }
    if ( (pg.getStatus() & ImageObserver.ABORT) != 0) {
      System.err.println("image fetch aborted or errored");
      return;
    }
    for (int j = 0; j < h; j++) {
      for (int i = 0; i < w; i++) {
// for all pixels in the image
        handlesinglepixel(x + i, y + j, pixels[j * w + i],
image);
// extract pixel values for all pixels of image
      }
    }
  }

  public frameDifference(FileDialog inputFileName,
FileDialog inputFileName2, int Frame) {
    setTitle("Input Image");
    setSize(840, 530);

    Container contentPane = getContentPane();

    String fileName1, fileName2; //selected input file

    String Prefix1, Prefix2;
    Prefix1 = inputFileName.getFile().substring(0, 4);
    Prefix2 = inputFileName2.getFile().substring(0, 4);
// define Prefix, which defines the created image address

    fileName1 = Prefix1+Frame+").jpg";
    fileName2 = Prefix2+Frame+").jpg";
//image file name are defined

    System.out.println("Selected Frame One: " +
fileName1);
    System.out.println("Selected Frame Two: " +
fileName2);

    try {
      File imageFile1 = new
File(inputFileName.getDirectory() + fileName1);
// creates a new image – file 1
      File imageFile2 = new
File(inputFileName2.getDirectory() + fileName2);
// creates a new image – file 2
      compImage1 =
getToolkit().getImage(imageFile1.toURL());
// defines compImage 1
      compImage2 =
getToolkit().getImage(imageFile2.toURL());
// defines compImage 2
      repaint();
// display the images in the image frame
      handlepixels(compImage1, 0, 0, 640, 480, 1);
      handlepixels(compImage2, 0, 0, 640, 480, 2);
/* Takes the images and adds the pixel values to the arrayList. */
```

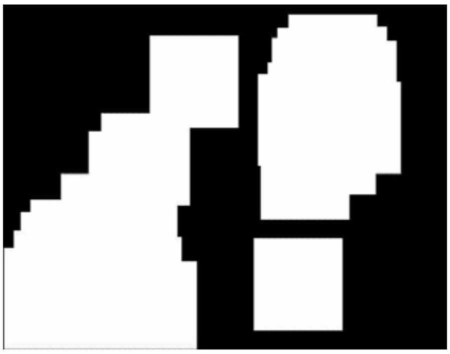
```
    System.out.println("EXPECTED: " + (640 * 480));
    System.out.println("First: " + FirstImage.size());
    System.out.println("Second: " + SecondImage.size());
// print the size of the arraylists
    ArrayList difference = new ArrayList();
    for (int i = 0; i < FirstImage.size(); i++) {
/* For all pixels – the following code calculates the
difference between the two images */
      int x;
      int y;
      int red;
      int green;
      int blue;
      x = ( (pixelData) FirstImage.get(i)).getXValue();
      y = ( (pixelData) FirstImage.get(i)).getYValue();
      if ( ( ( (pixelData) FirstImage.get(i)).getRedValue())
>
          ( ( (pixelData)
SecondImage.get(i)).getRedValue()))
        red = ( ( (pixelData)
FirstImage.get(i)).getRedValue() -
            ( ( (pixelData)
SecondImage.get(i)).getRedValue());
      else
        red = ( ( (pixelData)
SecondImage.get(i)).getRedValue() -
            ( ( (pixelData) FirstImage.get(i)).getRedValue());
      if ( ( ( (pixelData)
FirstImage.get(i)).getGreenValue() >
          ( ( (pixelData)
SecondImage.get(i)).getGreenValue()))
        green = ( ( (pixelData)
FirstImage.get(i)).getGreenValue() -
            ( ( (pixelData)
SecondImage.get(i)).getGreenValue());
      else
        green = ( ( (pixelData)
SecondImage.get(i)).getGreenValue() -
            ( ( (pixelData)
FirstImage.get(i)).getGreenValue());
      if ( ( ( (pixelData) FirstImage.get(i)).getBlueValue())
>
          ( ( (pixelData)
SecondImage.get(i)).getBlueValue()))
        blue = ( ( (pixelData)
FirstImage.get(i)).getBlueValue() -
            ( ( (pixelData)
SecondImage.get(i)).getBlueValue());
      else
        blue = ( ( (pixelData)
SecondImage.get(i)).getBlueValue() -
            ( ( (pixelData) FirstImage.get(i)).getBlueValue());
      pixelData pg = new pixelData(x, y, red, green, blue);
      difference.add(pg);
    }
    System.out.println("DIFFERENCE: " +
difference.size());
// print the size of Difference arrayList
    int[] pixels = new int[640 * 480];
/* Array created to contain pixels difference information.
Values taken from arrayList and added to array. */
    for (int i = 0; i < (640 * 480); i++) {
      int redDifference = ((pixelData)
difference.get(i)).getRedValue();
      int blueDifference = ((pixelData)
difference.get(i)).getBlueValue();
      int greenDifference = ((pixelData)
difference.get(i)).getGreenValue();
      int alphaDifference = 255;
      // Combine the RGB and alpha values
      pixels[i++] = (alphaDifference << 24) |
(redDifference << 16)
          | (greenDifference << 8) | blueDifference;
    }
/* create an image from the difference array and refresh
the image in the image window */
    DifferenceImage = createImage(new
MemoryImageSource(640, 480, pixels, 0, 640));
    repaint();
/* The following code creates an output image from the
new image.*/
    BufferedImage outImage = new BufferedImage(640,
480, BufferedImage.TYPE_INT_RGB);
    Graphics2D graphics2D =
outImage.createGraphics();
graphics2D.setRenderingHint(RenderingHints.KEY_IN
TERPOLATION,
RenderingHints.VALUE_INTERPOLATION_BILINE
AR);
    graphics2D.drawImage(DifferenceImage, 0, 0, 640,
480, null);
    BufferedOutputStream out = new
BufferedOutputStream(new
FileOutputStream("F:\\TEMP\\EYE_VID\\SP\\diff
_"+Frame+".jpg"));
// define output location
    JPEGImageEncoder encoder =
JPEGCodec.createJPEGEncoder(out);
    JPEGEncodeParam param = encoder.
        getDefaultJPEGEncodeParam(outImage);
    int quality = 50;
// 50 out of 100 quality
    quality = Math.max(0, Math.min(quality, 100));
    param.setQuality( (float) quality / 100.0f, false);
    encoder.setJPEGEncodeParam(param);
    encoder.encode(outImage);
    System.out.println("Done.");
   contentPane.add(new ImagePanel());
  }
/* compress image as JPEG – use of BMP would be
practically unrealistic */
  catch (Exception e) {
    System.err.println(e.getMessage());
  }

  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
      System.exit(0);
// close the Window or problems occur
    }
  });
 }
}
```

testframeDifference.java – to facilitate use of frameDifference.java

```
package definingroi;
/* The testframeDifference class is a runable program to
test the frameDifference class. */
import java.util.*;
import java.io.*;
import java.awt.*;

class testframeDifference {
  public static void main(String[] args) {
/* The following code allows the user to define two
image files. */
    Frame dialogInputFrame = new Frame(); //creates a
new frame
    FileDialog inputFileName = new
FileDialog(dialogInputFrame, "FIRST COMPARISON
DIRECTORY");
    inputFileName.setVisible(true);
    String selectedItem; //selected input file
    do {
      selectedItem = inputFileName.getDirectory() +
inputFileName.getFile();
      System.out.println("Selected File: " + selectedItem);
    }
    while (selectedItem == null);
    dialogInputFrame.dispose(); // remove frame

    Frame dialogInputFrame2 = new Frame(); //creates a
new frame
    FileDialog inputFileName2 = new
FileDialog(dialogInputFrame2, "SECOND
COMPARISON DIRECTORY");
    inputFileName2.setVisible(true);
    String selectedItem2; //selected input file
    do {
      selectedItem2 = inputFileName2.getDirectory() +
inputFileName2.getFile();
      System.out.println("Selected File: " + selectedItem2);
    }
    while (selectedItem2 == null);
    dialogInputFrame2.dispose(); // remove frame
```

```
    try { // loads data from file into inputVector
      for (int i = 0; i < 918; i++) {
/* Change the maximum value of i depending on the
video being compared. */
        frameDifference test = new
frameDifference(inputFileName, inputFileName2, i);
/* FileDialog inputFileName, FileDialog,
inputFileName2, int Frame – creates an images for all
files.*/
        test.dispose();
      }
    }
    catch (Exception e) {
      System.err.println(e.getMessage());
    }
  }
}
```

RoIMaps.java – to produce RoI fixation Maps from RoI scripts



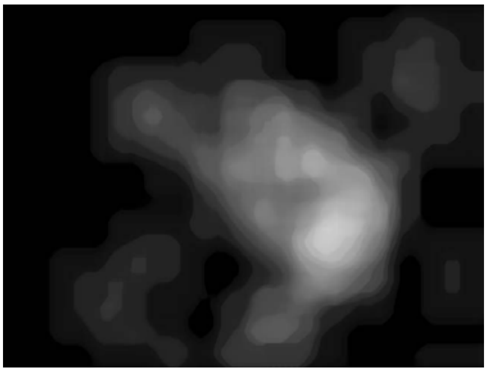
Region of Interest Map
(foreground = 0 / background = 254)

```
package definingroi;
/* RoIMaps takes eye tracker data and manipulates the
image accordingly: 0 (black) - no eye tracker
information; 254 (white) – eye tracking data provided */
import java.io.*;
import java.awt.*;
import java.util.*;
import java.lang.*;

public class RoIMaps {
  public static void main(String[] args) {
    int x1 = 0, x2 = 0, y1 = 0, y2 = 0;
    int currentFrame = 0;
    int max = 0;

    int[][] pixels = new int[640][480];
    for (int i = 0; i < 479; i++) {
      for (int j = 0; j < 639; j++) {
        pixels[j][i] = 0;
      }
    }
// Array defined and contents filled
    Frame dialogInputFrame = new Frame();
// Creates a frame within in the program
    FileDialog inputFileName = new
FileDialog(dialogInputFrame,"Open 'ROI DATA' file ");
    inputFileName.setVisible(true);
    String selectedItem;
//Declaring value for the input file
    do {
      selectedItem = inputFileName.getDirectory() +
inputFileName.getFile();
      //System.out.println(SELECTED FILE: " +
selectedItem); //Display output file
    }
```

```
    while (selectedItem == null);
    dialogInputFrame.dispose();
// Remove frame, once parameters are met
    Vector inputVector = new Vector();
//creates a vector - to put allow input of text file
    try {
      FileReader inputFile = new FileReader(selectedItem);
      BufferedReader reader = new
BufferedReader(inputFile);
      String bufferInput;
      while ( (bufferInput = reader.readLine()) != null)
        inputVector.add(bufferInput);
    }
    catch (Exception e) {
      System.err.println(e.getMessage());
    }
//For the number of lines in the text file
    int numOfLines = inputVector.size();
    int frameNumber = 0;
    int linecounter = 0;
    int currentbufferposition = 0;
    int internal = 0;
    int external = 0;
// for the number of lines in the text file
    for (int lineNum = 0; lineNum < numOfLines;
lineNum++) {
      String inputString;
      inputString = (String) inputVector.get(lineNum);
      if (inputString.startsWith("f")) {
// if line starts with an F tag (frame)
        String lineToken;
        StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
         lineToken = lineTokenizer.nextToken();
        currentbufferposition =
Integer.parseInt(lineTokenizer.nextToken());
// currentbufferposition defines the frame number
        ArrayList picture = new ArrayList();
        for (int i = 0; i < 479; i++) {
          for (int j = 0; j < 639; j++) {
            if (pixels[j][i] > max)
              max = pixels[j][i];
          }
        }
// max = maximum value

        for (int i = 0; i < 480; i++) {
          for (int j = 0; j < 640; j++) {
            int red = (pixels[j][i]) * (254);
            int green = (pixels[j][i]) * (254);
            int blue = (pixels[j][i]) * (254);

// value in array multiplied by 254 to give visible split
```

```
        pixelData pg = new pixelData(j, i, red, green,
blue);
        picture.add(pg);
/* for each pixel value add pixelData to picture arraylist,
therefore producing a full picture definition */

       }
      }
//re-initialise max variable and pixels array
     max = 0;
     for (int i = 0; i < 479; i++) {
       for (int j = 0; j < 639; j++) {
         pixels[j][i] = 0;
       }
     }
     try {
// loads data from file into inputVector
       createImage test = new createImage(picture,
frameNumber);
       test.dispose();
     }
     catch (Exception e) {
       System.err.println(e.getMessage());
     }
    }
    else if (inputString.startsWith("c")) {
//XYCoordinate data
     String lineToken;
     StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
     //System.out.println("NEXT LINE");
     lineToken = lineTokenizer.nextToken();
     //System.out.println("SHOULD BE C: " +
lineToken);
//extracts data from the XYcoordinate line
     x1 = (int)
Integer.parseInt(lineTokenizer.nextToken());
     y1 = (int)
Integer.parseInt(lineTokenizer.nextToken());
     x2 = (int)
Integer.parseInt(lineTokenizer.nextToken());
     y2 = (int)
Integer.parseInt(lineTokenizer.nextToken());

     if (x1 > 639)
       x1 = 639;
     if (y1 > 478)
       y1 = 478;
     if (x1 < 1)
       x1 = 1;
     if (y1 < 1)
       y1 = 1;
     if (x2 > 639)
       x2 = 639;
     if (y2 > 479)
       y2 = 479;
     if (x2 < 1)
       x2 = 1;
     if (y2 < 1)
       y2 = 1;
/* If the value is incorrect then the value is reduced to
prevent image manipulation errors. */
     frameNumber = currentFrame +
currentbufferposition;
     System.out.println("FrameNumber:" +
frameNumber);

/* for all pixels within the given coordinates, define the
value as one - i.e. eyetracker data present. */
     for (int i = y1; i < y2; i++) {
       for (int j = x1; j < x2; j++) {
         pixels[j][i] = 1;
/* Can be replaced by pixels[j][i]= (pixels[j][i]) + 1 – see
fixationMaps.java */
       }
       }

       }
     else if (inputString.startsWith("h")) {}
/* Extracts information about frame rate, internal frame
rate and external frame rate */
     else if (linecounter == 0) {
       String lineToken;
       StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
       currentFrame =
Integer.parseInt(lineTokenizer.nextToken());
//current frame rate
       linecounter++;
     } else if (linecounter == 1) {
       String lineToken;
       StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
       internal =
Integer.parseInt(lineTokenizer.nextToken());
//internal frame rate
       linecounter++;
     } else if (linecounter == 2) {
       String lineToken;
       StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
       external =
Integer.parseInt(lineTokenizer.nextToken());
//external frame rate
       linecounter = 0;
     }
    }
//define the maximum value
    for (int i = 0; i < 479; i++) {
      for (int j = 0; j < 639; j++) {
        if (pixels[j][i] > max)
          max = pixels[j][i];
      }
    }

    ArrayList picture = new ArrayList();
    for (int i = 0; i < 480; i++) {
      for (int j = 0; j < 640; j++) {
        int red = (pixels[j][i]) * (254);
        int green = (pixels[j][i]) * (254);
        int blue = (pixels[j][i]) * (254);
/* 254 can be replaced by 254/max – see
fixationMap.java */
        pixelData pg = new pixelData(j, i, red, green, blue);
// create pixelData object
        picture.add(pg);
// add pixel values to the pictures arrayList
      }
    }

    max = 0;
    for (int i = 0; i < 479; i++) {
      for (int j = 0; j < 639; j++) {
        pixels[j][i] = 0;
      }
    }
// blank the pixels array

    try { // loads data from file into inputVector
      createImage test = new createImage(picture,
frameNumber);
// creates a picture from the picture arrayList
      test.dispose();
    }
    catch (Exception e) {
      System.err.println(e.getMessage());
    }
   }
}
```

fixationMaps.java – to facilitate use of RoIMaps.java



A filtered fixation Map

```
package definingroi;
/* FixationMaps takes eye tracker data and manipulates
the image accordingly: 0 (black) - no eye tracker
information; 254 (white) – eye tracking hotspot. The
colours in between signify less important visual areas. */
import java.io.*;
import java.awt.*;
import java.util.*;
import java.lang.*;

public class fixationMaps {

  public static void main(String[] args) {
    int x1 = 0, x2 = 0, y1 = 0, y2 = 0;
    int currentFrame = 0;
    int max = 0;
    int[][] pixels = new int[640][480];
    for (int i = 0; i < 479; i++) {
      for (int j = 0; j < 639; j++) {
        pixels[j][i] = 0;
      }
    }
// Array defined and contents filled
    Frame dialogInputFrame = new Frame();
//Creates a frame within in the program
    FileDialog inputFileName = new
FileDialog(dialogInputFrame,"Open 'ROI DATA' file ");
    inputFileName.setVisible(true);
    String selectedItem;
//Declaring value for the input file
    do {
      selectedItem = inputFileName.getDirectory() +
inputFileName.getFile();
      //System.out.println(SELECTED FILE: " +
selectedItem); //Display output file
    }
    while (selectedItem == null);
    dialogInputFrame.dispose();
// Remove frame, once parameters are met
    Vector inputVector = new Vector();
//creates a vector - to put allow input of text file
    try {
      FileReader inputFile = new FileReader(selectedItem);
      BufferedReader reader = new
BufferedReader(inputFile);
      String bufferInput;
      while ( (bufferInput = reader.readLine()) != null)
        inputVector.add(bufferInput);
    }
    catch (Exception e) {
      System.err.println(e.getMessage());
    }
//For the number of lines in the text file
    int numOfLines = inputVector.size();
    int frameNumber = 0;
    int linecounter = 0;
    int currentbufferposition = 0;
    int internal = 0;
    int external = 0;
    //for the number of lines in the text file set-up
tokeniser and
    for (int lineNum = 0; lineNum < numOfLines;
lineNum++) {
      String inputString;
      inputString = (String) inputVector.get(lineNum);
      //System.out.println(inputString);

      if (inputString.startsWith("f")) {
// if line starts with an F tag  (frame)
        String lineToken;
        StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
        //System.out.println("NEXT LINE");
        lineToken = lineTokenizer.nextToken();
        //System.out.println("SHOULD BE F: " +
lineToken);
        currentbufferposition =
Integer.parseInt(lineTokenizer.nextToken());
        //System.out.println("current buffer position" +
currentbufferposition);
// currentbufferposition defines the frame number
        ArrayList picture = new ArrayList();
        for (int i = 0; i < 479; i++) {
          for (int j = 0; j < 639; j++) {
            if (pixels[j][i] > max)
              max = pixels[j][i];
          }
        }
// max = maximum value

        for (int i = 0; i < 480; i++) {
          for (int j = 0; j < 640; j++) {
            int red = (pixels[j][i]) * (254/max);
            int green = (pixels[j][i]) * (254/max);
            int blue = (pixels[j][i]) * (254/max);

// value in array multiplied by 254 to give visible split
            pixelData pg = new pixelData(j, i, red, green,
blue);
            picture.add(pg);
/* for each pixel value add pixelData to picture arraylist,
therefore producing a full picture definition */
          }
        }
//re-initialise max variable and pixels array
        max = 0;
        for (int i = 0; i < 479; i++) {
          for (int j = 0; j < 639; j++) {
            pixels[j][i] = 0;
          }
        }

        try {
// loads data from file into inputVector
          createImage test = new createImage(picture,
frameNumber);
          test.dispose();
        }
        catch (Exception e) {
          System.err.println(e.getMessage());
        }
      }
      else if (inputString.startsWith("c")) {
//XYCoordinate data
        String lineToken;
        StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
        //System.out.println("NEXT LINE");
        lineToken = lineTokenizer.nextToken();
        //System.out.println("SHOULD BE C: " +
lineToken);
//extracts data from the XYcoordinate line
        x1 = (int) (((
Integer.parseInt(lineTokenizer.nextToken()) + 64) *640 )
/ 10000) - 64;
        y1 = (int) (((
Integer.parseInt(lineTokenizer.nextToken()) + 64) *480 )
/ 10000)- 64;
        x2 = (int) (((
Integer.parseInt(lineTokenizer.nextToken()) - 64) *640 )
/ 10000) + 64;
        y2 = (int) (((
Integer.parseInt(lineTokenizer.nextToken()) - 64) *480 )
/ 10000) + 64;

        if (x1 > 639)
          x1 = 639;
        if (y1 > 478)
          y1 = 478;
        if (x1 < 1)
          x1 = 1;
        if (y1 < 1)
          y1 = 1;
        if (x2 > 639)
          x2 = 639;
        if (y2 > 479)
          y2 = 479;
        if (x2 < 1)
          x2 = 1;
        if (y2 < 1)
          y2 = 1;
/* If the value is incorrect then the value is reduced to
prevent image manipulation errors.  */

       frameNumber = currentFrame +
currentbufferposition;
        System.out.println("FrameNumber:" +
frameNumber);

/* for all pixels within the given coordinates, define the
value as one  - i.e. eyetracker data present. */

//external area
        for (int i = y1; i < y2; i++) {
          for (int j = x1; j < x2; j++) {
            pixels[j][i] = (pixels[j][i] +1);
          }
        }

//internal area mapping

        for (int i = (y1 + 32); i < (y2 - 32); i++) {
          for (int j = (x1 + 32); j < (x2 - 32); j++) {
            pixels[j][i] = (pixels[j][i] + 1);
          }
        }

      }
      else if (inputString.startsWith("h")) {}
/* Extracts information about frame rate, internal frame
rate and external frame rate */
      else
      if (linecounter == 0) {
        String lineToken;
        StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
        currentFrame =
Integer.parseInt(lineTokenizer.nextToken());
//current frame rate
        //System.out.println("CURRENTFRAME: " +
currentFrame);
        linecounter++;
      }
      else if (linecounter == 1) {
        String lineToken;
        StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
        //System.out.println("NEXT LINE"); ;
        internal =
Integer.parseInt(lineTokenizer.nextToken());
//internal frame rate
        linecounter++;
      }
      else if (linecounter == 2) {
        String lineToken;
        StringTokenizer lineTokenizer = new
StringTokenizer(inputString, "\t");
        //System.out.println("NEXT LINE"); ;
        external =
Integer.parseInt(lineTokenizer.nextToken());
//external frame rate
        linecounter = 0;
      }
    }
//define the maximum value
    for (int i = 0; i < 479; i++) {
      for (int j = 0; j < 639; j++) {
        if (pixels[j][i] > max)
          max = pixels[j][i];
      }
    }

    ArrayList picture = new ArrayList();
    for (int i = 0; i < 480; i++) {
      for (int j = 0; j < 640; j++) {
        int red = (pixels[j][i]) * (254/max);
        int green = (pixels[j][i]) * (254/max);
        int blue = (pixels[j][i]) * (254/max);
        pixelData pg = new pixelData(j, i, red, green, blue);
// create pixelData object
        picture.add(pg);
// add pixel values to the pictures arrayList
      }
    }

    max = 0;
    for (int i = 0; i < 479; i++) {
      for (int j = 0; j < 639; j++) {
        pixels[j][i] = 0;
      }
    }
// blank the pixels array

    try { // loads data from file into inputVector
      createImage test = new createImage(picture,
frameNumber);
// creates a picture from the picture arrayList
      test.dispose();
    }
    catch (Exception e) {
      System.err.println(e.getMessage());
    }
    //System.out.println("PICTURE: " + picture.size());
  }
}
```