Appendix M

RoI Adaptation Software

The following code was used, in conjunction with RoI scripts (see Appendix L), to produce RoID video. I would like to give my deepest thanks Jimin Patel who worked with me on this code.

Package roimplayer

PreAccessCodec.java

```
/**
 * Title:        Region of Interest Media Player<p>
 * Description:  <This class deals with accessing
individual decoded frames
 *               during the processor operation.p>
 * Copyright:    Copyright (c) Stephen Gulliver<p>
 * Company:      Brunel University<p>
 * @authors Stephen Gulliver / Jimin Patel
 * @version 1.0
 * Code contained within this application has been
extracted and adapted from:
 *
 * *** Accessing Individual Decoded Frames ***
 * *** Availble at:  http://java.sun.com  ***
 *
 * *** Building your own media player, Jeff Friesen
(2001):
 * *** Availble at: http://www.informit.com ***
 *
 * *** Lab 8 of MM Applications 2003 (Harry Aguis,
Brunel University)
 * *** Availble at: http://disc.brunel.ac.uk***
 *
 *
 */

package roimplayer;

// Import of Java libraries

import java.awt.*;
import javax.media.*;
import javax.media.control.*;
import javax.media.Format;
import javax.media.format.*;
import javax.swing.*;
import java.io.*;
import java.awt.image.*;

import java.awt.Dimension;
import java.awt.event.*;
import java.awt.Image.*;
import java.awt.Dimension;
import java.awt.image.renderable.*;
import java.io.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import com.sun.image.codec.jpeg.*;

import javax.media.*;
import javax.media.format.*;
import javax.media.format.VideoFormat;
import javax.media.util.*;
import javax.media.util.BufferToImage;
import javax.media.control.*;
import javax.media.protocol.*;
import javax.media.control.FormatControl;

   public class PreAccessCodec implements Codec {
/* PreAccess facilitates the manipulation of  different
video formats frames */
       void accessFrame(Buffer frame) {
//Frame access and output details
         long frameNum = frame.getSequenceNumber();
         long t =
(long)(frame.getTimeStamp()/10000000f);
          System.err.println("Pre: frame #: " +
frame.getSequenceNumber() +
                     ", time: " + ((float)t)/100f +
                     ", len: " + frame.getLength());
       }
/*  The following code for allow multimple videos
codecs to function correctly - All video formats are
considered for input and output. */

// supports all input /output formats
       protected Format supportedIns[] = new Format [] {
          new VideoFormat(null)
       };
       protected Format supportedOuts[] = new Format []
       {
       new VideoFormat(null)
       };
       Format input = null, output = null;

       public String getName() {
          return "Pre-Access Codec";
       }

       public void open() {
       }

       public void close() {
       }

       public void reset() {
       }

// supports all output formats
       public Format [] getSupportedInputFormats() {
          return supportedIns;
       }

//determines content format
       public Format []
getSupportedOutputFormats(Format in) {
          if (in == null)
             return supportedOuts;
```

```
        else {
            Format outs[] = new Format[1];
            outs[0] = in;
            return outs;
        }
    }

//defines input format
    public Format setInputFormat(Format format) {
        input = format;
        return input;
    }

//defines input format
    public Format setOutputFormat(Format format) {
        output = format;
        return output;
    }

    public int process(Buffer in, Buffer out) {
// defines input / output formats and buffers
                accessFrame(in);

                Object data = in.getData();
                in.setData(out.getData());
                out.setData(data);
                // Input attributes are processed for the output
                out.setFormat(in.getFormat());
                out.setLength(in.getLength());
                out.setOffset(in.getOffset());
                return BUFFER_PROCESSED_OK;
            }

        // The following code defines the process controls
            public Object[] getControls() {
                return new Object[0];
            }

            public Object getControl(String type) {
                return null;
            }
ta e    }
```

ProcessRoI.java

```
/**
 * Title:        Region of Interest Media Player<p>
 * Description:  <p>
 * Copyright:    Copyright (c) Stephen Gulliver<p>
 * Company:      MSc Dissertation for Multimedia
Information Systems<p>
 * @authors Stephen Gulliver / Jimin Patel
 * @version 1.0
 * Code contained within this application has been
extracted and adapted from:
 *
 * *** Accessing Individual Decoded Frames ***
 * *** Availble at:  http://java.sun.com  ***
 *
 * *** Building your own media player, Jeff Friesen
(2001):
 * *** Availble at: http://www.informit.com ***
 *
 * *** Lab 8 of MM Applications 2003 (Harry Aguis,
Brunel University)
 * *** Availble at: http://disc.brunel.ac.uk***
 *
 */

package roimplayer;

// Import Java libraries

import java.awt.*;
import javax.media.*;
import javax.media.control.*;
import javax.media.Format;
import javax.media.format.*;
import javax.swing.*;
import java.io.*;
import java.awt.image.*;
import java.awt.image.FilteredImageSource;
import java.awt.Dimension;
import java.awt.event.*;
import java.awt.Image.*;
import java.awt.image.renderable.*;
import java.io.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import com.sun.image.codec.jpeg.*;
import javax.media.*;
import javax.media.format.*;
import javax.media.format.VideoFormat;
import javax.media.util.*;
import javax.media.util.BufferToImage;
import javax.media.control.*;
import javax.media.protocol.*;
import javax.media.control.FormatControl;

 public class PostAccessCodec extends PreAccessCodec
{
/* PostAccessCodec extracts single frames and converts
the frame to an image */
    Processor p;
    Object waitSync = new Object();
    boolean stateTransitionOK = true;
    private File file;

    public BufferToImage btoi = null;
    public ImageToBuffer itob = null;
    public Image frameImage = null;
    public BufferedImage tempFrame = null;
    public static BufferedImage tempFrame2 = null;
    private int counter = 0;
    private int counter2 = 100;
    private int tempCounter = 0;

            public PostAccessCodec() {
            supportedIns = new Format [] {
                new RGBFormat()
            };
        }
// Callback to access individual video frames.
        void accessFrame(Buffer frame) {
// Output the frame #, time & data length.
         long t = (long)(frame.getTimeStamp()/10000000f);

        System.err.println("Post: frame #: " +
frame.getSequenceNumber() +
                    ", time: " + ((float)t)/100f +
                    ", len: " + frame.getLength());

/* check if it this the first frame in the five frame count
and if so load the temp frame. Used in 5, 15 and 25 fps
video.*/
          if (counter == 0){
          try{

              btoi = new BufferToImage((VideoFormat)
frame.getFormat());
              frameImage = btoi.createImage(frame);
              tempFrame = new
BufferedImage(frameImage.getWidth(null),
frameImage.getHeight(null),
BufferedImage.TYPE_INT_RGB);

          } catch (Exception e){
```

```
            System.out.println("Error !");
           }
          }
         int x = 20;
             if (frame == null)
                frame.isDiscard();
             else{

/* If frame has contents - Converts the buffer into an
image and then a buffered image */
                btoi = new BufferToImage((VideoFormat)
frame.getFormat());
                Image img = btoi.createImage(frame);
                BufferedImage bi = new
BufferedImage(img.getWidth(null), img.getHeight(null),
BufferedImage.TYPE_INT_RGB);
                tempFrame2 = bi;
                Graphics2D g2 = bi.createGraphics();
                g2.drawImage(img, null, null);
                g2.drawString("test" + counter, 30, 30);
/* the following lines facilitates manipulation of the
image – see CropImageFilter */
                ImageFilter filter = new CropImageFilter(x,
20, 60, 60);
                FilteredImageSource fis = new
FilteredImageSource(bi.getSource(), filter);
                Graphics2D g1 =
tempFrame.createGraphics();
                g1.drawImage(tempFrame, 0, 0, null);
               counter ++;
// increment general frame counter
          }//end else

/* We use a five frame counter to facilitate reduction in
frame rate via the duplication of frames passed to the
tempFrame variable. Additional remainer values of
counter can be used to repeat duplicated frames*/

          if (counter%5 == 1){
            tempFrame = tempFrame2;
            System.out.println("temp Frame " +
tempCounter);
             tempCounter++;
          }
          Buffer finalImg = itob.createBuffer(tempFrame,
25f);

//finalImg defines the final image that is passed to frame
          Object data = finalImg.getData();
          frame.setData(data);


          // Copy the input attributes to the output
          frame.setFormat(finalImg.getFormat());
          frame.setLength(finalImg.getLength());
          frame.setOffset(finalImg.getOffset());
 } // end of PostAccessCodec



  public void saveJpeg(BufferedImage bi, int counter){
/* saves all frames as single jpeg files – uses counter
value to differenciate between frames */
             try{

                FileOutputStream fos = null;
                   String file = "C:\\JPEGS/myJpeg" +
counter + ".jpg";
// defines output location
                   fos = new FileOutputStream(file);
                   JPEGImageEncoder encoder =
JPEGCodec.createJPEGEncoder(fos);
                   encoder.encode(bi);
                   fos.flush();
                   fos.close();
             }
             catch(FileNotFoundException e) {
                   System.out.println(e);
             }
             catch(IOException ioe) {
                   System.out.println(ioe);
             }
             catch(Exception e){
                   System.out.println(e);
             } // end catch

  }


      public String getName() {
         return "Post-Access Codec";
      }
   }
```

WindowHandler.java

```
/**
 * Title:        Region of Interest Media Player<p>
 *
 * Description:  <Window Handler code deals with the
opening and closing of
 *               windows, here are a few examples
windowActivated, windowClosed,
 *               windowClosing, windowDeactivated,
windowDeiconified, windowIconified and
 *               windowOpened
 *
 *              The WindowHandler also handles the
deallocation of resources,
 *               failure to adhere to this will gradually use
your computers resources and could
 *               cause failure >
 *
 * Copyright:    Copyright (c) Stephen Gulliver<p>
 * Company:      MSc Dissertation for Multimedia
Information Systems<p>
 * @authors Stephen Gulliver / Jimin Patel
 * @version 1.0
 * Code contained within this application has been
extracted and adapted from:
 *
 * *** Accessing Individual Decoded Frames ***
 * *** Availble at:  http://java.sun.com  ***
 *
 * *** Building your own media player, Jeff Friesen
(2001):
 * *** Availble at: http://www.informit.com ***
 *
 * *** Lab 8 of MM Applications 2003 (Harry Aguis,
Brunel University)
 * *** Availble at: http://disc.brunel.ac.uk***
 *
 */

package roimplayer;

import java.awt.event.*;
import java.awt.*;

/*The class WindowHandler extends WindowAdapter
and deals with the closing of used windows*/

 public class WindowHandler extends WindowAdapter
{

  public void windowClosing(WindowEvent wevent) {
     ProcessRoi myWindow;
       myWindow = (ProcessRoi) wevent.getWindow();
```

```
/* Code to stop, close and deallocate the player and it's
associated resources*/

    if (myWindow.p !=null){
       myWindow.p.stop();
       myWindow.p.close();
       myWindow.p.deallocate();
      System.exit (0);
      }
```

SinglePlayer.java

```
/**
 * Title:        Region of Interest Media Player<p>
 * Description:  <The code contained within this class
executes a media player with the
 *               use of JMF>
 * Copyright:    Copyright (c) Stephen Gulliver <p>
 * Company:      MSc Dissertation for Multimedia
Information Systems<p>
 * @authors Stephen Gulliver / Jimin Patel
 * @version 1.0
 * Code contained within this application has been
extracted and adapted from:
 *
 * *** Accessing Individual Decoded Frames ***
 * *** Availble at:  http://java.sun.com  ***
 *
 * *** Building your own media player, Jeff Friesen
(2001):
 * *** Availble at: http://www.informit.com ***
 *
 * *** Lab 8 of MM Applications 2003 (Harry Aguis,
Brunel University)
 * *** Availble at: http://disc.brunel.ac.uk***
 *
 */
package roimplayer;
import javax.media.*;
import java.awt.*;
import java.awt.Frame;
import java.awt.event.*;
import javax.media.control.FramePositioningControl.*;
import javax.swing.JOptionPane;
import javax.swing.filechooser.*;
import javax.swing.*;

// Declarations,assignments and initialisation

public class singlePlayer extends Frame implements
// single Player creates and sets up player window
ActionListener, ControllerListener, ItemListener
{
  Player myPlayer;
  Component myVisualComp, myControlComp;
  boolean first_file = true, loop = false;
  String current_Directory;

  singlePlayer (String title)
  {
    super (title);
    addWindowListener
           (new WindowAdapter ()
            {
              public void windowClosing (WindowEvent
e)
              {
// User selected close from System menu.
// Call dispose to invoke windowClosed.
                dispose ();
              }
              public void windowClosed (WindowEvent
e)
              {
                if (myPlayer != null)
                   myPlayer.close ();
   myWindow.p.stop();
   myWindow.p.deallocate();
   myWindow.p.close();

   // Exit the program.
   System.out.println("PROGRAM EXITED ");
   System.exit(0);
  }
}
```

```
                   System.out.println("YOU ARE HAVE
EXITED THE MEDIA PLAYER ");
                   System.exit (0);
                 }
               });
//Options available from the menu bar
    Menu menu = new Menu ("Player Options");
    MenuItem menuitem = new MenuItem ("Open...");
    menuitem.addActionListener (this);
    menu.add (menuitem);
    menu.addSeparator ();
    menuitem = new MenuItem ("Start");
    menuitem.addActionListener (this);
    menu.add (menuitem);
    menu.addSeparator ();
    menuitem = new MenuItem ("Stop");
    menuitem.addActionListener (this);
    menu.add (menuitem);
    menu.addSeparator ();
/* Check Box Item, allows checking and unchecking of
loop function */
    CheckboxMenuItem cbox_mitem = new
    CheckboxMenuItem ("Loop", false);
    cbox_mitem.addItemListener (this);
    menu.add (cbox_mitem);
    menu.addSeparator ();
// adds separator for the menu list
    menuitem = new MenuItem ("Exit");
    menuitem.addActionListener (this);
    menu.add (menuitem);

    MenuBar menu_bar = new MenuBar ();
    menu_bar.add (menu);
    setMenuBar (menu_bar);

    setSize (250, 200);
    setLocation(440,200);
    setVisible (true);
  }

  public void actionPerformed (ActionEvent e)
  {
//Action performed when a function has been selected
    if (e.getActionCommand ().equals ("Start"))
    {
// Call to start video file
      myPlayer.start();
      return;
    }
    if (e.getActionCommand ().equals ("Stop"))
    {
// Call to stop video file
      myPlayer.stop();
      return;
    }

    if (e.getActionCommand ().equals ("Exit"))
    {
// Call dispose to invoke windowClosed.
      dispose ();
      return;
    }
//File dialog box setup, to open a file.
```

```
    FileDialog file_dialog = new FileDialog (this, "Open
File",
                            FileDialog.LOAD);
    file_dialog.setDirectory (current_Directory);
    file_dialog.show ();
// If user cancelled, exit.
    if (file_dialog.getFile () == null)
       return;
    current_Directory = file_dialog.getDirectory ();
    if (myPlayer != null)
       myPlayer.close ();
    try
    {
      myPlayer = Manager.createPlayer (new
MediaLocator
                                ("file:" +
                                 file_dialog.getDirectory () +
                                  file_dialog.getFile ()));
//Error checking for the media player.
    }
    catch (java.io.IOException e2)
    {
      System.out.println (e2);
      return;
    }
    catch (NoPlayerException e2)
    {
      System.out.println ("PLAY NOT FOUND.");
      return;
    }
    if (myPlayer == null)
    {
       System.out.println ("DIFFICULY CREATING
PLAYER.");
       return;
    }
    first_file = false;
    setTitle (file_dialog.getFile ());

    myPlayer.addControllerListener (this);
    myPlayer.prefetch ();
  }
/* Instances that may take place when a player is not
assigned.*/

  public void controllerUpdate (ControllerEvent e)
  {
    if (e instanceof ControllerClosedEvent)
    {
       if (myVisualComp != null)
       {
          remove (myVisualComp);
          myVisualComp = null;
       }
       if (myControlComp != null)
       {
          remove (myControlComp);
          myControlComp = null;
       }
       return;
    }
/*  End of media events to handle functionality such as
reseting of the player */
    if (e instanceof EndOfMediaEvent)
    {
       if (loop)
       {
          myPlayer.setMediaTime (new Time (0));
          myPlayer.start ();
       }
       return;
    }
    if (e instanceof PrefetchCompleteEvent)
    {
       myPlayer.start ();
       return;
    }
/* Once the player is realised, visual and control
components are added */
    if (e instanceof RealizeCompleteEvent)
    {
       myVisualComp = myPlayer.getVisualComponent
();
       if (myVisualComp != null)
          add (myVisualComp );
       myControlComp =
myPlayer.getControlPanelComponent ();
       if (myControlComp != null)
          add (myControlComp, BorderLayout.SOUTH);
       pack ();
    }
  }
 // Handling of the state changes within items

  public void itemStateChanged (ItemEvent e)
  {
    loop = !loop;
  }

  public static void main (String [] args)
  {
    new singlePlayer ("Media Player");
  }
}
```

RoIFrame.java

```
/**
 * Title:        Region of Interest Media Player<p>
 * Description:  <This class creates a processor and uses
that processor
 *               as a tool to embed region of interest
parameters
 *               During the processor's configured state, two
"pass-thru" codecs,
    *               PreAccessCodec and PostAccessCodec,
are set on the video track.
    *               These codecs are used to get access to
individual video frames
 *               of the media.>
 *
 * Copyright:    Copyright (c) Stephen Gulliver<p>
 * Company:      Brunel University<p>
 * @authors Stephen Gulliver
 * @version 1.0
 * Code contained within this application has been
extracted and adapted from:
 *
 * *** Accessing Individual Decoded Frames ***
 * *** Availble at:  http://java.sun.com  ***
 *
 * *** Building your own media player, Jeff Friesen
(2001):
 * *** Availble at: http://www.informit.com ***
 *
 * *** Lab 8 of MM Applications 2003 (Harry Aguis,
Brunel University)
 * *** Availble at: http://disc.brunel.ac.uk***
 *
 */
package roimplayer;
import java.util.*;
import java.io.*;
import java.awt.*;
import java.lang.Object.*;

class ROIFrame {
 int xCoord1 =0;
 int xCoord2 =0;
```

```
  int yCoord1 =0;
  int yCoord2=0;
  int framevalue =0;
  int inter = 0;
  int exter = 0;

public ROIFrame(int currentFrame, int x1, int x2, int y1,
int y2, int internal, int external)
// Assign internal values
{
  xCoord1 = x1;
  xCoord2 = x2;
  yCoord1 = y1;
  yCoord2 = y2;
  inter = internal;
  exter = external;
  framevalue = currentFrame;
}

/* The following code allows the programmer to retreive
the values of internalfps, externalfps, current frame and
coordinate values */

public int getInternalfps(){
  return inter;
}

public int getExternalfps(){
  return exter;
}

public int getCurrentFrame(){
  return framevalue;
}

public int getx1(){
  return xCoord1;
}

public int gety1(){
  return yCoord1;
}

public int getx2(){
  return xCoord2;
}

public int gety2(){
  return yCoord2;
}

}
```

PostAccessCodec.java

```
/**
 * Title:        Region of Interest Media Player<p>
 * Description:  <p>
 * Copyright:    Copyright (c) Stephen Gulliver<p>
 * Company:      MSc Dissertation for Multimedia
Information Systems<p>
 * @authors Stephen Gulliver / Jimin Patel
 * @version 1.0
 * Code contained within this application has been
extracted and adapted from:
 *
 * *** Accessing Individual Decoded Frames ***
 * *** Availble at:  http://java.sun.com  ***
 *
 * *** Building your own media player, Jeff Friesen
(2001):
 * *** Availble at: http://www.informit.com ***
 *
 * *** Lab 8 of MM Applications 2003 (Harry Aguis,
Brunel University)
 * *** Availble at: http://disc.brunel.ac.uk***
 *
 */

package roimplayer;
// Import Java libraries
import java.awt.*;
import javax.media.*;
import javax.media.control.*;
import javax.media.Format;
import javax.media.format.*;
import javax.swing.*;
import java.io.*;
import java.awt.image.*;
import java.awt.image.FilteredImageSource;
import java.awt.Dimension;
import java.awt.event.*;
import java.awt.Image.*;
import java.awt.image.renderable.*;
import java.io.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
import com.sun.image.codec.jpeg.*;
import javax.media.*;
import javax.media.format.*;
import javax.media.format.VideoFormat;
import javax.media.util.*;
import javax.media.util.BufferToImage;
import javax.media.control.*;
import javax.media.protocol.*;
import javax.media.control.FormatControl;

   public class PostAccessCodec extends
PreAccessCodec {

// program declares and define program variables
      Processor p;
   Object waitSync = new Object();
   boolean stateTransitionOK = true;
   private File file;
   public BufferToImage btoi = null;
   public ImageToBuffer itob = null;
   public Image frameImage = null;
   public BufferedImage tempFrame = null;
   public static BufferedImage tempFrame2 = null;
   private int counter = 0;
   private int counter2 = 100;
   private int tempCounter = 0;
           public PostAccessCodec() {
//sets supported video and image formats
         supportedIns = new Format [] {
            new RGBFormat()
         };
      }

// Callback to access individual video frames.
      void accessFrame(Buffer frame) {
// Output the frame #, time & data length.
       long t = (long)(frame.getTimeStamp()/10000000f);
      System.err.println("Post: frame #: " +
frame.getSequenceNumber() +
                  ", time: " + ((float)t)/100f +
                  ", len: " + frame.getLength());
/* check if it this the first frame in the buffer count and
if so load the temp frame */
         if (counter == 0){
         try{
            btoi = new BufferToImage((VideoFormat)
frame.getFormat());
            frameImage = btoi.createImage(frame);
            tempFrame = new
BufferedImage(frameImage.getWidth(null),
```

```
frameImage.getHeight(null),
BufferedImage.TYPE_INT_RGB);

        } catch (Exception e){
          System.out.println("Error !");
         }
        }
       int x = 20;
            if (frame == null)
               frame.isDiscard();
            else{
/*Convert the buffer into an image and then a buffered
image */
               btoi = new BufferToImage((VideoFormat)
frame.getFormat());
               Image img = btoi.createImage(frame);
               BufferedImage bi = new
BufferedImage(img.getWidth(null), img.getHeight(null),
BufferedImage.TYPE_INT_RGB);
               tempFrame2 = bi;
               Graphics2D g2 = bi.createGraphics();
               g2.drawImage(img, null, null);
               g2.drawString("test" + counter, 30, 30);
               ImageFilter filter = new CropImageFilter(x,
20, 60, 60);
               FilteredImageSource fis = new
FilteredImageSource(bi.getSource(), filter);
               Graphics2D g1 =
tempFrame.createGraphics();
               g1.drawImage(tempFrame, 0, 0, null);
              counter ++;
//increment counter
        }//end else

/* We use a five frame counter to facilitate reduction in
frame rate via the duplication of frames passed to the
tempFrame variable. Additional remainer values of
counter can be used to repeat duplicated frames*/

        if (counter%5 == 1){
          tempFrame = tempFrame2;
          System.out.println("temp Frame " +
tempCounter);
          tempCounter++;
        }
        Buffer finalImg = itob.createBuffer(tempFrame,
25f);

//finalImg defines the final image that is passed to frame
        Object data = finalImg.getData();
        frame.setData(data);


        // Copy the input attributes to the output
        frame.setFormat(finalImg.getFormat());
        frame.setLength(finalImg.getLength());
        frame.setOffset(finalImg.getOffset());
} // end of PostAccessCodec



 public void saveJpeg(BufferedImage bi, int counter){
/* saves all frames as single jpeg files – uses counter
value to differenciate between frames */
           try{

              FileOutputStream fos = null;
                 String file = "C:\\JPEGS/myJpeg" +
counter + ".jpg";
// defines output location
                 fos = new FileOutputStream(file);
                 JPEGImageEncoder encoder =
JPEGCodec.createJPEGEncoder(fos);
                 encoder.encode(bi);
                 fos.flush();
                 fos.close();
           }
           catch(FileNotFoundException e) {
                 System.out.println(e);
           }
           catch(IOException ioe) {
                 System.out.println(ioe);
           }
           catch(Exception e){
                 System.out.println(e);
           } // end catch

 }


    public String getName() {
       return "Post-Access Codec";
    }
  }
```