Jeremy Woods  
CSS 584  
Assignment 2

**How to run program**

This program has 2 executable parts. One being ProcessVideo.exe and the other being VideoControl.exe. I have created a batch file that has the ability to run both of these with one click. If you open the executable files zip in the assignment and click on launcher.bat both parts of the program will run and that is all that is need. If you do not wish to do it that way, first launch ProcessVideo.exe, then wait for the frame text file to be created and then run VideoControl.exe. It is done in this order so that ProcessVideo.exe can first create the files that VideoControl.exe will need.

**Step-by-step use of System**

When the system first comes up there will be a panel to the upper left with an image of the beginning of the first shot, on the upper right there will be 3 rows. In the first row are the simple instructions for using the system (instructions). In the second row the current shot’s start frame and end frame numbers (video info). In the last row is a play button (play button). The bottom panel holds all of the images that are the first image of a different shot and it is in a scrollbar. Here you can either select a different shot from the bottom panel, or you can play the first shot in one of two ways. First you can simply use the play button, or you can click on the image in that panel, and that will also begin the shot.

After a shot is selected, it will appear in the upper left panel and the video info panel will change to contain the start and end frame for that shot. You can select to play this shot or select another image from the bottom. If you choose the play the shot, you can press the play (or click on the image) to pause the shot or choose a different shot while the current on is playing. Once a shot ends, its last frame is shown. If you want to play again, you can press the play button (or click on the image) and it will start the shot over from the beginning.

**File Functionalities**

To create this program, I used the following program files: VideoControl.java, VideoImage.java, ProcessFrames.java, ProcessVideo.java, and VideoFrame.java.

VideoControl.java is a file based of the Xuggler demo file DecodeAndPlayAudioAndVideo.java. It is the file in charge of playing/pausing the video and interacting with the GUI to change frames and receive instructions from the user. It also uses ProcessFrames.java to get the value of the first frames for the shots.

VideoImage.java is the GUI. It uses a JPanel to show the video as the underlying VideoControl.java class controls the playing of audio and tries to synchronize them both. It manages the information given to it by the VideoControl.java class and pass user input to the VideoControl.java class as well.

ProcessFrames.java reads a .txt file form the disk and use that file to create a matrix. With that matrix, it calculates the distances between all of the frames and using those distances it uses the twin-comparisons algorithm to find the first frames of all of the shots in from the matrix that it analyzed.

ProcessVideo.java takes a video file and uses the Xuggler API to read each frames of the video from number 1000 to number 4999 and create an intensity matrix. Once it is done reading from the video, it writes that matrix to the disk.

VideoFrame.java is used to represent a frame is a class context. It has 3 parts, a buffered image, a frame number, and a time stamp. These objects are what the VideoControl.java and VideoImage.java classes use to communicate and simple the consolidation of information.

**Library**

I used a library called Xuggler for the entire project. Xuggler is an open source video manipulation and playback API for java. It is no longer supported and a lot of the methods are deprecated, but it is openly available on github and it contained the methods and things I needed in order to correctly complete the project. My main reason for sticking with Xuggler was that it read the video frame by frame and then just process each image at the proper framerate in sequence to produce a video. Since it was already using frames, I had no need to try to find a way to grab the frames, all I had to do was set the range that it read frames from and I would have all of the frames I needed to complete the project.

**Tools**

As for tools I coded in eclipse and in order to use Xuggler without having to rebuild the code with Linux using a cross-platform for windows I have to use Maven. Maven just basically finds dependencies from a URI in a pom.xml file and brings them into your project so you do not have to worry about having the correct ones. I am not sure how the executable for the program deals with that and if it stores the dependencies from the maven or if it needs internet access to run. Other APIs I used the same BufferedImage and color libraries as in the first two assignments to build the feature matrix for the frames. I used VirtualDub to check for matching frames between my program and what was the expected frame. The video file I used for Xuggler was 20020924\_juve\_dk\_02a.avi while I had to use the mpg for VirtualDub so I assumed some of it would be different.

**Techniques**

My approach to this project was all based around the ability to get a single frame. Once I could find a way to get a frame I used that throughout the entire project to get it done. To process the video for building the matrix off line, I read through the proper 4000 frames and just cut out the delays for playing at the proper rate and the audio, and instead of trying to place the images on the screen I process it and place it in a matrix to be written to a file.

For playing and pausing, because Xuggler just uses a while loop that reads from specified frames I used thread synchronization on a static thread in my VideoControl Class. If a user presses pause, the VideoControl class waits until play is pressed at which point VideoImage sends the notify message to the VideoControl.

To play only a specified shot I seek through the video. The audio method did not like the seeking and because the audio is buffered before it is played, seeking to certain frames would cause errors with the original checking because the number of bytes decoded for the audio might be incorrect. To fix this I had to remove their error checking at that point, because it was not that the information wasn’t their it just could not be read the way it expected, so once the error check was removed I stopped throwing the exception and the audio worked fine.

Maintaining all of the proper information for a specific video in a VideoFrame object was smart. I used that to retrieve information for my GUI.

I used JList and wrapped it in a JScrollPanel to make the scrolls bar and try to use the space in the GUI expected sizes the best. To do this I had to write a custom renderer for the VideoFrame object that placed the information from the object into a nice 200x200 pixel JPanel. The beginning size and maximized are the ideal sizes for the user to use this application, otherwise there can be a large space at the end of the line if the icons in the bottom frame are off just enough to barely miss the next picture.

**GUI Pictures**

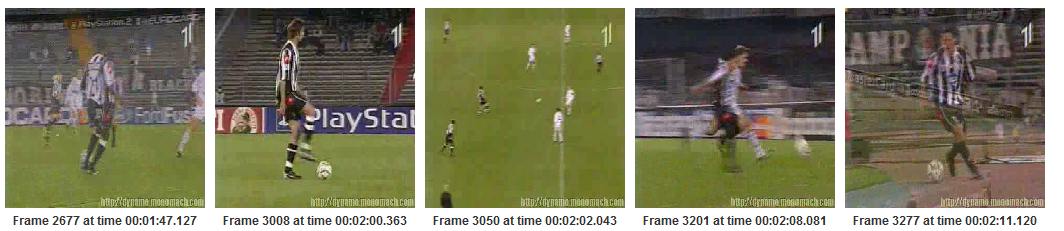
Note: Viewing the shot works, but cannot demo here. You must use application to see.

First Frame #s

*Figure 1*: Frame Pages 1 - 6

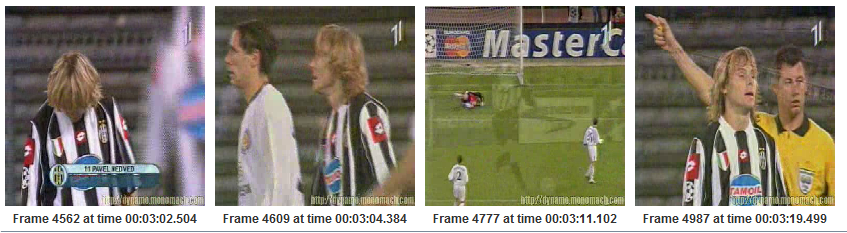












*Figure 2*: Entire Program

