CMPT 365
Final Project

Collaborators:

Sung Jun (Brian) Pak	301168028	bpak@sfu.ca
Chazz Young	301209440	chazzy@sfu.ca

Design

The project was written in Java. It utilizes the ImageJ library (http://imagej.nih.gov/ij/), as well as the AVI_Reader class from the same site. The ImageJ library was chosen due to it's simplicity and power to easily read uncompressed .avi files, as well as separate it into frames. The AVI_Reader and ImageProcessor classes are most notably used in the project's main class, AVIFrames. On top of this, a simple GUI was created to easily run the program.

Instructions

1) Extract to your preferred directory
2) Open CMPT365P1.jar
3) Click on the "Select AVI File" button to select the file to test.
4) If the file is read successfully, a small dialog box will appear. Ensure that the "Use Virtual Stack" and press OK.
5) Hit either "Horizontal STI", "Vertical STI", "Horizontal Histogram STI", or "Vertical Histogram STI" to see the STI.

Known Bugs/Issues

1. An input video needs to be an uncompressed AVI file due to the library we selected to use. (AVI_Reader; details in the description below)

Description of Classes and methods

1. P1TransitionFinder.java - Simple main class to start the applet.

2. Applet.java - This is the primary GUI class, mostly generated code from Netbeans's GUI builder. It's responsibility is to connect the AVIFrames.java while also providing the GUI. This class is responsible for calling the appropriate methods for each STI, selecting the threshhold value for the histogram difference STIs, and displaying the output STI of an input video. File selection is done here.

3. AVI_Reader – Taken from http://rsb.info.nih.gov/ij/plugins/download/AVI_Reader.java. This class is used to read in uncompressed AVI files. Unfortunately, it is very limited, and AVIs had to be uncompressed using VirtualDub: http://www.virtualdub.org/ and http://forum.videohelp.com/threads/289379-VirtualDub-being-stupid-(VFW-codec-)

4. AVIFrames.java - Main class nearly entirely written by us with reference to the ImageJ API documentation. It contains an AVI_Reader object, that is responsible for generating the resulting ImageStack, literally a stack of images generated by AVI_Reader. It abstracts away the major details of dealing with AVI_Reader and key functions (further explained below) return an int[][] array (STI with pixels) to print the image

4.1 getVericalSTI() - This method returns the vertical STI (an int[][] array) of the input video. The 2d array has a size of frameCount x columnSize. There is an outer loop that iterates through the frameCount of the video, and we get the current image of the frame at i. Then, there is an inner loop that iterates through the columns of the frame, and we get the pixel of the frame at (rowSize / 2, column at j). Hence, an image of the center column from each frame is generated and returned.

4.2 getHorizontalSTI() - This method returns the horizontal STI (an int[][] array) of the input video. The 2d array has a size of frameCount x rowSize. There is an outer loop that iterates through the frameCount of the video, and we get the current image of the frame at i. Then, there is an inner loop that iterates through the rows of the frame, and we get the pixel of the frame at (columnSize / 2, row at j). Hence, an image of the center row from each frame is generated and returned.

4.3 getVericalHistDifferences(boolean thresh, float thr) - This method is the key function that combines all of the necessary methods to calculate the vertical STI using histograms (explained below). First, we create a 4d array, “hists[rowSize][frameCount][r][g]” that contains the histogram per frame per column. Next, each histogram is normalized iteratively. Then, there is an outer loop that iterates through the columns (index i) and an inner loop that iterates through the frames (index j).We get the two histograms, hists1[][] = hists[i][j] and hists2[][] = hists2[i][j + 1] and call getHistogramIntersection(hist1, hist2) to grab the “I” value, a measure of difference between them. We save the “I” values into another 2d array[columnSize][frameNum] to print out the STI. Finally, we convert the “I” values to a grey color image, or black/white image if a threshold is enabled (the default threshold:80 percent produces the best result).

4.4 getHorizontalHistDifferences(boolean thresh, float thr) - This method is the key function that combines all of the necessary methods to calculate the horizontal STI using histograms (explained below). First, we create a 4d array, “hists[colSize][frameCount][r][g]” that contains the histogram per frame per row. Next, each histogram is normalized iteratively. Then, there is an outer loop that iterates through the rows (index i) and an inner loop that iterates through the frames (index j).We get the two histograms, hists1[][] = hists[i][j] and hists2[][] = hists2[i][j + 1] and call getHistogramIntersection(hist1, hist2) to grab the “I” value, a measure of difference between them. We save the “I” values into another 2d array[rowSize][frameNum] to print out the STI. Finally, we convert the “I” values to a grey color image, or black/white image if a threshold is enabled (the default threshold:80 percent produces the best result).

4.5 normalize(int[][][] intarray) - This method is used to normalize the output array from getVerticalSTIHistogram(int col). It converts an int[][][] array to a float[][][] array by dividing every element of the int array (each counter of a histogram for every frame) by the total value of all counters of a histogram.

4.6 getVerticalSTIHistograms(int col) - This method returns the histograms for vertical STI of every frame (a int[][][] array). The 3d array has a size of frameCount() x N x N, where N = log base2(rowSize) + 1 as provided in the project specification. Next, we get the arrays of chromacity values of r and g using the chromacity(String color) method described below. There is an outer loop that iterates through the frameCount of the video, which is used to create a histogram (int[N][N] array) for each frame. Then, there is an inner loop that iterates through the columns of the frame, and we get the chromacity r and g of the frame at (row at i, column at j). Since r and g are float values in the range between 0 and 1, the float values are converted to integer values, rh and gh in the range between 0 and N. The histogram is incremented at [rh][gh]. Hence, an array of vertical STI histogram (for each frame) is generated and returned.

4.7 getHorizontalSTIHistograms(int row) - This method returns the histograms for horizontal STI of every frame (a int[][][] array). The 3d array has a size of frameCount() x N x N, where N = log base2(columnSize) + 1 as provided in the project specification. Next, we get the arrays of chromacity values of r and g using the chromacity(String color) method described below. There is an outer loop that iterates through the frameCount of the video, which is used to create a histogram (int[N][N] array) for each frame. Then, there is an inner loop that iterates through the columns of the frame, and we get the chromacity r and g of the frame at (row at i, column at j). Since r and g are float values in the range between 0 and 1, the float values are converted to integer values, rh and gh in the range between 0 and N. The histogram is incremented at [rh][gh]. Hence, an array of horizontal STI histogram (for each frame) is generated and returned.

4.8 getHistogramIntersection(float[][] hist1, float[][] hist2) - This method is designed to calculate the histogram intersection between two histograms. There is an outer loop that iterates through rows of a histogram (index i) and an inner loop that iterates through the columns of a histogram index(j), that gets the minimum value between hist1[i][j] and hist2[i][j] and sum it to get the "I" value. The final value of "I' will be returned. As each value has been normalized to from 0 to 1/total, the "I" value is in the range 0 to 1

4.9 chromacity(String color) - This method returns the chromacity scale of every pixel of every frame (a float[][][] array). The 3d array has a size of frameCount x rowSize() x columnSize.There is an outer loop that iterates through the frameCount of the video which gets the image at the frame k. Then, there is the 1st inner loop that iterates through the rows of the frame and 2nd inner loop that iterates through the columns of the frame. We can get the pixel value at the row at i and the column at j. If color == "r" then we insert r = R / (R + G + B) to the array. Else if color == "g" then we insert g = G / (R + G + B) to the array. There is a condition to check if a pixel has a black color to avoid the integer-divide-by-0 error. Hence, an array of either chromacity r or g is generated and returned.

Reflections on the project

1. We believe that the project had an adequate level of difficulty.
2. We have used a lot of multi-dimensional arrays inside of our functions to carry around the pixels for each row, column, frame, ...etc, we could have created our own object, Pixel and have row and column attributes, so that it's more clear to see what's going on inside the arrays.

3. We can expand our code by importing other libraries for reading different kinds of video files, so that we can process videos that do not have .avi format.
4. Since we have dealt with Cuts and Wipes, for the next project, we could implement a program detecting a Dissolve for example.
5. Our project could also be expanded to include functionality such as choosing the scale of the image, dynamically allocated space for an image with given dimensions, as well as functionality for other functions for the STI differences (such as Euclidian distance, or IBM's formula for color difference)