Motion Detection

Xi (Donny) Wu 301300922

Xiaoting (Ridge) Zhang 301315460

Video: https://www.youtube.com/watch?v=p4DECf02x9g&feature=youtu.be

Overview

Originally, we will program motion detection using the difference method, and implement difference matting in the GUI interface, which should assuage the issue of changes in the background. By doing so, we can see if our method, being a lot easier than Nikolov and Kostov's machine learning method, is just as effective or almost as effective. The result will be should be a motion tracked object or person that has its background cleanly removed by the color difference method and placed onto a new background. In addition, we will aim to cleanly remove all background motion with the keying options provided in the GUI.

After implementing the difference matting (see Work Description section), we have decided to replace it with additional functions including sliders which changes gamma of the output image, changes the proxy image by reducing the number of number of pixels needed to be processed, frame stacking where separate frames are stacked together, and delta which controls how many frames of difference. We have also implemented multi-threading to improve the performance of our code. This report will walk through the steps that we have done, including implementations that we have decided to remove once implemented.

Research

One previous work was motion Detection Based on Frame Difference Method published in the International Journal of Information & Computational Technology, which is one paper we will review in depth [1]. This work uses the difference method and is able to detect motion in four steps. First, a background frame without the moving object and sequence frames containing the moving object needs to be captured [1]. Then the absolute difference is calculated between the consecutive frames. Third, the difference image is converted into a grayscale image and then into binary image [1]. Last the morphological filter is used to remove noise [1].

In another previous work on Motion Detection Using Adaptive Temporal Averaging Method, done in Technical University of Varna, Nikolov and Kostov are building on top of the most comply used method, being the temporal average algorithm, for motion detection [2]. This is the second research paper we will review. The Temporal average algorithm is one method for background subtraction [2]. This paper was successful in solving the background change issues which was an issue for the previous paper with machine learning [2]. For example, if the background changes, the difference image will be inaccurate. For our project, we will try to address that issue in another way.

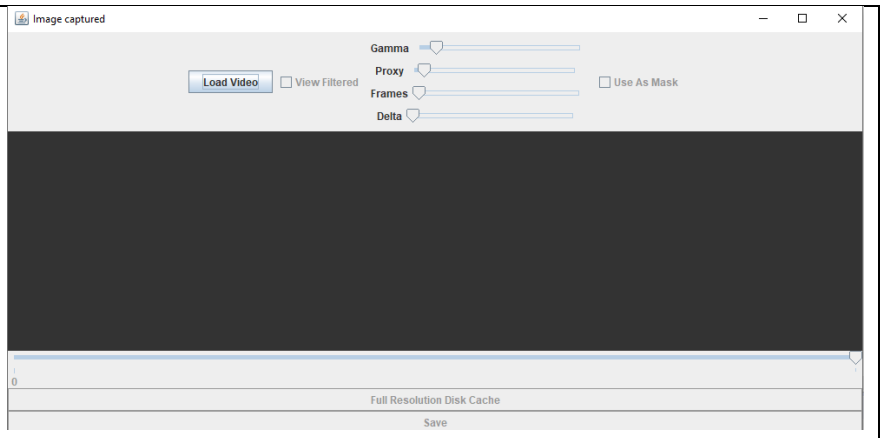
Work Description

Note, this includes work that was removed from the final project and the code is what we had during that point of the project, which may not be the same as the final submission.

Task	Person	Details
The first step was creating a Java file which loads a video, separates the frames, and export the frames as JPEG files.	Ridge	

		`` VideoCapture camera = new VideoCapture(newPath); if (!camera.isOpened()) { System.out.println("Error! Camera can't be opened!"); return; } Mat frame = new Mat(); int video_length = (int) camera.get(Videoio.CAP_PROP_FRAME_COUNT); int frame_number = (int) camera.get(Videoio.CAP_PROP_POS_FRAMES); //save frame as jpg while(camera.read(frame)) { Imgcodecs.imwrite(basePath +"\\output\\" + frame_number +".jpg", frame); frame_number++; if (frame_number == video_length-1) { break; } } ``` First, we have wrote the code to extract the frames from a video and saved them as images in the folder called output, with the help for the OpenCV library. This was achieved by using the VideoCapture class to instaniate a camera object and open it. Then we have instantiated a frame object from the class Mat. Then, as long as there are frames in the video, save the frame as a jpg image in the folder output, and if all the frames have been saved, break the loop. Our current implementation uses png instead of jpg so it will be saved as a lossless format.
The second task was loading the frames as buffered images so they can be manipulated with the subtraction method for motion detection.	Ridge	``` //read the previously extracted frames that were saved and load it in the array BufferedImage[] bImg = new BufferedImage[video_length]; for (int i=0; i<video_length-1;i++) { try { bImg[i] = ImageIO.read(new File(basePath +"\\output\\"+i+".jpg")); } catch (Exception e) { System.out.println("Cannot load the provided image"); } } ``` Then, the program will load the images into a BufferedImage array which will allow us to manipulate the images to achieve our desired outcome. Considering that we have named our exported jpg images with numbers incrementing from 0, we can use the for method to load in all the images to the BufferedImage array and we have implemented the try and catch functions to prevent crashes.
The third task was to write the subtraction method which will allow motion detection.	Ridge	``` public BufferedImage subtract(BufferedImage src1, BufferedImage src2) { int width = src1.getWidth(); int height = src1.getHeight(); BufferedImage out = new BufferedImage(width, height, src1.getType()); for (int y = 0; y < height; y++) { for (int x = 0; x < width; x++) { Color pixel1 = new Color(src1.getRGB(x, y)); Color pixel2 = new Color(src2.getRGB(x, y)); int r = pixel1.getRed() - pixel2.getRed(); int g = pixel1.getGreen() - pixel2.getGreen(); int b = pixel1.getBlue() - pixel2.getBlue(); Color pixelOut = new Color(clamp(r), clamp(g), clamp(b)); out.setRGB(x, y, pixelOut.getRGB()); } } return out; } ``` Third, we have written a subtract method which subtracts the next frame with the current frame. This method can be called after the image is loaded into the

		array, which is the second task. For each pixel, subtract the pixel value from image 1 by image 2 for each channel.
During the fourth task, we have discovered that exporting the images is not necessary.	Donny	`// read in all frames` `System.out.println("Starting to read individual frames ----------");` `Mat frame = new Mat();` `while (videoCapture.read(frame)) {` `  sourceFrames.add(matToBuf(frame));` `  currentFrame++;` `  System.out.println("Reading frame " + currentFrame + " of " + frameCount);` `}` `videoCapture.release();` `System.out.println("Done!");` The frames can be saved into a BufferedImage array and being stored with Ram, rather than Rom. This approach is better than the previous one as it eliminates hard drive writing time; however, sufficient ram is required. For longer videos the previous method is much more applicable. For this method, read the frames and add it to the BufferedImage ArrayList
The fifth task was to write code to export the video.	Donny	`// save video` `System.out.println("Saving video ----------");` `VideoWriter videoWriter = new VideoWriter("1output.avi", VideoWriter.fourcc('x', '2', '6', '4'), fps,` `  new Size(width, height));` `Iterator<BufferedImage> iterator = outputFrames.iterator();` `while (iterator.hasNext()) {` `  Mat mat = bufToMat(iterator.next());` `  videoWriter.write(mat);` `}` `videoWriter.release();` `System.out.println("Done!");` This is done with the VideoWriter using the OpenCV Library. It requires us to set an appropriate frame rate and tell it which frames to use to convert it into a video and where it should be saved with which filename.
The sixth step was to tell the user the process of producing the motion detection video	Donny	`Processing frame 98 of 98` `Done!` `Saving video ----------` `Done!` `OpenH264 Video Codec provided by Cisco Systems, Inc.` Using the Console, we will tell the user which frame is currently being extracted, or which frame is currently being processed, or if the video has been successfully exported. We will also notify the user if the video cannot be exported.
The seventh step was to start on a functionable UI. (improved)	Ridge	 This GUI has a FileChooser which allows the user to select the video that the want to motion detect on. The user can also select a colour they wish to suppress to black with the colour difference method. Once the motion detect button is pressed, the GUI will be disposed and the execution will begin.
The eighth step was to write the method for the colour difference method. (removed)	Ridge	`public void difference(int r, int g, int b) {` `  if (removeRed == true) {` `  if (r > g) {` `  r = g;` `  }` `  }` `  if (removeGreen == true) {` This is a place holder for the difference method where if the remove colour Boolean is true, which is triggered by the UI check boxes, the operation will be performed.

The ninth step is to add four additional functions being gamma, proxy, frame stack, and delta to the UI and removing the implementation of the difference method.	Donny	 This is what the UI looks like when the program runs. All the buttons are disabled except for the load video button which opens a file chooser. We have implemented four additional including sliders which changes gamma of the output image, changes the proxy image by reducing the number of number of pixels needed to be processed, frame stacking where separate frames are stacked together, and delta which controls how many frames of difference. The view filtered button allows the video to show the changes. The "Use as Mask" button multiplies the mask with the original frame so the user can actually see the original video in the resulting video. For example, if a person is moving, they can actually see a person is moving rather than a white object.
The tenth step was implementing the methods for the functions being gamma, proxy, frame stack, and delta.	Donny	``` // this is the gamma slider, the first slider on the UI public JSlider initGammaSlider() { JSlider newSlider = new JSlider(0, 1000); //set initial value to 100 so when multiplied by 0.1 it becomes 1 newSlider.setValue(100); gamma = 1; newSlider.addChangeListener(new ChangeListener() { @Override //change the gamma value public void stateChanged(ChangeEvent e) { gamma = newSlider.getValue() * .01; if (showFiltered) { showFrame(slider.getValue()); } } }); return newSlider; } ``` The gamma value chosen by the slider allows the user to change the brightness of the manipulated video.

```
//slider for proxy
public JSlider initSampleSlider() {
    JSlider newSlider = new JSlider(1, 50);

    newSlider.setValue(4);
    sample = 4;

    newSlider.addChangeListener(new ChangeListener() {
        @Override
        public void stateChanged(ChangeEvent e) {
            sample = newSlider.getValue();
            if (showFiltered) {
                showFrame(slider.getValue());
            }
        }
    });
    return newSlider;
}
```

The proxy value chosen by the slider allows the user to change quality of the video allows the user to view changes made to the proxy image a lot faster and it also increases the speed of the processing. It basically merges pixels together.

```
//slider for frames
//stacks the frames
public JSlider initMultiFrameSlider() {
    JSlider newSlider = new JSlider(2, 20);

    newSlider.setValue(2);
    stackFrames = 2;

    //listener to detect changes in slider
    newSlider.addChangeListener(new ChangeListener() {
        @Override
        public void stateChanged(ChangeEvent e) {
            //get value of the slider
            stackFrames = newSlider.getValue();
            diffFrameSlider.setMaximum(stackFrames - 1);
            //if greater set it as new value
            if (diffFrameSlider.getValue() > diffFrameSlider.getMaximum()) {
                diffFrameSlider.setValue(diffFrameSlider.getMaximum());
            }
            if (showFiltered) {
                int usableFrames = numberOfUsableFrames();
                slider.setMaximum(usableFrames);
                //if value of slider is greater than usable frames, then set usable frames as max
                if (slider.getValue() > usableFrames) {
                    slider.setValue(usableFrames);
                }
                showFrame(slider.getValue());
            }
        }
    });
    return newSlider;
}
```

The frame value chosen by the slider allows the user stack frames together to make the overall video shorter.

```
//slider for delta
public JSlider initDiffFrameSlider() {
    JSlider newSlider = new JSlider(1, stackFrames - 1);

    newSlider.setValue(1);
    diffFrames = 1;

    //action listener for
    newSlider.addChangeListener(new ChangeListener() {
        @Override
        public void stateChanged(ChangeEvent e) {
            diffFrames = newSlider.getValue();
            if (showFiltered) {
                showFrame(slider.getValue());
            }
        }
    });
    return newSlider;
}
```

The delta value chosen by the slider allows the user manipulate how they want the frames to stack and the difference amount. This works together with the frame slider.

To improve the performance, we have implemented multi-threading so a thread can be used for different tasks.	Donny	`public void loadVideo(String filePath) {` `loadVideoButton.setEnabled(false);` `loadVideoButton.setText("Loading...");` `//run on new thread` `Thread thread = new Thread() {` `@Override` `public void run() {` `System.loadLibrary(Core.NATIVE_LIBRARY_NAME);` For example, when loading the video, we have dedicated a thread for that video.

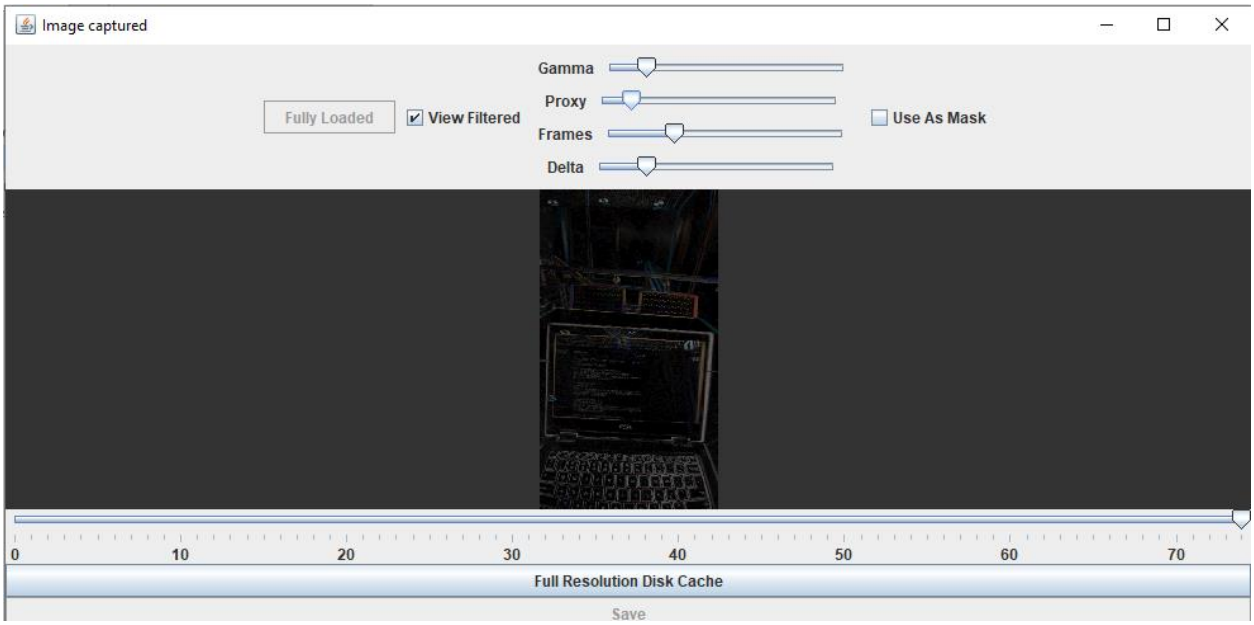
Final Result

Replaced Colour Difference Method with Four Functions

As previously mentioned, we have implemented four additional including sliders which changes gamma of the output image, changes the proxy image by reducing the number of number of pixels needed to be processed, frame stacking where separate frames are stacked together, and delta which controls how many frames of difference. The view filtered button allows the video to show the changes. Use as mask button multiplies the mask with the original frame resulting in a video with less noise.

Built an Interactive UI

Our UI is very interactive as it allows the user to choose a file of their choice, use the sliders to make manipulations, and see the result frame by frame as each frame is being displayed as a proxy image inside the UI. This will allow the user to make adjustments while looking at their changes and export only when they are happy with what the file will look like. Considering that exporting takes some time, the proxy image viewer will allow them to only export once rather than making test exports.

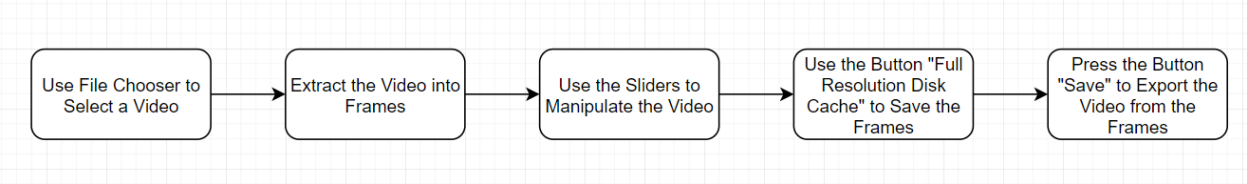


Multi-Threading to Improve Performance

Multi-threading is considered a bonus functionality for our project because it does not affect the front end of the software, but the back end. By allocating a specific thread for a specific task, the overall

performance of our program will improve resulting in faster a computing speed. If this is not implemented, only one thread will be used and multiple tasks cannot be executing simultaneously.

Complete Process from Importing the Original Video to Exporting



Overall Challenges

The first challenge encountered was linking the libraries in a way that it is linked not only on my computer, but also on other computers. Our project uses a .dll extension for the library, which only works for PC Windows computers, not Macs.

One challenge was learning how to extract the frames from a video and save them as images. It took a white to research existing libraries, learning the OpenCV library, and understanding how to implement this library. This library requires specific format such as instantiating the class for VideoCapture and Mat in order for the library to read the data.

Another challenge was learning how to compile the images which were subtracted in the BufferedImage array into a video and exporting that video. Similar to the previous challenge, it took a white to research existing libraries, learning the JavaCV library, and understanding how to implement this library. This library requests the instantiation of the class FFmpegFrameRecorder which can select the type of file export, bitrate and so on for the exported video.

By using the second method, one that stores the frames in the BufferedImage ArrayList rather than exported as files, an issue could be not enough random-access memory (RAM). I was filming the sample video at 4k resolution; hence, if the video is short and is captured at 720p, this should not be an issue.

```
Processing frame 64 of 80
Processing frame 65 of 80
Processing frame 66 of 80
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
        at java.awt.image.DataBufferByte.<init>(Unknown Source)
        at java.awt.image.Raster.createInterleavedRaster(Unknown Source)
        at java.awt.image.BufferedImage.<init>(Unknown Source)
        at video.Video.subtract(Video.java:112)
        at video.Video.<init>(Video.java:88)
        at video.Video.main(Video.java:32)
```

In the end, we decided to not use ram and save the frames in a folder instead. This way, running out of ram will not be an issue, especially for larger videos.

Distribution of tasks

Aside previously mentioned, both Donny and Ridge has taken part in programming this project and we have alternated frequently through the different steps to ensure equal contribution. Moreover,

by alternating, we have the opportunity to check each other's work and test for errors that may not present on one's computer but on the other.

Aside from repeating the distribution of tasks that was clearly labelled in the Work Description section of this report; on a wholistic sense, Ridge is more responsible to finding pieces of code that will create the functionalities of this project such as implementing a file choose, methods for reading the frame or methods for achieving the desired outcome where Donny is responsible for linking the elements together into a single cohesive program, debugging errors and improving where necessary.

Two large, yet unmentioned tasks were distributed: being the written report and the video. The report was written by Ridge and the video was created by Donny.

Conclusion

Donny

Before starting this class, I was interested in computer graphics but did not get into it too much. After this class. however, I've now known the underlaying ways images, video, and audio can be computerized and processed. Reading Google's HDR+ paper has made me realize that to make a good image there are a lot of things behind the scene that a camera is doing. During the creation of the final project, I have learnt how to read in video files as individual frames and how ram intensive video processing can be. I've also learnt how to use multithreading within Java to allow the UI thread to remain fluid while processes are going on in the background, and speeding up a task by splitting it up into several threads. In the future, I will be looking more into computer graphics in my spare time and learning more about the advances made within.

Ridge

During the first week of class where my fellow classmates and I were asked on what we wanted to obtain or learn from this class, my response was to improve my understanding on what computing can do and by completing this project, I have achieved that goal. Prior to this course, I knew how motion can be tracked but I had no idea what process was needed. Also, I knew that images on the screen were comprised of red, green and blue dots, but nothing more. After taking this class, I learned what channels are and how channels together make what we see on the screen. By doing this project, I learned that motion tracking is fairly easy to implement. Initially, prior to reading the research paper, I was kind of demotivated as I thought the implementation will be a lot more difficult, but once I read the paper, I realized that a simple subtraction method can be used to detect motion. After completing this project, I realized that there are endless applications that I can use motion detection from. Moreover, I have learned that computing is a very powerful tool to understand as the functions and capabilities of it is endless.

Resources

Java Awt Library https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html

Javax Swing Library https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

Bytedeco FFMpeg library https://github.com/bytedeco

Bytedeco JavaCV library https://github.com/bytedeco/javacv

OpenCV library https://opencv.org/

Royalty Free clips from archive.org https://archive.org/details/stock_footage?&sort=-downloads&page=4

Royalty Free clips from https://www.youtube.com/watch?v=dbAselSIQUQ