Brian Nguyen
664.02
12/13/2022

Foreground-Background Separation In Video

1. **Introduction**

The general problem is to completely understand the "real-time foreground-background using codebook model" PDF and to implement it into a usable running script that will take a video and separate the foreground and background. The reason why understanding the PDF completely is important is because there are a lot of complicated terms and concepts and without understanding them, it would be hard to translate them into code. This would greatly impact the solution due to the algorithm being tricky to understand. I greatly underestimated how much I needed to know about the PDF when I first tackled the problem, which was reflected in the first demo since it was not what was expected for the project. To resolve this- I will explain the proposed thought process and then go on to how I went about solving the problem. Since the codebook's algorithm is the only thing I need to use for this project, I did not read any other outside source articles or books to help me with this project. I will show step by step how I approached the problem and the challenges that occurred along the way, and how I was able to overcome those challenges.

2. **Proposed problem formulation**

What I am trying to do is to implement the codebook's algorithm for any type of video and be able to isolate the foreground and background. Whenever you are looking at a video, there will be a foreground and background. This project- proposed to take the video and break it down into frames then combine it back into a video after the codebook's algorithm has been implemented by isolating the foreground and background. The foreground will be white while the background is black.

3. **High-Level description of your idea**

By using the codebook's algorithm- rather than going frame by frame, it is better to go through every frame all at once. Within my program, it will loop through every frame and take in the input from the frames that were broken down by Matlab and check through the lambdaMax and check the code word size. After it is processed it will produce an output that has run through the codebook's algorithm. The reason why this approach is the preferable method is due to the codebook's algorithm is designed to go through multiple frames to keep the previous frame data. Then by following the codebook's algorithm, and improving on it, I had to play around with the parameters to see how it affects the output.

4. **Prior work in the area**

As I have mentioned previously, I did not use any outside research papers or articles to help me with this project. I did however try to look on youtube for different videos explaining the concept of the codebook in further detail, so I could get a better understanding. I did not implement any of their methods, but it did help me understand how the algorithm is supposed to work. I followed the PDF's algorithm and only made modifications to the parameters.

5. **Description of your work**
   - **Overview of system description**

Call buildCodeWord

Loop through 307 BMP frame files

Read the first 72 bytes of BMP structure data

Loop through BMP file

Read pixel data(RGB 3 bytes) in the BMP file

Call **processPixel** routine to initialize the codeword array list (for the first time) or to check if the pixel is in the list or not

If not, add in the list

If exists, update the list with processPixel routine for brightness(min, max), frequency, Run-length, first and last time access

Warp around run-length for each code word (section III in pdf material)

Calculate maximum negative run-length (**lamdaMax**)

Call buildBackFrontGndOutput implement Background subtraction

Loop through 307 BMP frame files

Read the first 72 bytes of BMP structure data

Save data in aux_data array list for BMP output file

Loop through BMP file

Read pixel data(RGB 3 bytes) in the BMP file

Save pixel in the pixelData array list

Write structure data(aux_data array list) to the BMP output file

Loop through pixelData list to implement Background subtraction algorithm

Call **CodeWordForbackGnd** to determine a match in code word array list

If match => background pixel (0,0,0) to the output BMP file

If not match => frontground pixel (255,255,255) to the output BMP file

- **Specific technical details for each part of the system on which work was done**

- **Relate how technical work supports your idea**

Section: MatLab Implementation (Brian Nguyen, Anthony Nguyen)

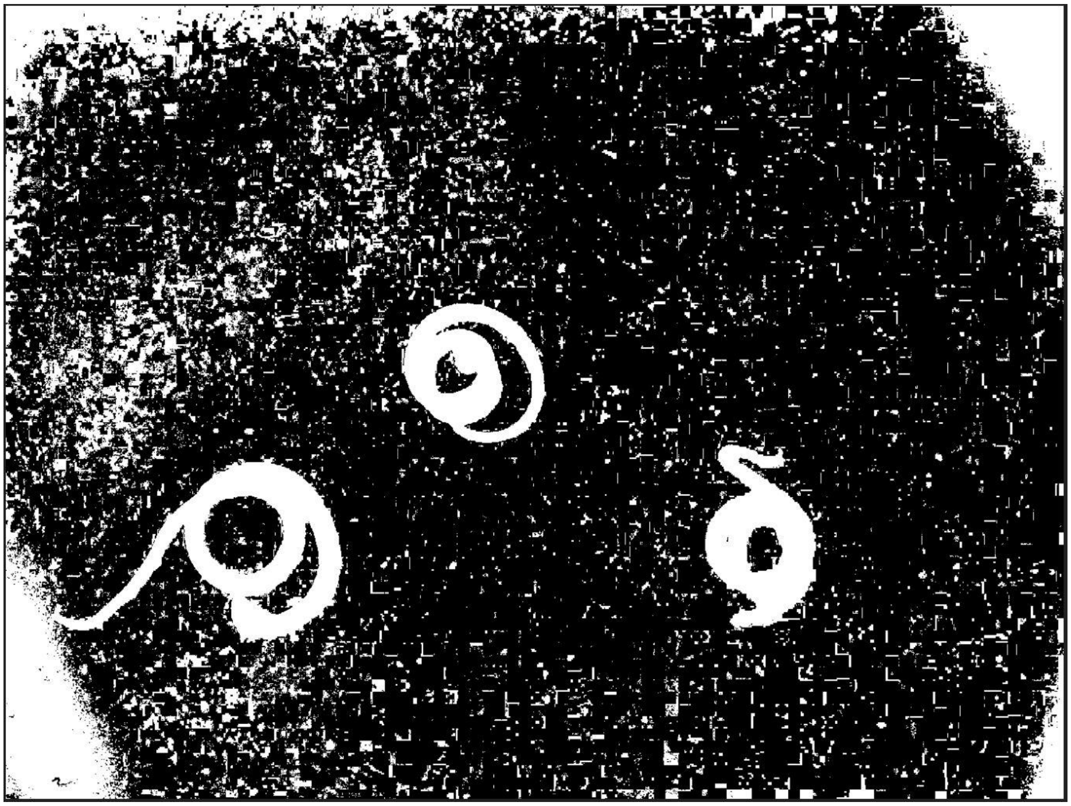
By using Matlab's functionality, there was a way to track how the foreground's movements.

Section: Implement Codebook In Java (frame by frame) (Brian Nguyen)

Matlab to break down the 20 sec compressed video in MOV (H264) format frame by frame with the uncompressed raw RGB data frame in BMP format (BMP, 1036x1384 pixels for each frame, each pixel has 3 bytes for Red, Green, and Blue color values, a total of 1433824 pixels for each frame, 440183968 pixels for the entire video, a total of 1320551904 bytes if the movie data needs to store in memory). The 307 BMP frame files use the codebook's algorithm to isolate the background.

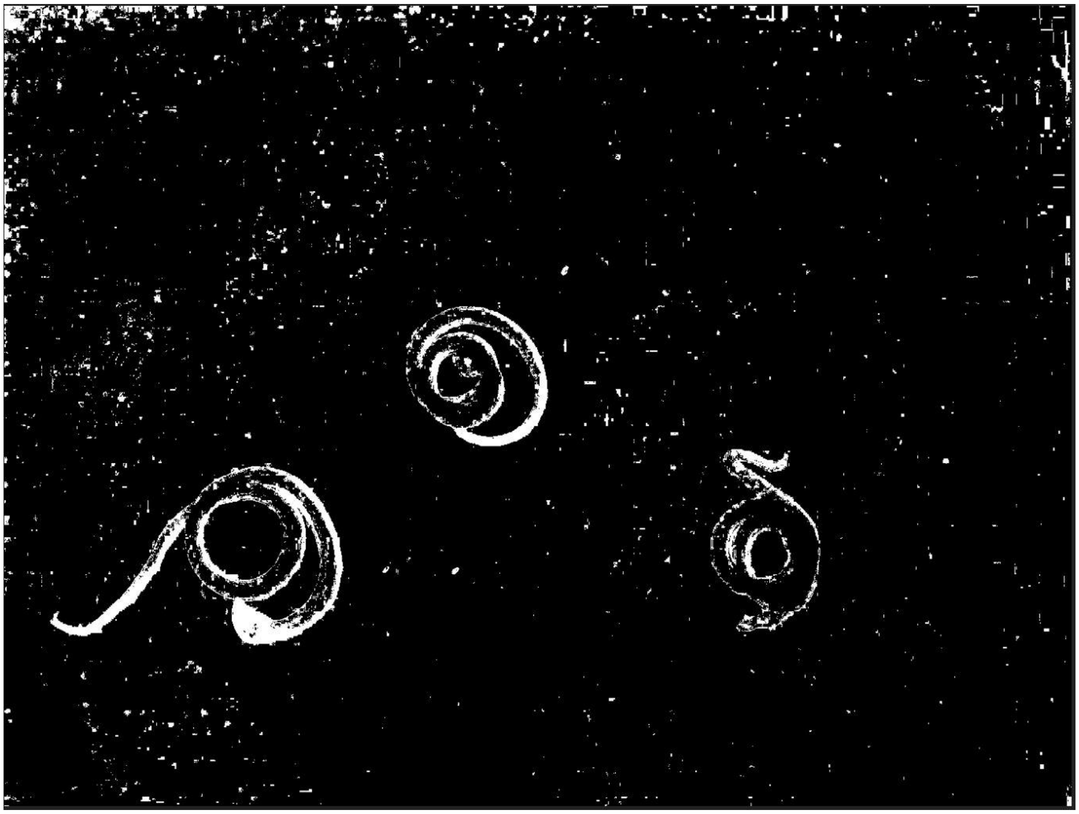
Section: Implement Codebook In Java All Frames (Brian Nguyen)

307 Frames delta(pixel comparison accuracy)=1, N(total number of raw pixels for the movie)/174



I was able to get the foreground isolated, but I could not figure out how to deal with the background with the intensity spiking.

307 Frames delta=1, N(total number of raw pixels)/60, frequency > 600



Then I decided to play with the frequency as well to see if that would produce a better result. The background does look a lot better, but the foreground was not completely separated and there were spots of black in it. This confused me due to how I set up the code, if it is above 600 frequency, it would be black and anything less would be white from the codeword. As in the PDF paper, background has high frequency, foreground has small lower frequency.

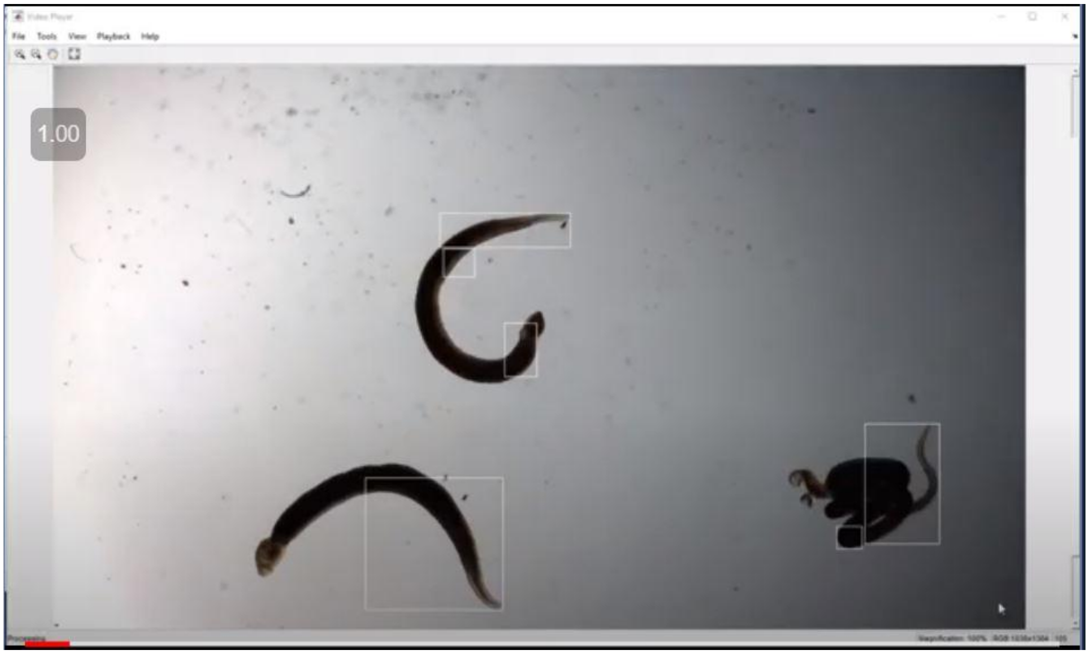
Section: Implement Changing Delta And Playing With Parameters (Brian Nguyen)

The front ground picture with delta (R, G, B) color variation

307 Frames delta=101, N(total number of raw pixels)/2

6. **Experimental evaluations**
    - **Quantitative data shows how well your ideas and implementations work**

Section: First Demo (Brian Nguyen, Anthony Nguyen)



This was the first demo that was completely wrong. Due to thinking that we were tracking the foreground movement and isolating the movement from the background. The feedback we received from this demo, showed me that I had to put more work into this project. At first, I thought that I understood what the project entails, but it was clear that I did not. This demo reflected that due to only using Matlab to produce this result.

Section: Midterm Demo (Brian Nguyen)

In the second demo- I used Matlab to break down the 20-sec compressed video in MOV (H264) format frame by frame. With the uncompressed raw RGB data frame in BMP format. In the BMP format, it contains 1036x1384 pixels for each frame. Each pixel has 3 bytes for Red, Green, and Blue color values, a total of 1433824 pixels for each frame, 440183968 pixels for the entire video, and a total of 1320551904 bytes if the movie data needs to be stored in memory. I wanted to understand how the codebook's algorithm works; because the algorithm is complicated, I broke it down into smaller pieces to grasp the concept better. How I chose to approach this was that I should try to separate the background from the foreground. I did not fully understand the codebook's algorithm at this time around, and I wanted to implement background subtraction first. I was trying to understand how the codewords work, so I had to use an application called Hex Editor Neo to give me information about the BMP. By doing this, I was able to slowly comprehend how the codebook works, but in the PDF it did say that the algorithm is supposed to go through all the frames at once. Due to the time constraint, the best I could do was to have this output for the demo and to implement the codebook to go through all the frames for the next demo. I did consider doing this project in python because I believe there are applicable libraries, but I am more comfortable with java. For some reason, my program did take roughly 15 minutes to process one picture frame, then when I had to run all 307 frame by frame, it took a whole day. Also, the number of pixels for the movie is massive to store in computer memory; the implemented approach is to read pixels for each frame in memory to do calculations, not the entire movie. So the code read frame BMP file twice. The first time is to get the codeword array list from the BMP frame data according to the algorithm. The second time, read the pixels in the BMP file to compare against the codeword array list and determine whether the pixels are foreground/background. In Java, a file cannot be read twice. The 307 BMP frame files are duplicated to read twice. So we pay for the penalty of time to do the file IO read since data cannot be saved in computer memory. It was time-consuming, and I did not realize it would take so long, so I was unable to give myself more time for different demo samples. I knew that the code needed a lot more improvement, but I knew I was on the right path and made decent progress toward a better output.

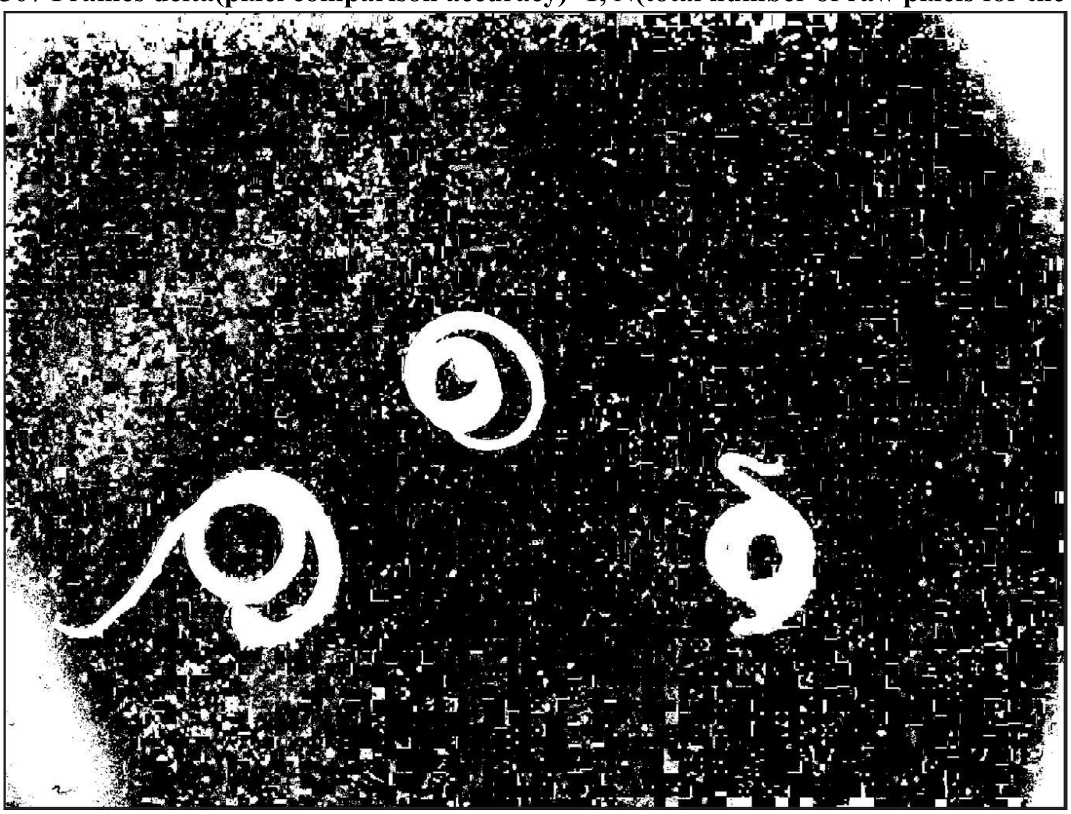
During this demo- my partner was nowhere to be seen and did not contact me at all. He only reached out to me about five days before the demo was due and did not do anything. He did not seem to understand the difficulty of this project and how much time I had already invested into the project. He did not work on the project at all and told me to send the code that I have been working on and that he would fix it. I had emphasized multiple times that even the professor said that this is a hard course that requires a lot of hours and it cannot be done within five days prior to the due date. He did not understand a single thing about the codebook or the algorithm, he decided to talk down to me. I sent him the code and he was unable to do anything to the code or "fix it". I still do not understand how someone thinks that they can tell someone who has been spending time on the project and say to send the code and would "fix it" without understanding what the code is supposed to do. This is also why the first part of the midterm powerpoint was a copy of the proposal powerpoint that I worked on, and I did not understand why he put the same information again.

Section: Last Demo (Brian Nguyen)

With the feedback I received from the previous demo, I knew that I had to reconstruct my code so it could run through all the frames at the same time rather than frame by frame. Before restructuring the code- I reread the PDF to get a better understanding of the algorithm, and then I restructured it. I was able to structure it so that it would take the inputs and run over the number of frames that I set for it. My code would still take a long time to process all 307 frames, so instead of running it all at once, I broke it down to 10 frames. This way I can try to improve the codebook's algorithm and not waste time. I followed the PDF to code the codebook's algorithm, but I could not produce a similar outcome to the PDF. I decided to play around with the parameters to test what works best. The pictures that are shown below show the different approaches I took. I decided that I wanted to see how these would look once I ran it through all 307 frames, and due to my code taking a day and a half to run, it was very time-consuming. I did want to find a way to cut down the time it took my code to run through the 307 frames, but I did not have the time to optimize it. I also wanted a code that works rather than a code that doesn't work, and I preferred having a working demo.

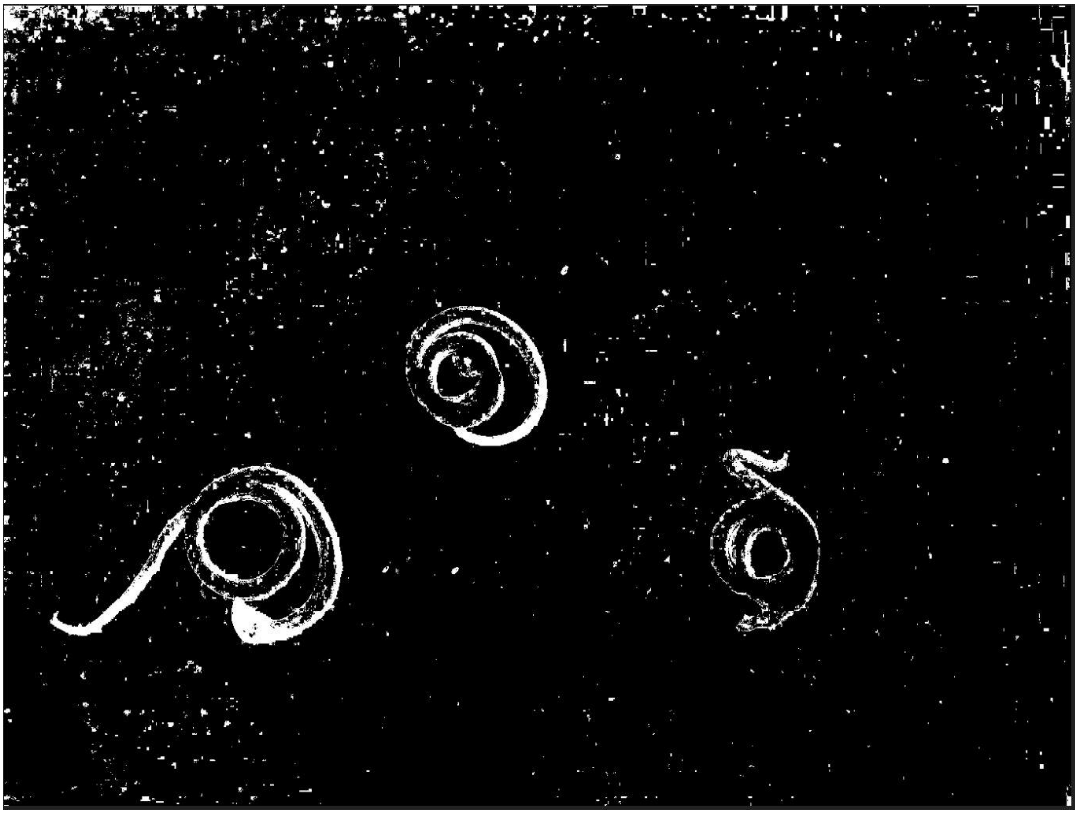
For the last demo- I told my partner that he need to do work because I was not willing to allow him to take credit for my work. He only reached out to me a week before the project was due to ask how to improve the code. He did not understand what the professor's feedback meant and did not know how to move forward with the project seeing that he did not understand the codebook's algorithm. He did not understand the basic concept of what the codebook does. I truly felt that he did not deserve to pass the class if he did not understand what the project entails. He continued to give me excuses for why he could not work on it this whole time. It did not feel right to me and I was going to tell the professor that he was not helping me in any shape or form, and did not understand the project's concepts or the project goal. It was only when I made that declaration to him that he decided to put some effort into the class. When he sent me his output and what he used, to me it seemed like he did not do it himself and that he only did slight modifications to it. Since it seems like he still couldn't explain the algorithm, and the video (x, y) position was off and it was backward. This is why we had two different demos for the final presentation.

307 Frames delta(pixel comparison accuracy)=1, N(total number of raw pixels for the movie)/174



I was able to get the foreground isolated, but I could not figure out how to deal with the background with the intensity spiking.

307 Frames delta=1, N(total number of raw pixels)/60, frequency > 600



Then I decided to play with the frequency as well to see if that would produce a better result. The background does look a lot better, but the foreground was not completely separated and there were spots of black in it. This confused me due to how I set up the code, if it is above 600 frequency, it would be black and anything less would be white from the codeword. As in the PDF paper, background has high frequency, foreground has small lower frequency.

Section: Current Verison (Brian Nguyen)

The front ground picture with delta (R, G, B) color variation
307 Frames delta=101, N(total number of raw pixels)/2

This was the result of my most recent using delta 101. It looks a lot better, but at the same time, I ran into time constraints, and I was unable to compile all of them back into a movie using Matlab. If I had a more efficient running time, I would have gotten to a better one. Inside my output file would be my previous attempt.

7. **Conclusions**

   In conclusion- this project was challenging in many different ways. One of them would be that having a reliable partner is necessary to take on a challenging project like this. It was also challenging because it was my first time reading a research paper and understanding the concept thoroughly. Then translate the theory into useable and working code. I can see how this will be useful in the real world because at any time the PM could give me a new project to work on, and I would have to learn about the topic and then translate it into code. The presentations were a good way to practice doing presentations over zoom and to get over the hurdle of being anxious. It is true that during code review or other kinds of presentations, there will be people that review your code, and you will have to explain it. I was able to learn more about how videos and image processing work through code. As for the result of the project, the goal was to have a foreground and background separation and to investigate how the lambda parameters could be determined automatically. I tried my hardest and played with the parameters and then tried to make it in a way that could be done automatically, but I was unable to obtain that goal. With how my program is at the moment and how long it takes for the program to run through all the images, it was very time-consuming which made it harder to get better results. The next step I would take- is to find a way to optimize my code into a more efficient way to use the codebook's algorithm to process the images for the video faster. That way, I could find the parameters that could be determined automatically. There are always more improvements that could be done.

8. References

Kim, Kyungnam, et al. “Real-Time Foreground–Background Segmentation Using Codebook Model.” *Real-Time Imaging*, vol. 11, no. 3, 2005, pp. 172–185., https://doi.org/10.1016/j.rti.2004.12.004.