CMPEN 454 Project 3 Report

Part a. Summary

Refer to `README.md` for the project summary.

Part b. Procedure Approach with Flowchart

Our program is pretty straightforward, and everything was kept in the main function. There is only one loop involved as it iterates through all the files inside the directory. We also output all the files into the same directory named "processed-images" for the convenience of video generation.

The background frame `B(0)` and `H(0)` were generated before looping through files. And we started the loop from the fourth file in the directory as first two are "`.`" and "`..`" and the third file (first actual frame) would generate nothing but a all-black image. We generated greyscale images by simply taking the green channel. We generate binary images using built-in `im2bw` function. We also normalized result for Persistent Frame Differencing in the end so that all four pictures generated are of the same intensity level.

Data Structure wise, we kept all four algorithms completely independent from each other. A detailed flowchart of our program can be found on next page.

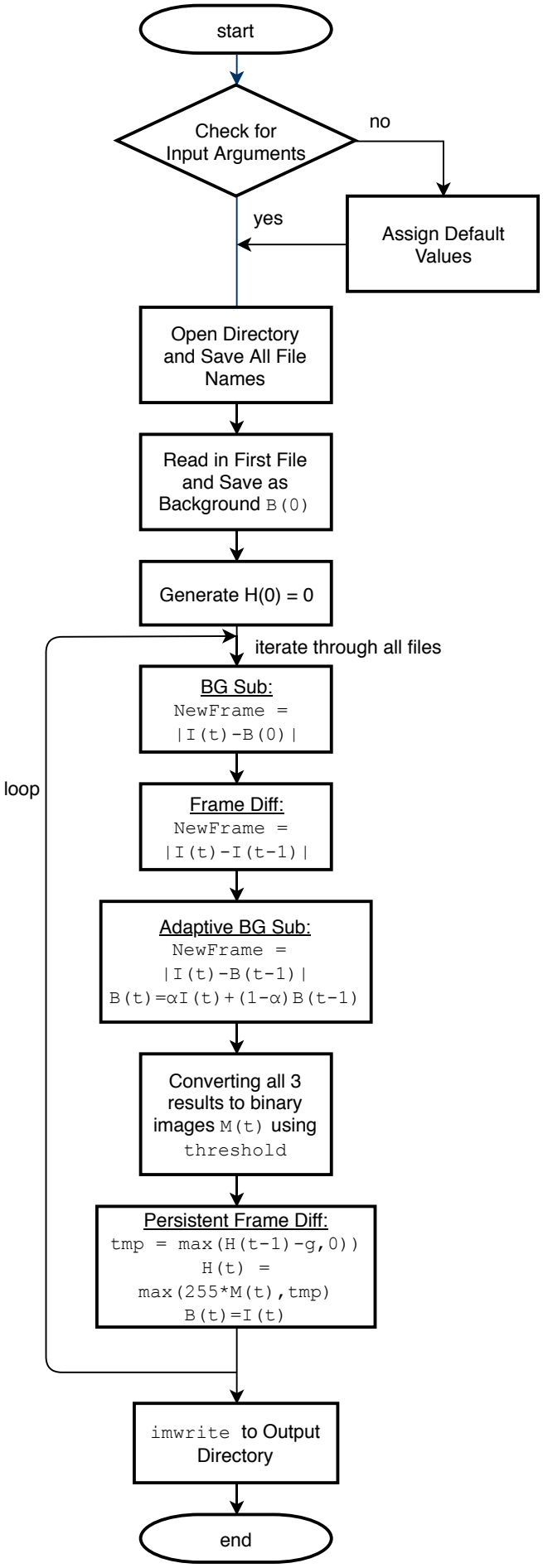
Part c. Experimental observations.

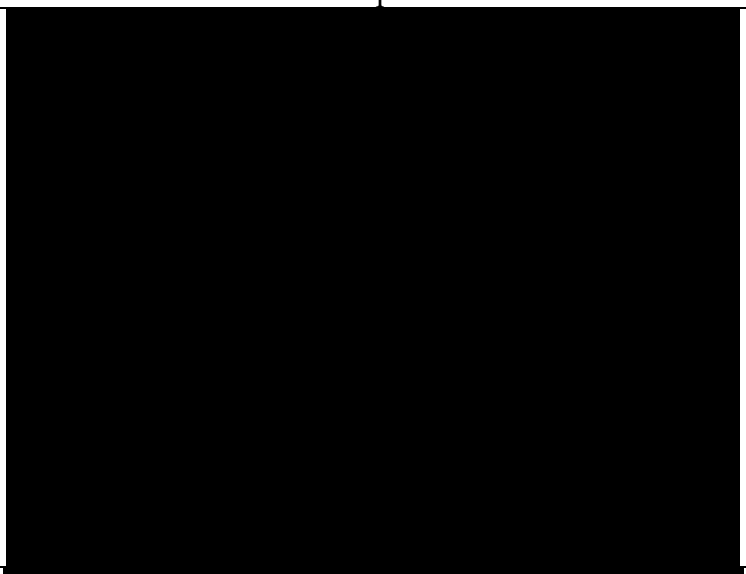
1. `threshold` value

   This is the pixel difference threshold value for determining what level of intensity change is needed to declare there has been a significant change at a pixel, not just sensor noise. In our program, we used MATLAB's built-in `im2bw` function to generate a binary image (i.e. the output results). The function takes in a `threshold` parameter between 0 and 1. The default threshold for our program is 0.08. You can also change it by manually input it in the parameter when calling `project3` function.

   We decide the default value by manually testing values out and compare the output images at the same frame (`f0125`). As shown below (`getin` dataset was used here), when we first run the program with threshold of 0.5, all the frames are dark as the intensity change was not big enough to be detected. So we changed the value to 0.1. As shown below, now the moving person can be detected. However, not all parts of the person were detected as color of those parts could be closer to the background color and hence was not picked-up. Thus we ran the program again with threshold of 0.08. Now we can see a fuller person, but some non-moving parts were also picked up. This is due to noise and our program thought those were moving objects too.

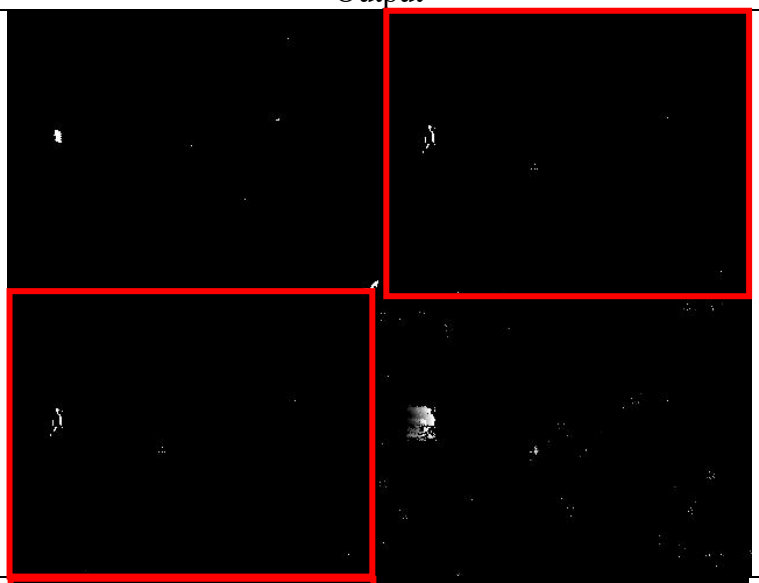
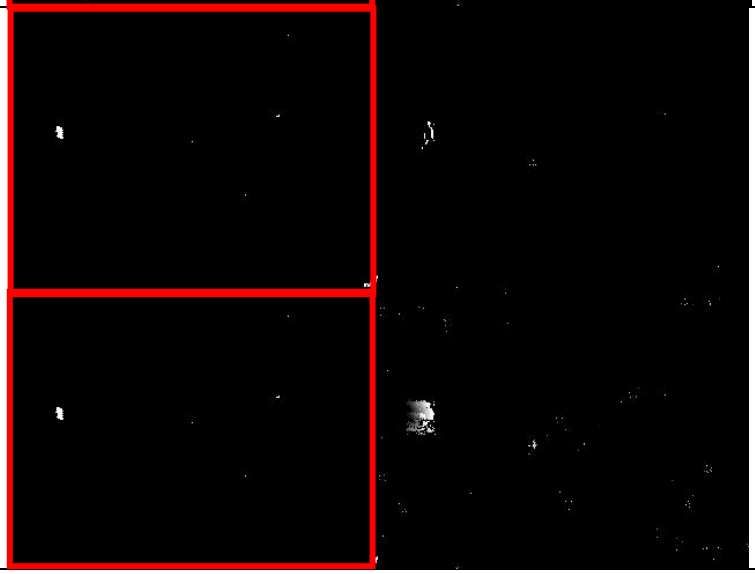
Program Flowchart



Threshold =	Output
0.5	
0.1	
0.08	

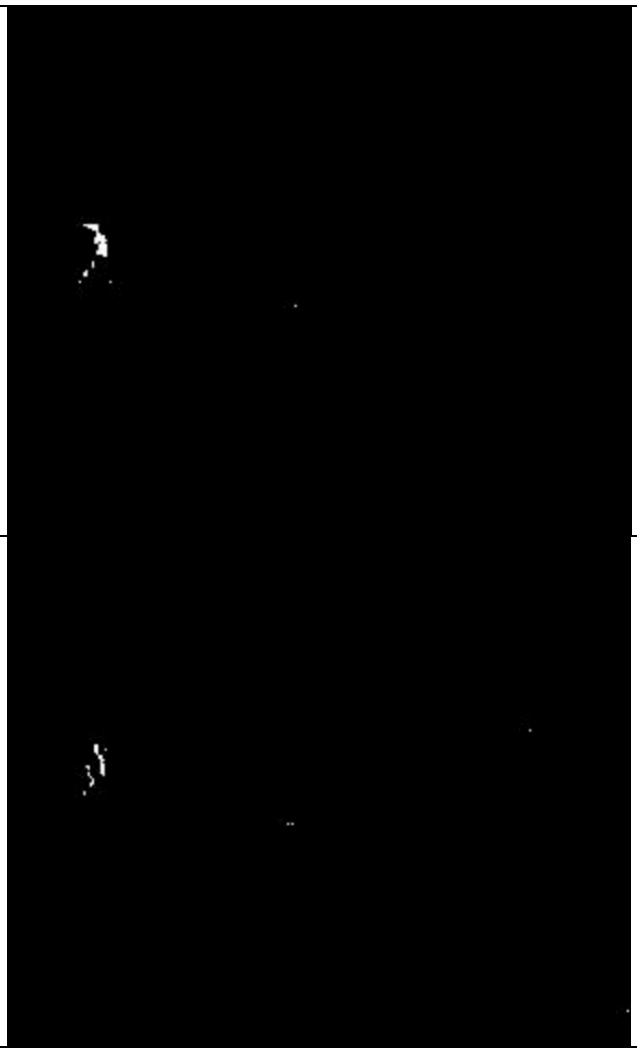
2. Alpha Value (`absAlpha`)

This parameter (for adaptive background subtraction) is used for merging new images into the background model. When $\alpha = 1$ yields simple background subtraction; $\alpha = 0$ yields frame differencing (verified and shown below using `getin` dataset at frame `f0125`).

`absAlpha` =	Output
0	
1	

In addition, different values in between 0-1 yield different output results – alpha value is directly proportional to decay rate. Meaning that higher alpha value keeps less trails from the previous frames while higher alpha value keeps more (verified and shown below).

`absAlpha` =	Output

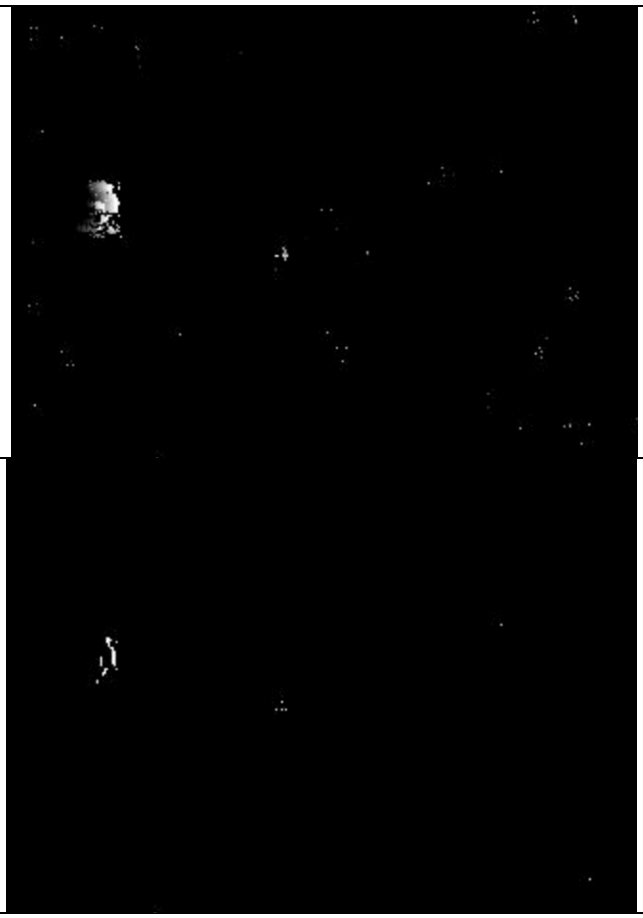
0.15 (more trails)			
0.85 (less trails)			

3. Gamma Value (`pfdGamma`)

This is the linear decay parameter for the persistent frame differencing algorithm. The default value was set to 10 as it gives more fading trails for the moving objects and shows the direction of the movements.

This parameter determines how many previous frames are kept while calculating the output result. The value ranges 0-255 as the highest intensity is 255. This algorithm gives motion images combined with a linear decay term, as well as provides motion history of the movement. For big Gamma value, $H(t-1)-\gamma$ decays quickly and hence less motion history is kept and fewer fading traits can be seen in the result. Similarly, for small Gamma value, the term decays slowly and leaves a fading trail showing movement direction as a result.

`pfdGamma` =	Output

15 (more fading trails)		
240 (less fading trails)		

Part d. Experimental Comparisons

1. Algorithm Comparison and Analysis (Why the Failures)

Background Subtraction and Frame Differencing are the simplest algorithms to implement here. Background Subtraction simply assumes the first frame as background and doing subtraction to show movement. However, if an object is already there in the first frame, it will leave a hole in all future frames (shown in Fig. d1); and similarly, if the movement stops, this algorithm keeps detecting such object although it's not moving any more (Fig. d2). Frame Differencing algorithm solves both issues (Fig. d1 & d2) as it subtracts with the previous frame value and is actively updating the "background". Background Subtraction also performs poorly when dealing with moving cameras.

However, Frame Differencing only detects the leading and trailing edge of a uniformly colored object, result in labelling very few pixels on the. In addition, it is difficult to detect an object moving towards or away from the camera. And that where the third algorithm comes in – Adaptive Background Subtraction is more responsive to changes in illumination and camera motion. Fast small moving objects are well segmented, but they leave behind short "trails" of pixels. Objects that stop, and ghosts left behind by objects that start, gradually fade into the background. (Fig d3)

Similarly, Persistent Frame Differencing algorithm is also responsive to changes in illumination and camera motion, with stopped objects / ghosts fading away. But in this

algorithm, objects leave behind gradually fading trails of pixels, indicating an apparent direction of object motion in the image. Although the centers of uniformly colored objects are still not detected, the leading and trailing edges are made wider by the linear decay, so that perceptually (to a person) it is easier to see the whole object. (Fig d3)

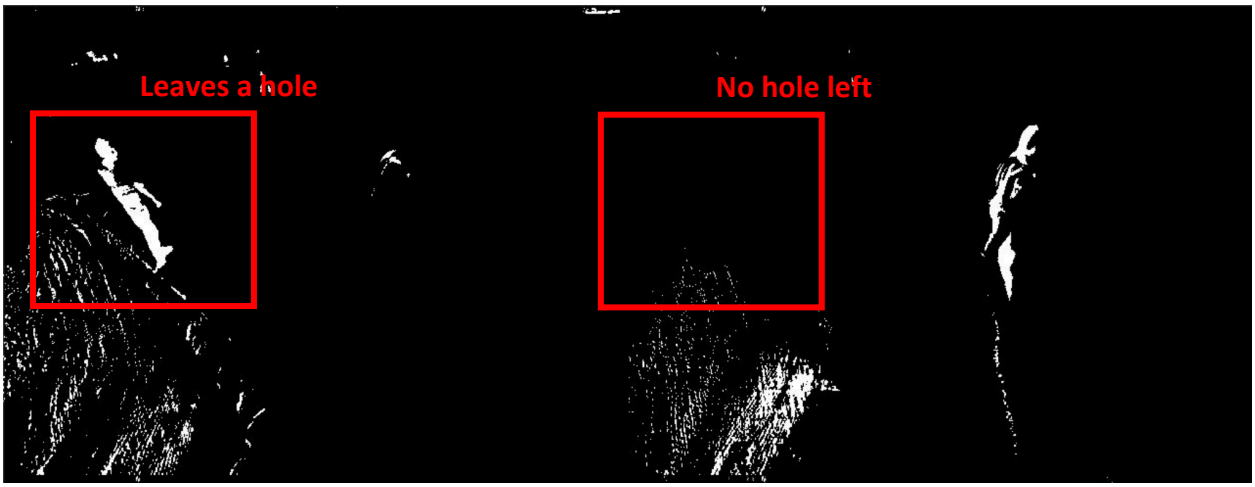


Fig. d1

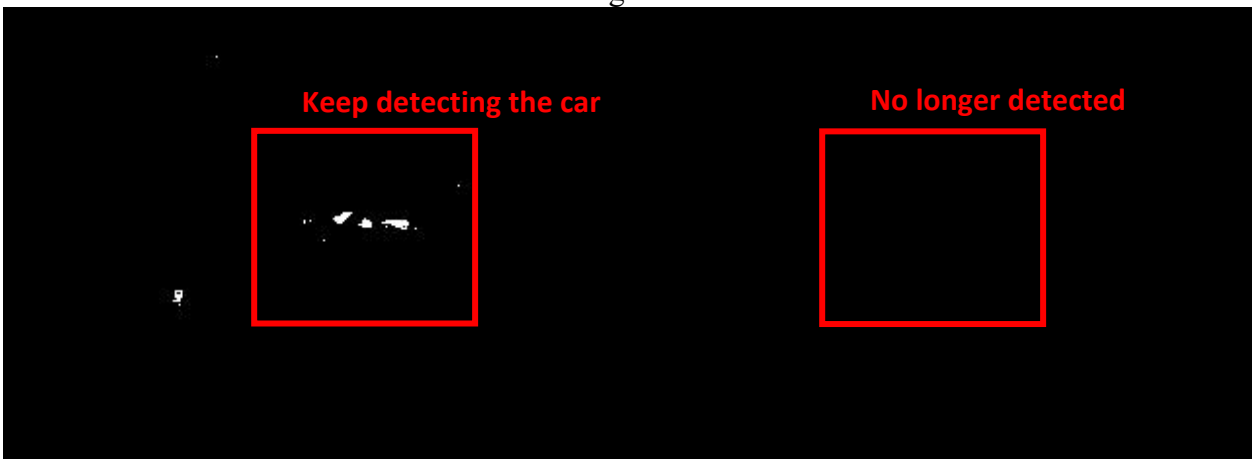


Fig. d2

Fig d3

2. My Preference

Overall, I prefer Persistent Frame Differencing. Although Background Subtraction is easy to implement, the algorithm has too many flaws and is not doing a good job to detect objects that are actually moving. And Frame Differencing just did not give a full and clear shapes of the objects. I think it's really cool and useful that Persistent Frame Differencing not only tracks video change but also shows the directions of the movements. At the same time, objects detected maintain full and proper shapes.