CMPEN 454 Project 3

Team Members: Brian Nguyen, Kyle Bradley, Jordan Reed, Drice Bahajak

Summary

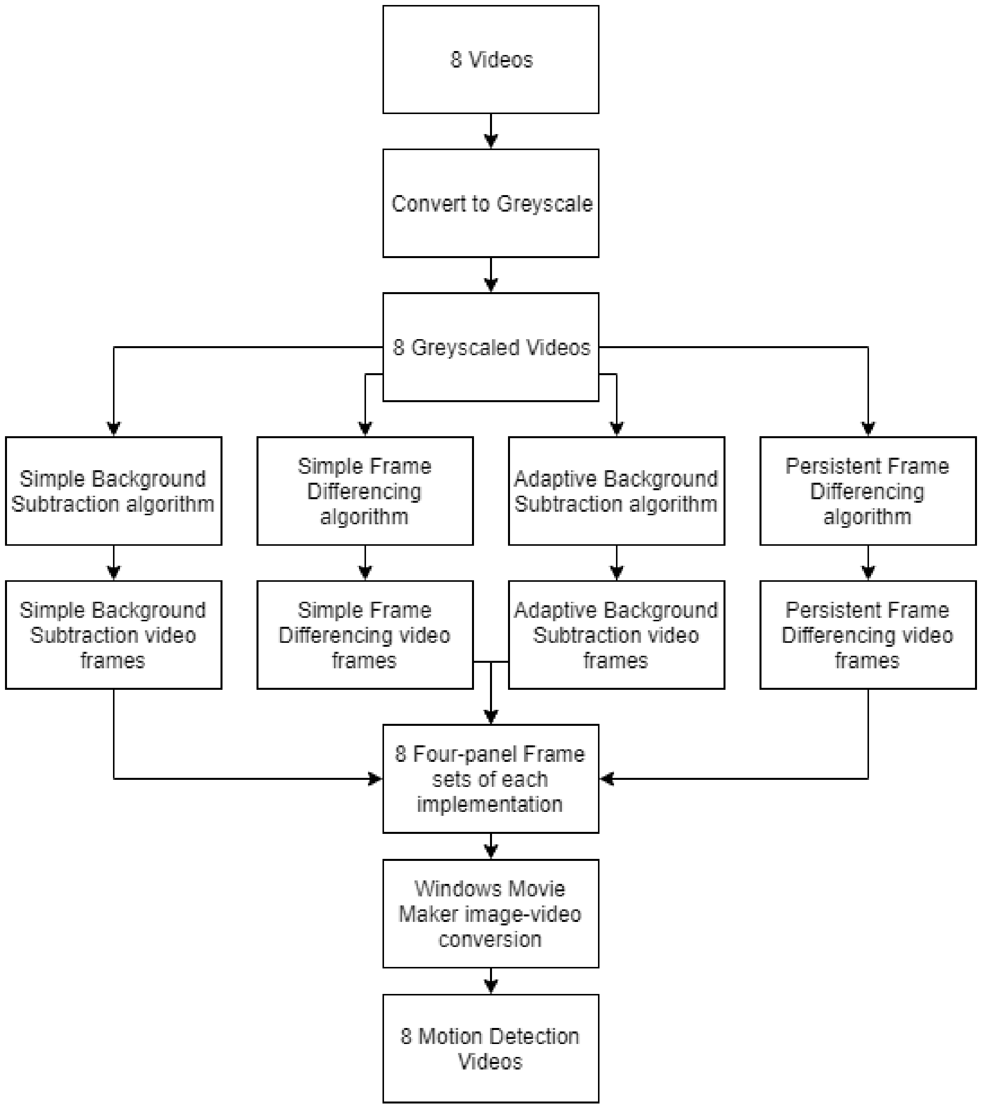
The goal of this project is to utilize Matlab to implement motion detection algorithms that we have learned in class. The four algorithms that we have implemented are simple background subtraction, simple frame differencing, adaptive background subtraction, and persistent frame differencing. After our implementation, we applied them to eight separate videos to create videos of the motion detection results. This allowed us to inspect the pros and cons of each algorithm. By checking the efficiency of our implementations, we can see how fast our whole project runs, as well as the individual algorithms. From this project, we expected to achieve a firm understanding of each algorithm after implementation, and have a clear understanding of the differences between them. Since the evaluation is primarily qualitative, we expect that the output videos of each algorithm, for each video are what is expected from our understandings of the algorithms. We ensured this by testing different parameter values for the threshold, alpha, and linear decay. This also helped us understand how these values affect the output and how to optimize them.

Procedure Outline

For this project, we were given eight separate videos (in the form of thousands of images) to run the four motion detection algorithms on each of them.

For our overall design, we decided to separate the creation of each algorithm so that they can be used independently from each other in case of any independent algorithm failure. This also allowed for each team member to have a take on a single algorithm.

Below is a flowchart and explanation of each algorithm implementation, the video creation, and other additional steps taken.

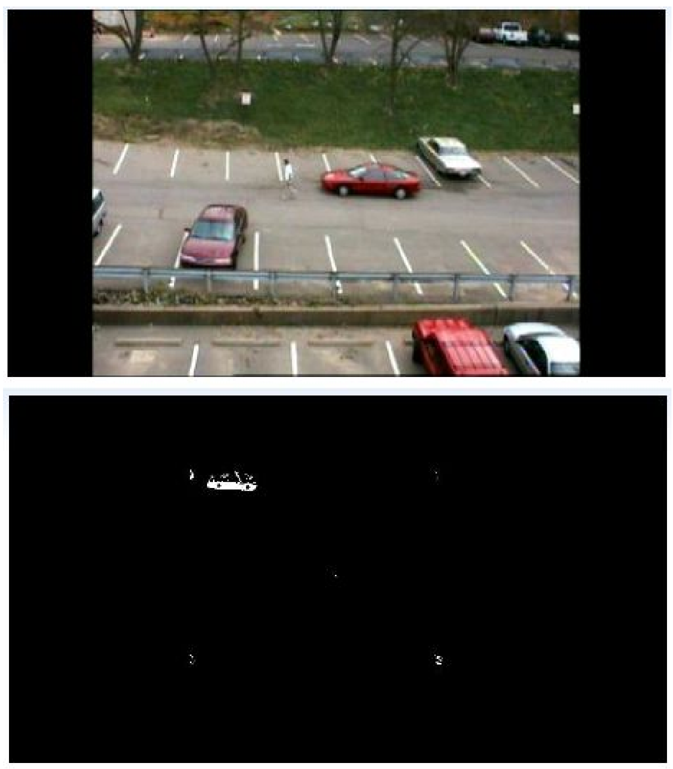

8 Videos
Convert to Greyscale
8 Greyscaled Videos
Simple Background Subtraction algorithm
Simple Frame Differencing algorithm
Adaptive Background Subtraction algorithm
Persistent Frame Differencing algorithm
Simple Background Subtraction video frames
Simple Frame Differencing video frames
Adaptive Background Subtraction video frames
Persistent Frame Differencing video frames
8 Four-panel Frame sets of each implementation
Windows Movie Maker image-video conversion
8 Motion Detection Videos

- Video Input format
    - We loaded in the provided videos by creating 8 matrices of dimensions (size of directory, 480, 640, 3). For each frame, we would use imread() to load in that frame to its corresponding index (position in the directory). This process was repeated for each frame for each video. This allowed us to have sorted matrices with respect to time for each video.
- Converting to Grayscale
    - The RGB images for each frame of each video were converted to grayscale by dividing the summation of the RGB values by 3
    - gimg = (r + b + g) / 3
- Algorithm Implementation
    - Simple Background Subtraction
        - This algorithm takes the absolute difference between a frame and the frame of a static image that has no present objects, and labels all of the pixels for the frame as a 1 or 0 depending on whether the absolute pixel differences are greater than the given threshold.
    - Simple Frame Differencing
        - This algorithm takes the absolute difference between a frame and the frame before it, and labels all of the pixels for the frame as a 1 or 0 depending on whether the absolute pixel differences are greater than the given threshold.
    - Adaptive Background Subtraction
        - This algorithm combines the ideas of frame differencing and background subtraction. At each step the current background is “blended” with the previous frame using a weight alpha. If alpha is 1, we get the same as simple frame differencing and if alpha is zero we get the same as simple background subtraction.
    - Persistent Frame Differencing
        - This algorithm takes the absolute difference between a frame and the previous frame. This then combines the images with the linear decay term. Therefore, there is a trail of pixels behind the object that fades away.
- Four-Panel Frame Creation
    - After getting processing each video using the different algorithms we combined each frame into a four panel box.
- Motion Detection Video Creation

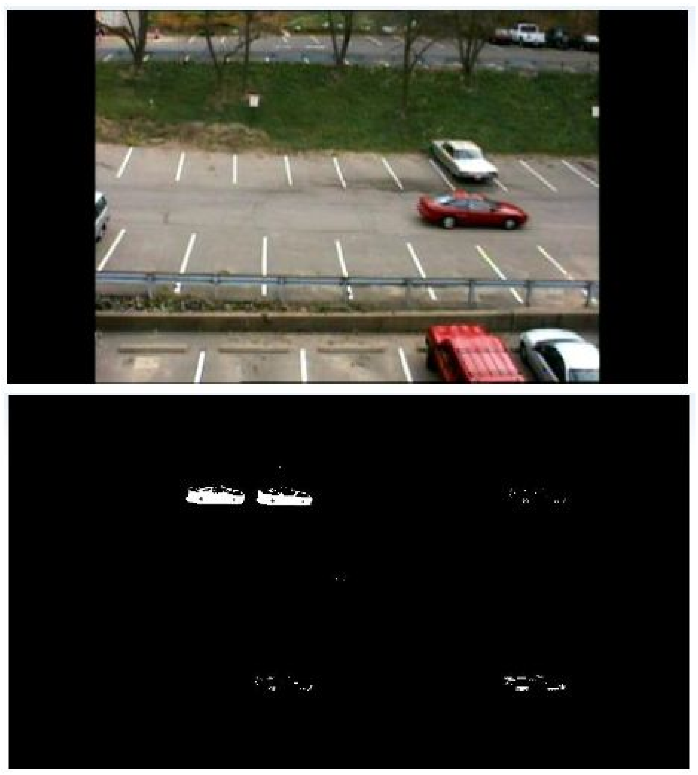
    - We chose to use Windows Movie Maker to create our resulting videos. We used its simple tool to create a video based on numbered image files and output the videos as an mp4 file.
- Efficiency
    - By running profile() on our project, we were able to analyze and quantify the efficiency of our project as a whole, as well as the individual algorithm implementations.

Qualitative Results

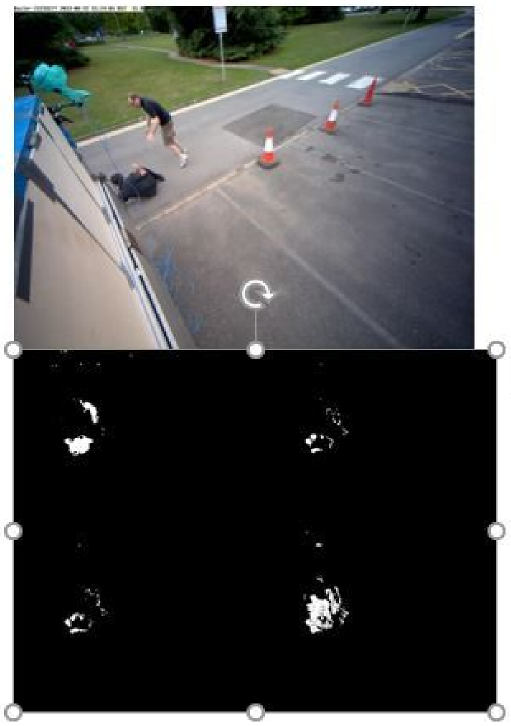
Below is one quad frame showing the results from each of the eight videos. There are times where all four boxes capture the image equally and there are times where it is obvious the background subtraction shows the best.



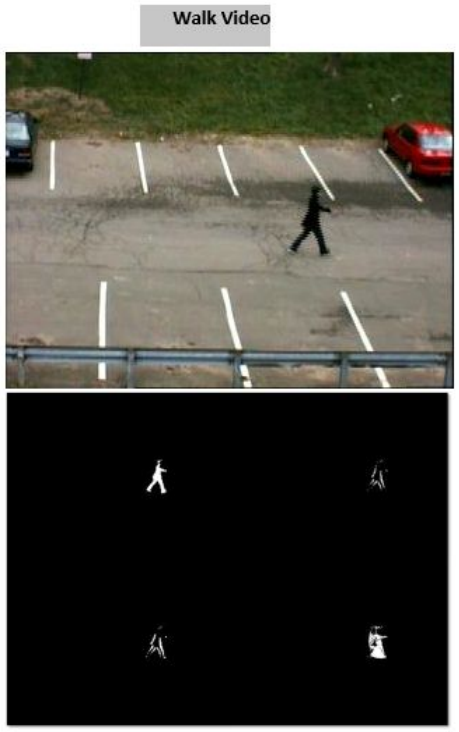
Get out Frame 472



Get in Frame 522

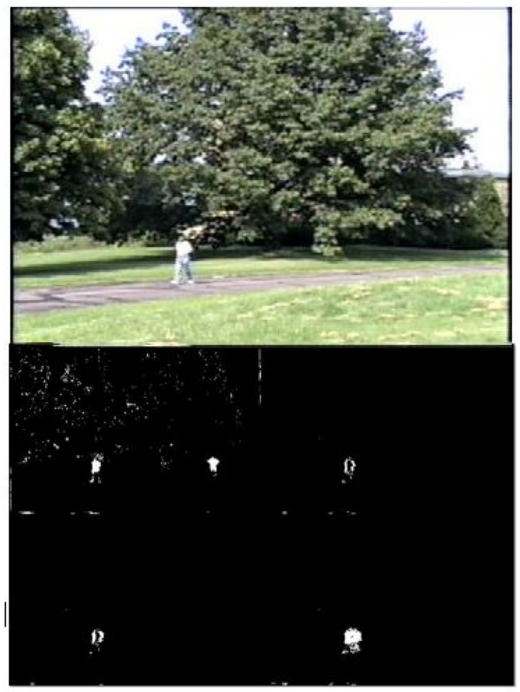


Arena W Frame 96

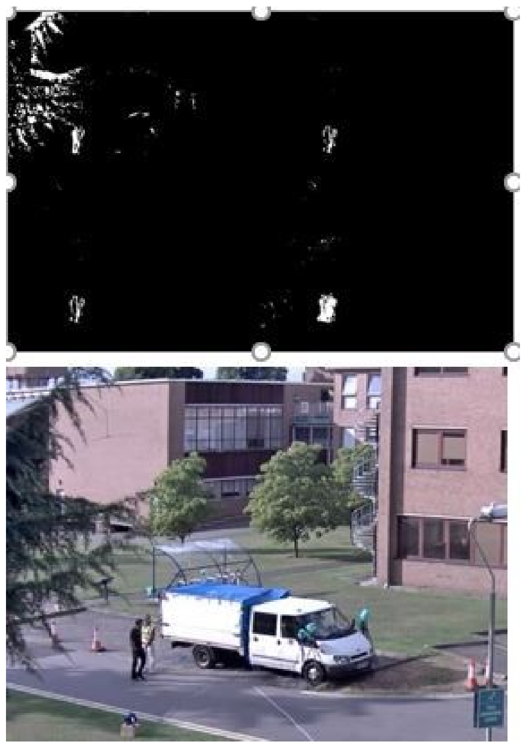


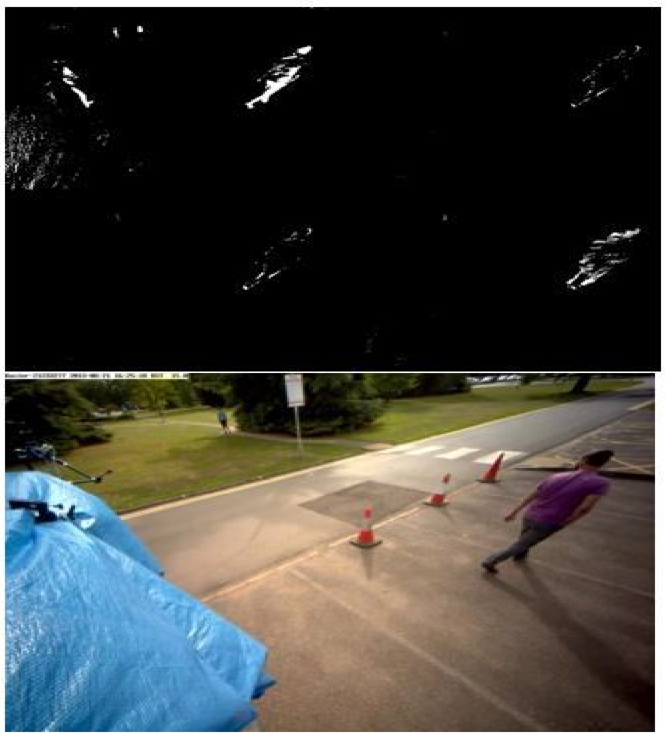
Walk Frame 204

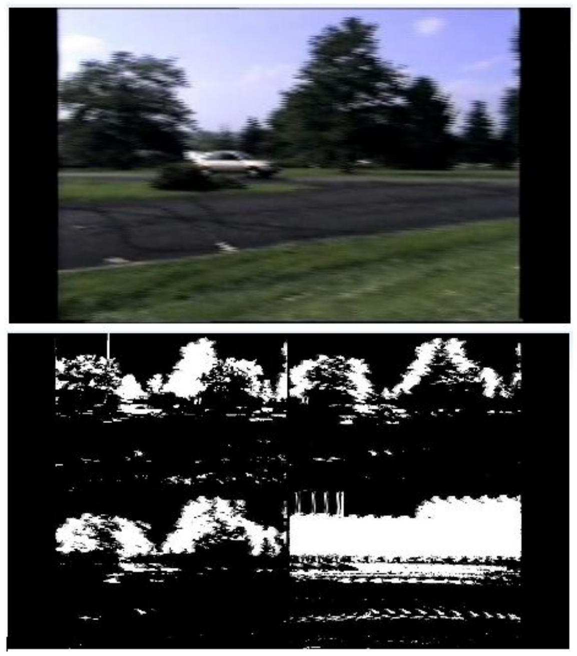
Trees Video



Trees Frame 118

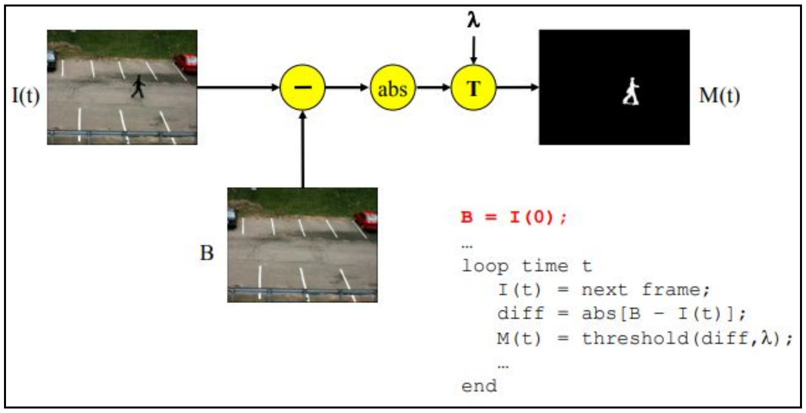

Arena A frame 122


Arena N Frame 96


Movecam Frame 163

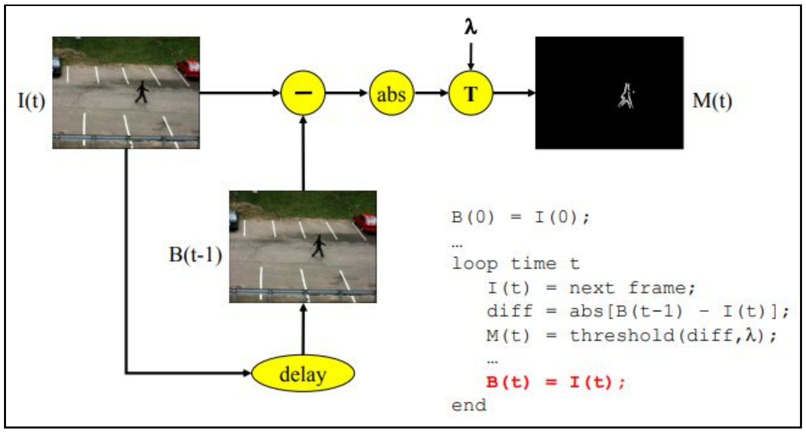
Mathematical Equations / Pseudocode for the Four Motion Detection Algorithms

Simple Background Subtraction



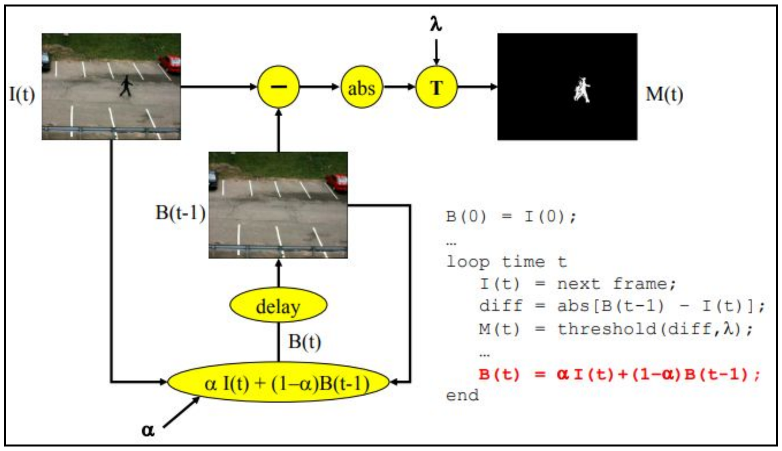
```
B = I(0);
…
loop time t
   I(t) = next frame;
   diff = abs[B - I(t)];
   M(t) = threshold(diff,λ);
   …
end
```

Simple Frame Differencing



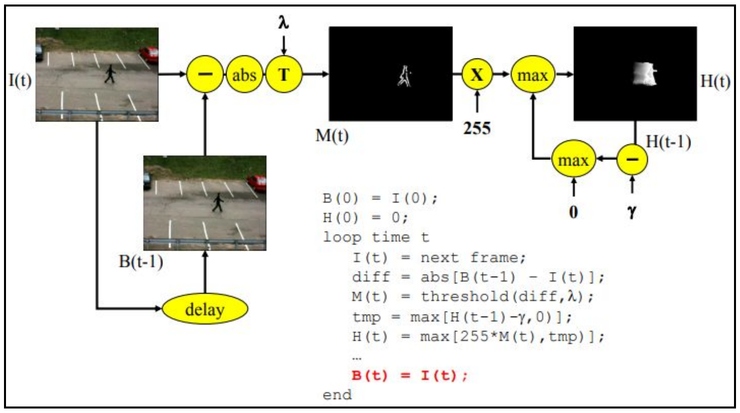
```
B(0) = I(0);
…
loop time t
   I(t) = next frame;
   diff = abs[B(t-1) - I(t)];
   M(t) = threshold(diff,λ);
   …
   B(t) = I(t);
end
```

Adaptive Background Subtraction



```
B(0) = I(0);
…
loop time t
   I(t) = next frame;
   diff = abs[B(t-1) - I(t)];
   M(t) = threshold(diff,λ);
   …
   B(t) = αI(t)+(1−α)B(t-1);
end
```

Persistent Frame Differencing



```
B(0) = I(0);
H(0) = 0;
loop time t
   I(t) = next frame;
   diff = abs[B(t-1) - I(t)];
   M(t) = threshold(diff,λ);
   tmp = max[H(t-1)-γ,0)];
   H(t) = max[255*M(t),tmp)];
   …
   B(t) = I(t);
end
```

Quantitative Results

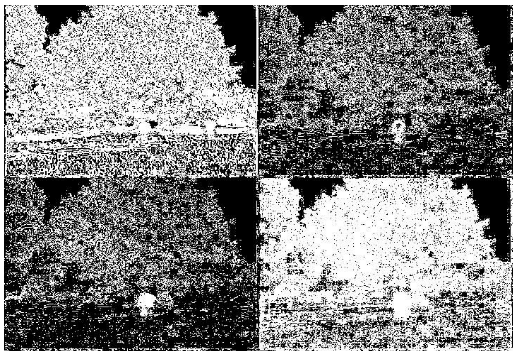
We decided to set our threshold value to 60 for the four algorithms. We chose this number after various trials of different threshold values by increments of 10. This threshold value was general and effective enough so that we were not having too much noise, and we were still getting significant areas of motion detection. By testing with lower values, we noticed that we were not seeing as much motion detection as we were supposed to. By testing with higher values, we noticed that we were picking up the motion detection, but also a lot of noise from the background that was not moving.

We chose the alpha parameter value of .5 for the adaptive background subtraction algorithm. We chose this because we wanted to be in the middle of the two extremes of having the same as background subtraction and the same as frame differencing.

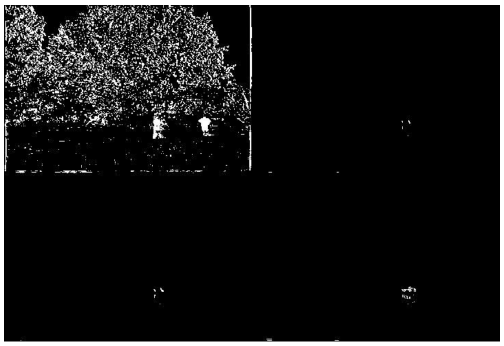
We set the linear decay parameter value to 50 for the persistent frame differencing algorithm. We experimented with a few values, ranging from 5 to 250, and found a relatively low value best illustrated the motion of the object without showing too much blur and confusing the viewer.

Experimentation with High/Low Threshold values:

Too Low:



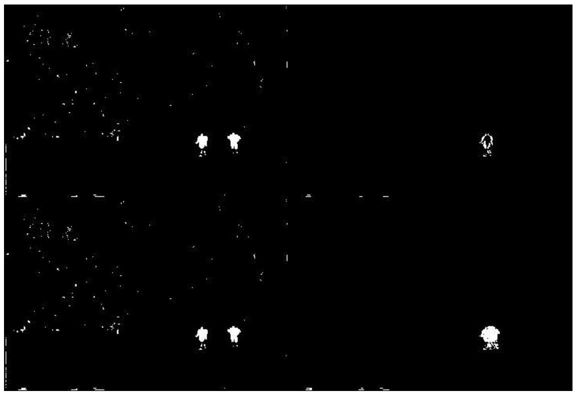
Too High:



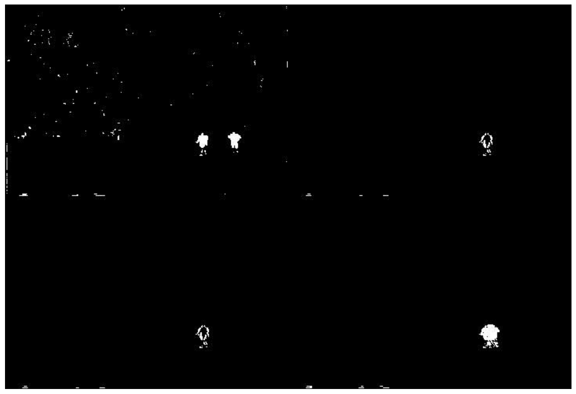
As seen above, setting the threshold too low makes the result too sensitive to noise, but setting the threshold too high makes the result not very effective.

Experimentation with One vs Zero Alpha:

Zero Alpha:



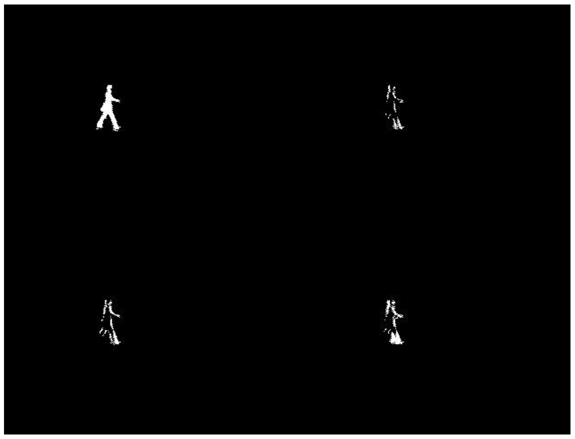
One Alpha:



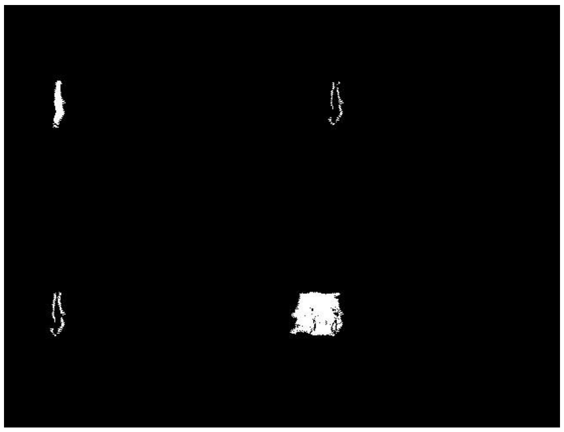
As seen above, setting the alpha to zero makes the adaptive background subtraction the same as background subtraction and setting it to one makes it the same as frame differencing.

Experimentation with High/Low Gamma:

High Gamma:



Low Gamma:



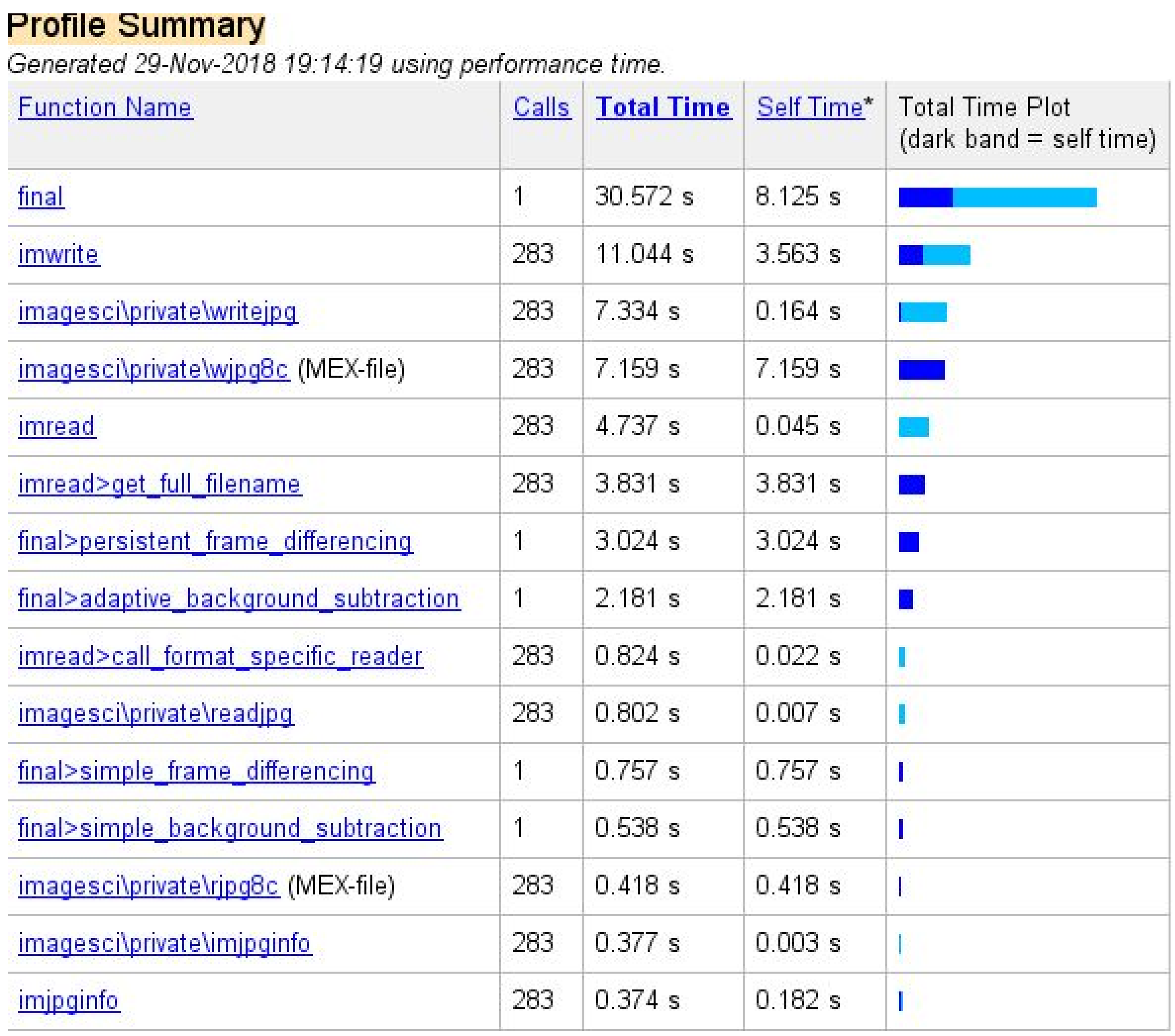
As seen above, a high gamma makes there be almost no trail, but a low gamma makes the trail very long.

The experimental observations for the testing of each of the discussed parameters can be seen in the above section.

Experimental Observations

The simple motion detection project was very interesting. There were many little things that you could change to make the videos a bit different, like alpha, gamma, and the threshold values as seen above. Each of the four frames perform a different task, but if you were just looking to see motion, the background subtraction is the easiest for the eye to follow. The persistent frame differencing is unique in the sense that you can "look in the past" with the shadow behind the moving object.

Algorithm Efficiency



Profile Summary

Generated 29-Nov-2018 19:14:19 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
final	1	30.572 s	8.125 s	
imwrite	283	11.044 s	3.563 s	
imagesci\private\writejpg	283	7.334 s	0.164 s	
imagesci\private\wjpg8c (MEX-file)	283	7.159 s	7.159 s	
imread	283	4.737 s	0.045 s	
imread>get_full_filename	283	3.831 s	3.831 s	
final>persistent_frame_differencing	1	3.024 s	3.024 s	
final>adaptive_background_subtraction	1	2.181 s	2.181 s	
imread>call_format_specific_reader	283	0.824 s	0.022 s	
imagesci\private\readjpg	283	0.802 s	0.007 s	
final>simple_frame_differencing	1	0.757 s	0.757 s	
final>simple_background_subtraction	1	0.538 s	0.538 s	
imagesci\private\rjpg8c (MEX-file)	283	0.418 s	0.418 s	
imagesci\private\imjpginfo	283	0.377 s	0.003 s	
imjpginfo	283	0.374 s	0.182 s	

This profile was generated for only one dataset, the walk dataset. For this run of profile(), we just changed our code so that our script ran on one directory. After we rewrote some functions to remove inefficient code, our most expensive operations were reading the input jpegs and writing our newly created jpeg files to a file in our local environment, which is unavoidable. Our algorithms took mere seconds to run over each frame in the dataset.

Contributions

	Tasks Done
Brian Nguyen	● Simple background subtraction implementation ● Simple frame differencing implementation ● Summary and procedure portions of report ● Cleaned up code and added comments ● Made code run for all 8 videos ● Transferred frames to videos
Kyle Bradley	● Adaptive background subtraction implementation ● Experiments with Alpha,Gamma,Threshold ● Image Preprocessing and Output code
Jordan Reed	● Persistent frame differencing implementation, wrote code ● Tested code with profile()
Drice Bahajak	● Persistent frame differencing implementation, qualitative results, observations.