EE 454 Computer Vision

Project 3

Motion Detection

November 30th, 2018

Gursahej Gandhoke

Emmanuel F. Werr

Project Outline and Implementation description:

The goal of this project is to implement the four simple motion detection algorithms described in Lecture 23, run them on short videos, and compare the results. The four algorithms are:

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) = α I(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
```

All four should be implemented in on program to generate, as output, a four-panel frame showing the results of each algorithm, on each video frame, and generate a video of the results.

Procedure Outline and Experimental Observations

We are going to show working code for one specific video “walk”, since the algorithm implementation is the same for all the videos with a change in the value of lambda.

- Reading

  We need to read in all the images to be processed before we do anything else. We read the first frame outside the loop so that we can get dimensions for the images that would be used to allocate memory for variables across the project, such as greyImages, M(t), B(t) etc. We then convert these images into greyscale inside the loop. This is what our workspace looks like for the reading in the image frames for the video “walk” :

baseFileName	'f0283.jpg'
columns	320
filePattern	'/Users/gursahejsingh/Deskto...
fullFileName	'/Users/gursahejsingh/Deskto...
greyImages	*243x320x283 double*
imageArray	*243x320x3 uint8*
k	283
lambda	40
myFolder	'/Users/gursahejsingh/Deskto...
rows	243
theFiles	*283x1 struct*

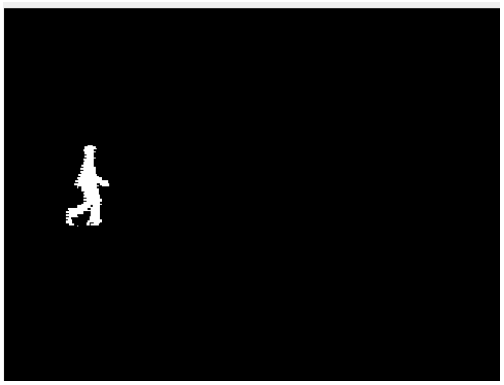
  Since walk has 283 frames, we can see that greyImages is of the size = (frame_row_pixels, frame_column_pixels, 283 frames), therefore it has read in all the greyscale frames.

  All the following algorithms are independent of each other and just need the reading section to be run first before they can be run themselves. We wanted to create a modular type code so that it is easy to debug. We use for instead of while since we know the bounds which makes it faster.

- Simple Background subtraction

  Now we come on to the first algorithm, the pseudo code for which is mentioned above. Firstly, we are allocating memory to all the variables that need it, secondly we go on to run a for loop until the number of frames, and all the necessary computations are done in that loop. Instead of a nested for loop, we chose to use a conditional statement to update the pixel values from lambda, since it is much faster (More in optimization). Look at the following images:

lambda	40
M_bs	*243x320x283 double*
myFolder	'/Users/gursahejsingh/Deskto...
output	*243x320 double*
rows	243
theFiles	*283x1 struct*

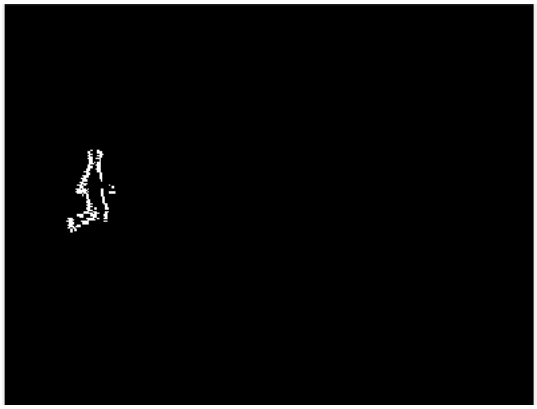


M_bs is basically my M(t) for this algorithm. We can see M(t) gets filled with as many values as greyImages so we know part of it is working properly, our doubt is removed when we see the output.

- Simple frame differencing

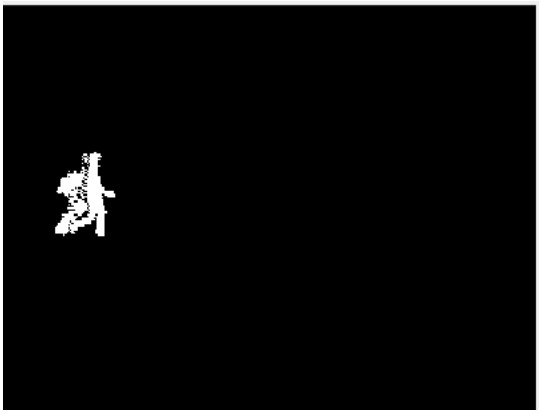
  Basically, we use the same principles as we did above- allocate variables, run for loop and computations inside it. In this case M(t) is M_fd and B is B_fd, B_fd is basically a history. The following images confirm its working:

B_bs	*243x320 double*
B_fd	*243x320x283 double*
baseFileName	'f0283.jpg'
columns	320
diff	*243x320 double*
filePattern	'/Users/gursahejsingh/Deskto...
fullFileName	'/Users/gursahejsingh/Deskto...
greyImages	*243x320x283 double*
i	283
I	*243x320 double*
imageArray	*243x320x3 uint8*
k	283
lambda	40
M_bs	*243x320x283 double*
M_fd	*243x320x283 double*
myFolder	'/Users/gursahejsingh/Deskto...
output	*243x320 double*
rows	243
theFiles	*283x1 struct*



- Adaptive Background Subtraction

  Again, like the other algorithms meantioned above but in this case we introduce a new variable called alpha, for blending. Its values are 0 to 1, Closer to zero means that it is leaning towards simple background subtraction, closer to 1 means it is leaning towards simple frame differencing. We chose the alpha as 0.1 for the “fiery man effect”. Look at the following images :



imageArray	243x320x3 uint8
B_abs	243x320x283 double
B_fd	243x320x283 double
greyImages	243x320x283 double
M_abs	243x320x283 double
M_bs	243x320x283 double
M_fd	243x320x283 double
theFiles	283x1 struct

- Persistent frame differencing

  Same as others, but instead of alpha we have the concept of gamma for delay. Gamma is the decay value for the history of the background H(t), which is the final output. Gamma came out to be 15-20, and we had to divide the final H(t) by 255 to show the shadow effect in the output frame. Here are the outputs :

alpha	0.1000
columns	320
gamma	17
i	283
k	283
lambda	40
rows	243
baseFileName	'f0283.jpg'
myFolder	'/Users/gursahejsingh/Deskto...
filePattern	'/Users/gursahejsingh/Deskto...
fullFileName	'/Users/gursahejsingh/Deskto...
B_bs	243x320 double
diff	243x320 double
I	243x320 double
output	243x320 double
tmp	243x320 double
imageArray	243x320x3 uint8
B_abs	243x320x283 double
B_fd	243x320x283 double
B_pfd	243x320x283 double
greyImages	243x320x283 double
H	243x320x283 double
M_abs	243x320x283 double
M_bs	243x320x283 double
M_fd	243x320x283 double
M_pfd	243x320x283 double
theFiles	283x1 struct

Quantitative Results

All methods have user-settable thresholds and parameters. One is the pixel difference threshold for determining what level of intensity change is needed to declare there has been a significant change at a pixel, not just sensor noise.

- What value did you decided to set it at, and how did you decide? What do you observe when you set this higher or lower?
    - The value of Lambda varied between each implementation but it usually hovered around 40. When we lowered Lambda too much the threshold decreases we can observe more interference in the image frame. This is because it starts to take into account very slight motion. Oppositely, when lambda is set too high, we lose detail of what is in motion in the frame.

- An important parameter for adaptive background subtraction is the “alpha” parameter for merging new images into the background model. Try different values for this and explain what you observed as you change it higher and lower. Can you verify that setting it to 0 and 1 yields results very similar to two of the other algorithms?
    - Alpha corresponds to the “blending”. Alpha = 0 yields simple background subtraction while alpha = 1 yields frame differencing. These effects are comparable to the last two algorithms.

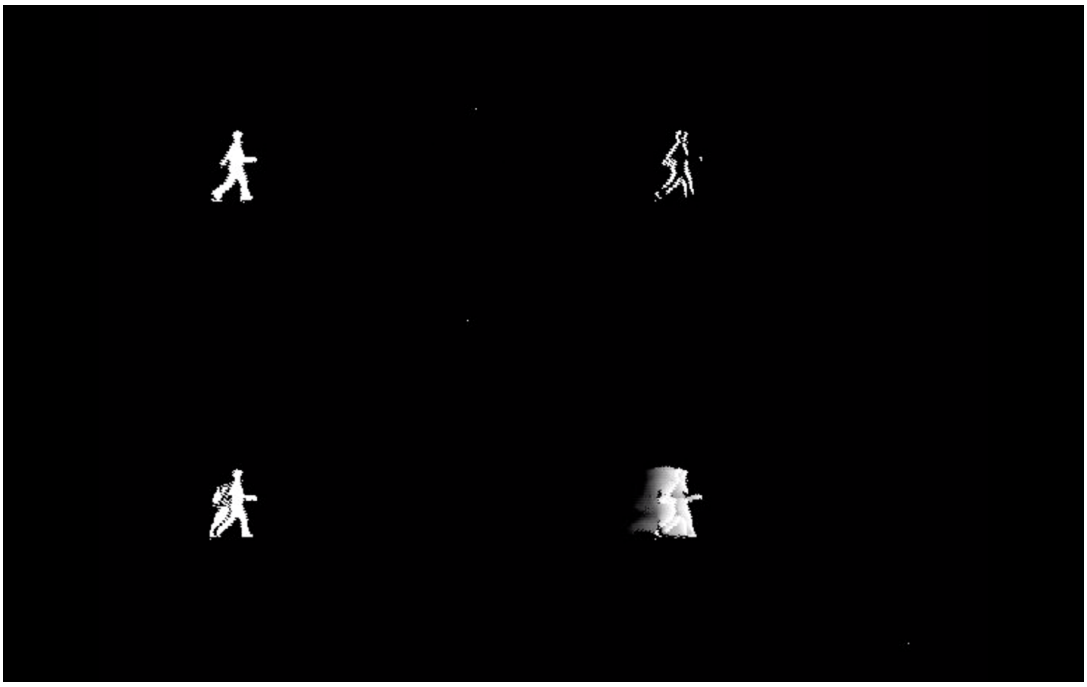
- How did you set linear decay parameter in the persistent frame differencing algorithm and what happens if you make it larger or smaller?
    - Hit and trial. If we decrease it, the shadow trail in the back increases in length. And if we increase it, vice versa happens.

Note: the function return values are arbitrary and don’t really contribute to the project.

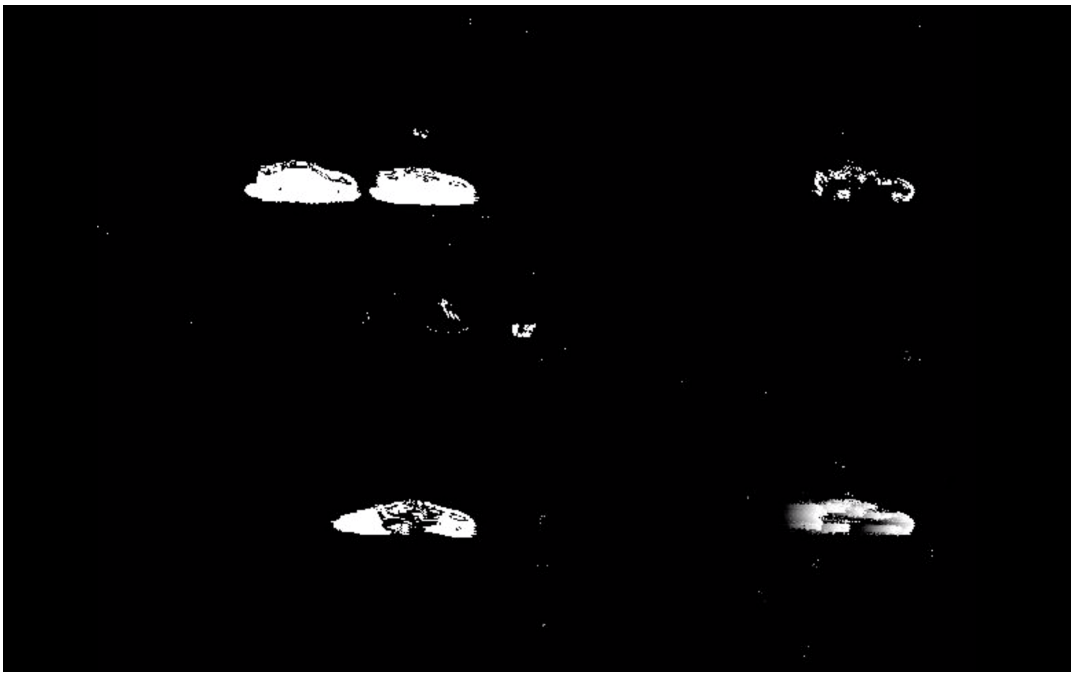
Qualitative Results

At least one quad frame showing the results from each of the 8 videos, should be included in the report in addition to the 8 results videos to be produced and included in the report:

Walk - All of them are good

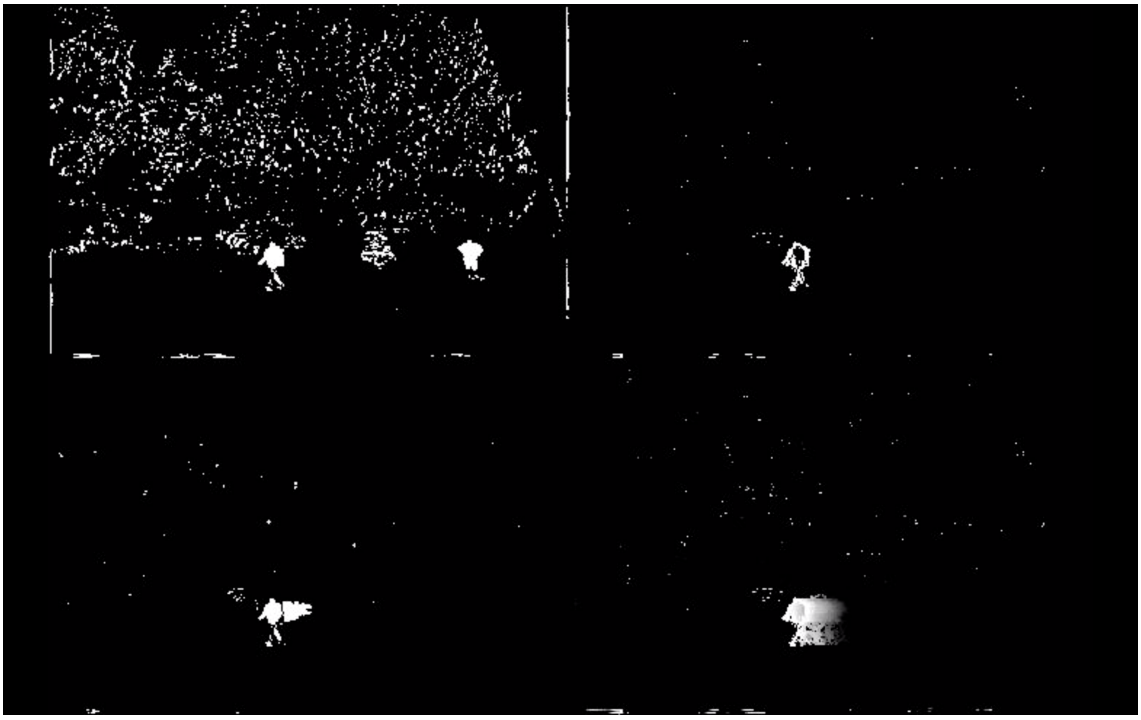


Get in - Simple frame differencing not that good

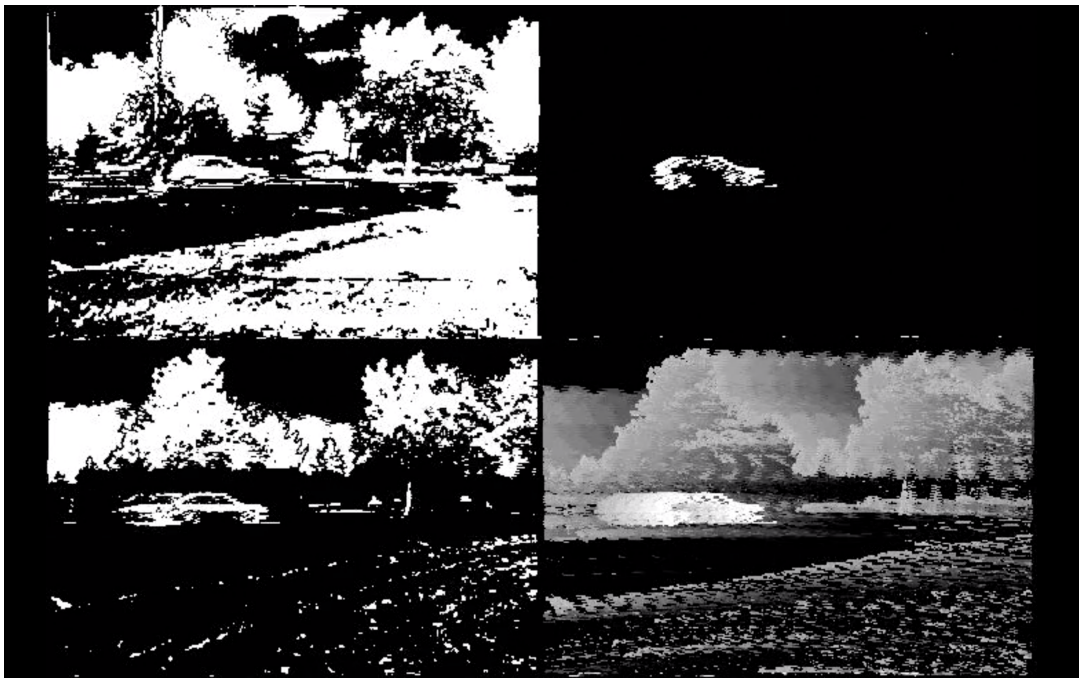


Get out - Obj small, last 2 algo are better

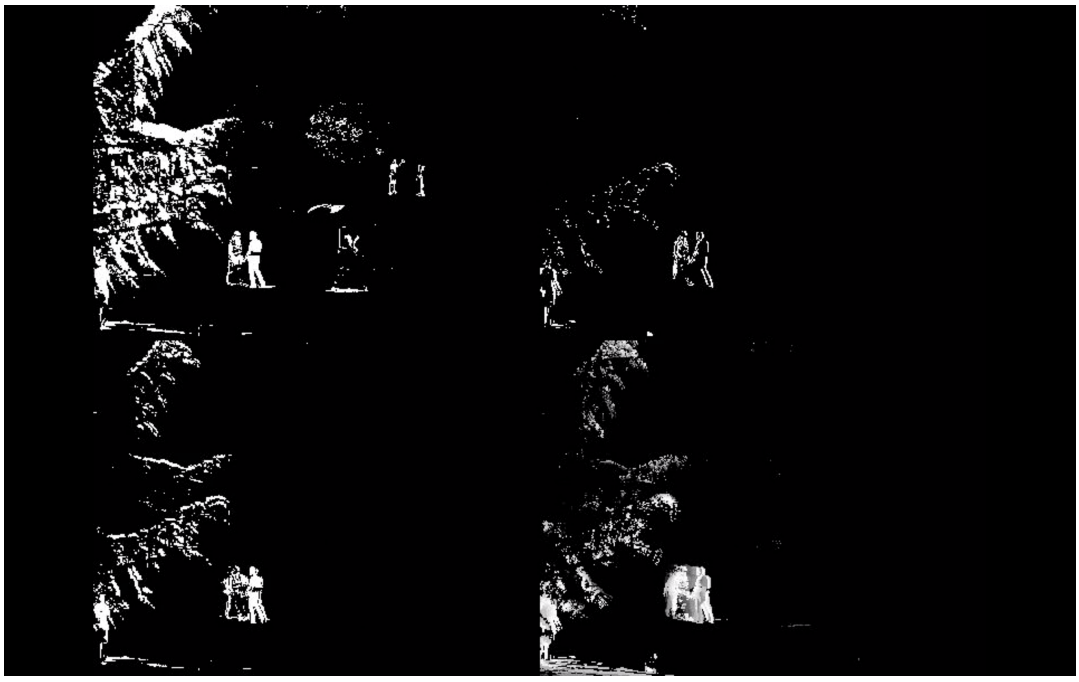
Trees - all are good



Move cam - SFD is not that good

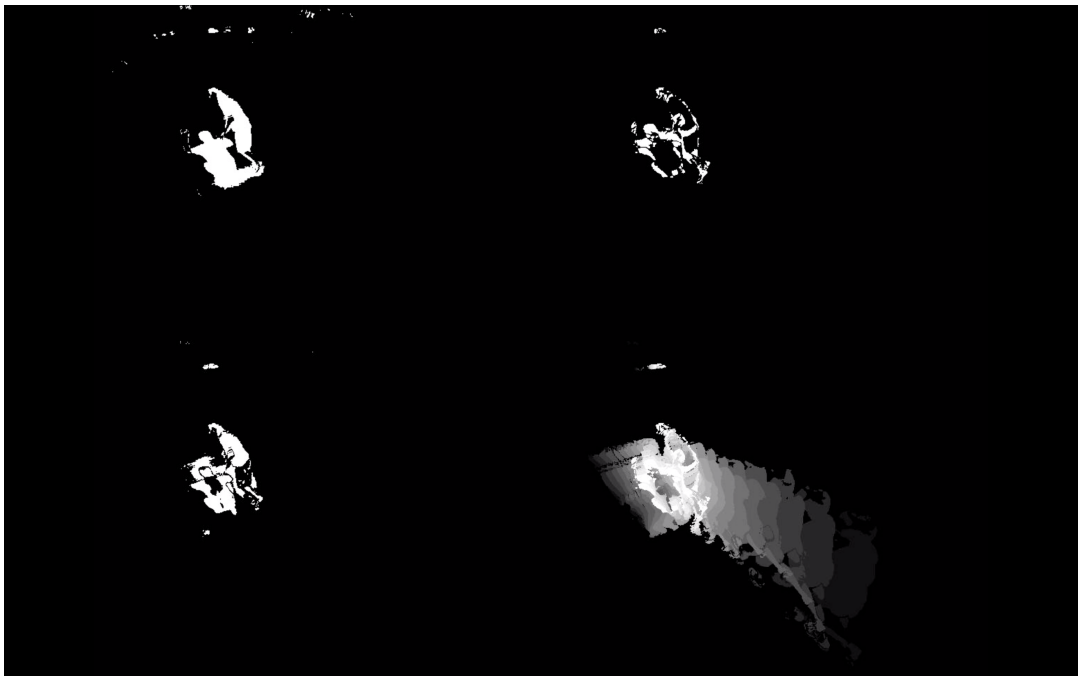


Arena A - Simple BG is the best, followed by the last 2



Arena N - all okay except SFD

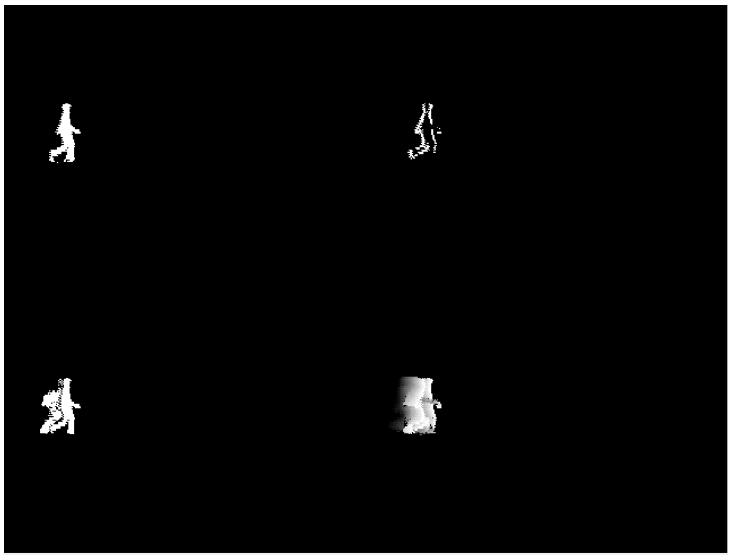
Arena W - Simple BG not as good since subjects close by



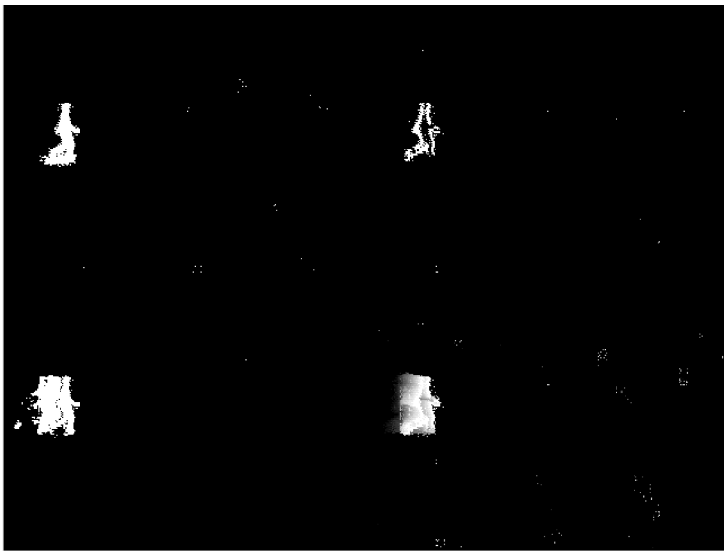
We can see that in videos in which the background changes, the Simple BG algorithm shows a different output that the others. This is because Simple BG does not store a history value but instead compares each frame to the first frame.

Additionally, the report should include intermediate images of parameter variations as discussed in the quantitative portion of the report:

L = 40, gamma = 17, alpha = 0.1



L = 20, gamma = 17, alpha = 0.1



L = 40, alpha = 0.6, gamma = 40

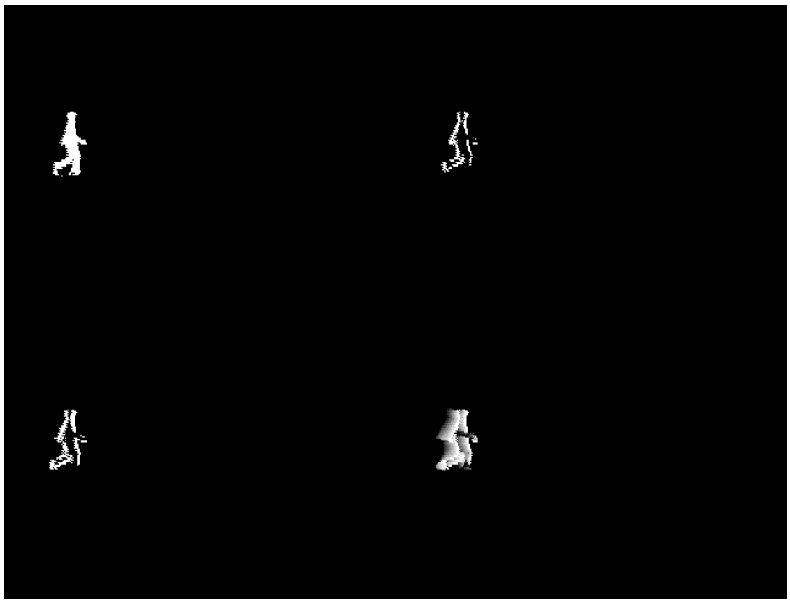


Image A (top left) is the image with the optimum parameters. As we switch Lambda (L) to a smaller value in image B (top right), the threshold decreases and, hence, we see more interference in the image frame. This is because it starts to take into account very slight motion. In Image C (bottom center), Lambda (L) is at its optimum value, but we switch the values of alpha and gamma. Increasing alpha from 0.1 to 0.6 makes the Adaptive

Background Subtraction (ABS) algorithm move closer to persistent frame differencing and, hence, we don't see the "man on fire" effect anymore. Essentially, this translates to increasing the blending. Furthermore, by increasing the value of gamma from 17 to 40, the decay value increases, hence, we move closer to having no history of the frames.

Algorithm Efficiency

- Provide images showing the MATLAB profiler output.
    - This is the profiler for the **walk function before optimization**:

Profile Summary

Generated 01-Dec-2018 04:05:07 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
LiveEditorEvaluationHelperESectionEval	1	22.589 s	0.006 s	
computeAlgos	1	22.583 s	21.393 s	
imread	283	1.057 s	0.053 s	
imread>call_format_specific_reader	283	0.897 s	0.030 s	
imagesci/private/readjpg	283	0.867 s	0.010 s	
imagesci/private/rjpg8c (MEX-file)	283	0.487 s	0.487 s	
imagesci/private/imjpginfo	283	0.370 s	0.003 s	
imjpginfo	283	0.367 s	0.166 s	
imjpgbaselineinfo	283	0.180 s	0.094 s	
imjpgbaselineinfo>recover_valid_marker	2547	0.086 s	0.086 s	
rgb2gray	283	0.072 s	0.057 s	
imread>get_full_filename	283	0.066 s	0.066 s	
fullfile	284	0.060 s	0.039 s	
fileparts	283	0.023 s	0.018 s	
rgb2gray>parse_inputs	283	0.015 s	0.015 s	
onCleanup>onCleanup.delete	283	0.015 s	0.007 s	
fullfile>ensureTrailingFilesep	284	0.011 s	0.003 s	
fullfile>refinePath	284	0.010 s	0.010 s	
imjpginfo>@()fclose(fid)	283	0.008 s	0.008 s	
fullfile>addTrailingFileSep	284	0.008 s	0.008 s	
cell.ismember	283	0.008 s	0.008 s	

```
for p = 1 : rows
        for q = 1 : columns
           if(diff(p,q) > lambda)
               output(p,q) = 1;
           end
           if(diff(p,q) <= lambda)
               output(p,q) = 0;
           end
        end
    end
```

```
output(diff <= lambda) = 0;
   output(diff > lambda) = 1;
```

- This is the profiler for the **walk function after optimization**:

Profile Summary

Generated 01-Dec-2018 04:08:17 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
LiveEditorEvaluationHelperESectionEval	1	3.060 s	0.006 s	
computeAlgos	1	3.054 s	1.904 s	
imread	283	1.038 s	0.048 s	
imread>call_format_specific_reader	283	0.901 s	0.026 s	
imagesci/private/readjpg	283	0.875 s	0.009 s	
imagesci/private/rjpg8c (MEX-file)	283	0.491 s	0.491 s	
imagesci/private/imjpginfo	283	0.375 s	0.003 s	
imjpginfo	283	0.372 s	0.199 s	
imjpgbaselineinfo	283	0.156 s	0.081 s	
imjpgbaselineinfo>recover_valid_marker	2547	0.076 s	0.076 s	
rgb2gray	283	0.061 s	0.048 s	
imread>get_full_filename	283	0.053 s	0.053 s	
fullfile	284	0.051 s	0.033 s	
fileparts	283	0.020 s	0.016 s	
rgb2gray>parse_inputs	283	0.013 s	0.013 s	
onCleanup>onCleanup.delete	283	0.013 s	0.006 s	
fullfile>ensureTrailingFilesep	284	0.010 s	0.003 s	
fullfile>refinePath	284	0.009 s	0.009 s	
cell.ismember	283	0.008 s	0.008 s	
imjpginfo>@()fclose(fid)	283	0.007 s	0.007 s	
fullfile>addTrailingFileSep	284	0.007 s	0.007 s	
imread>parse_inputs	283	0.006 s	0.006 s	
onCleanup>onCleanup.onCleanup	283	0.005 s	0.005 s	
ispc	283	0.004 s	0.004 s	

- This is the profiler for **all eight videos combined after optimization**:

Profile Summary

Generated 01-Dec-2018 04:10:04 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
main	1	54.328 s	0.006 s	
computeAlgos	8	54.322 s	38.515 s	
imread	2921	14.602 s	0.369 s	
imread>call_format_specific_reader	2921	13.451 s	0.246 s	
imagesci/private/readjpg	2921	13.206 s	0.077 s	
imagesci/private/rjpg8c (MEX-file)	2921	9.430 s	9.430 s	
imagesci/private/imjpginfo	2921	3.698 s	0.022 s	
imjpginfo	2921	3.676 s	2.154 s	
imjpgbaselineinfo	2921	1.354 s	0.749 s	
rgb2gray	2921	0.758 s	0.626 s	
imjpgbaselineinfo>recover_valid_marker	23145	0.605 s	0.605 s	
imread>get_full_filename	2921	0.506 s	0.506 s	
fullfile	2929	0.447 s	0.291 s	
fileparts	2921	0.155 s	0.119 s	
rgb2gray>parse_inputs	2921	0.131 s	0.131 s	
onCleanup>onCleanup.delete	2921	0.124 s	0.055 s	
fullfile>refinePath	2929	0.078 s	0.078 s	
fullfile>ensureTrailingFilesep	2929	0.078 s	0.023 s	
imjpginfo>@()fclose(fid)	2921	0.069 s	0.069 s	
fullfile>addTrailingFileSep	2929	0.055 s	0.055 s	
imread>parse_inputs	2921	0.050 s	0.050 s	
cell.ismember	2921	0.048 s	0.048 s	
onCleanup>onCleanup.onCleanup	2921	0.044 s	0.044 s	
ispc	2921	0.036 s	0.036 s	

Contributions

Both acknowledged members of this team (Gursahej Singh and Emmanuel F. Werr) performed equal work. This is true for the coding portion as much as the project report. The work was completed by making use of effective collaboration throughout all aspects of the project.