CMPEN/EE 454, Project 3, Spring 2020

Due Wednesday, December 9 by 11:59pm, submitted in Canvas

1 Motivation

The goal of this project is to implement the four simple motion detection algorithms described in Lecture 24 and run them on short video sequences. As described (and pseudocoded) in Lecture 24, the four algorithms are:

- Simple Background Subtraction
- Simple Frame Differencing
- Adaptive Background Subtraction
- Persistent Frame Differencing

You are to implement all four in one program and generate, as output, a four-panel frame showing the results of each algorithm, on each video frame (see Figure 1), writing the resulting images out to numbered files, i.e. out0001.jpg, out0002.jpg, etc. Note: you don't have to label the images with text as in Figure 1, but the results of the four algorithms should be displayed in the order as shown in Figure 1.

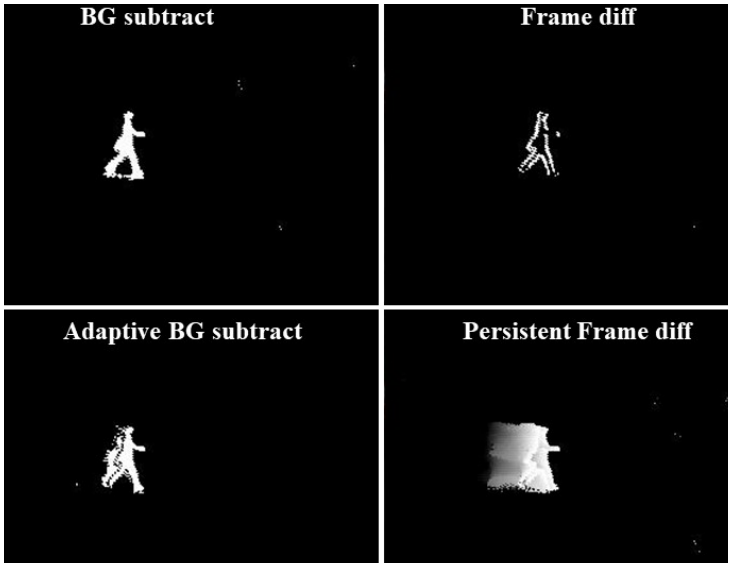

Figure 1: Example output frame showing results of the four motion detection algorithms run on the same input sequence. Simple Background Subtraction (top left), Simple Frame Differencing (top right), Adaptive Background Subtraction (bottom left), Persistent Frame Differencing (bottom right)

The specific project outcomes include:

- Experience in efficient and effective Matlab programming
- Understanding background subtraction and temporal frame differencing algorithms
- Generating videos visualizing the temporal performance of implemented algorithms
- Implementing efficient code to ensure timely operation and testing of algorithms

2 Detailed Description

Refer to lecture 24 slides for detailed descriptions of each of the four motion/change detection algorithms. The steps of your program should be roughly the following:

1) For a given video sequence, read in each image in a loop and convert it to greyscale using whatever method you are familiar with (for example by taking the green channel).
2) After reading each image, compute an output frame using each of the four motion detection algorithms and concatenate them together into a single four-panel (four quadrant) frame. Concatenation is easy to do in Matlab, since images are arrays. For example, if the four motion detection results image arrays are A, B, C and D (you will want to make them more descriptive names), the output quad image can be generated as "outimage = [A B; C D];". Caution: the persistent frame differencing image will have a different greyscale range (0-255) than the other three methods (0-1), so when generating the output quad image you want to convert it to the range 0-1 by dividing by 255 so all output brightness ranges are compatible. Alternatively, you could scale up the intensity range of the other three by multiplying by 255.
3) Generate a numbered filename for your output image (e.g. if you are currently processing frame 0036 of the input sequence your output image will be numbered 0036) and save the image, either as a JPG or PNG. If you save as a PNG, your results will have better resolution in the final movie, since it does not compress the image while writing it to a file. Continue the loop until all input images have been processed.
4) After your program finishes generating quad image results files, generate a video of those results offline. How you do this is totally up to you. One program that can take numbered images and combine them into a movie file is "ffmpeg", which is open-source software with versions that work on Windows, Linux or Mac. You may alternatively use Windows moviemaker, Mac iMovie, Matlab videoWriter, or whatever video creation tool you would like. Make your movies in some obviously playable format, like mp4.

Pro Tip: Although you are running all four algorithms at the same time, it is a good idea to keep their background image data structures independent from each other. For example, in the lecture pseudocode, variable B or B(t) is used to denote the current background frame. However, this image will be different for each algorithm, so you really should be keeping four current background frames, one computed by each algorithm.

Pro Tip 2: You don't want to be explicitly keeping indexed arrays of images around, even if that is what it looks like the pseudocode is doing. For example, when the pseudocode says B(t-1) and B(t), you really just have one background image B (per algorithm), and it is interpreted as B(t-1) when you start processing incoming frame I(t), and then after you update it you can consider it to be B(t). Similarly, with M(t) and H(t) in the various pseudocodes. The same is even true of incoming image I(t)… you can just read the image file for frame t when you need it in the loop. By not keeping all images in memory at once, you can in principle process real-time streams of frames coming in from a web camera, with no upper bound on number of frames you may eventually see.

3 Datasets

We've uploaded a DataSets.zip file containing several sequences to test on. Five of them are the test sequences that were used in Lecture 24. These are included so that you can compare your results against ours, to debug your algorithms to see that they are working correctly (results videos from Lecture 24 are available on Canvas).

Also included are three more challenging sequences, called ArenaA, ArenaN and AShipDeck. You will generate output videos of your results for each of these. These sequences are portions of publicly-available datasets originally located at https://motchallenge.net/data/PETS2017/ .

4 Evaluation

To allow us to quantitatively evaluate your code, we want you to write a main function that we can call with specific arguments and that produces output images that we will then be able to compare against results produced by our instructor code.

Write a main function "proj3main.m" that can be called as:

proj3main(dirstring, maxframenum, abs_diff_threshold. alpha_parameter, gamma_parameter)

where

dirstring = directory where numbered input image files are read from. You can assume the filename format of the images is f0001.jpg , f0002.jpg, f0003.jpg, … and so on.

maxframenum = integer which is the frame number of the last image file in the sequence (the frame number of the first image file will always be 1). For example, if maxframenum is 123, then you will be reading a sequence of files f0001.jpg to f0123.jpg

abs_diff_threshold = number between 0 and 255. It is the threshold called "lambda" in the lecture slide pseudocode, responsible for converting absolute intensity difference values into binary values 0 and 1.

alpha_parameter = floating point number between 0 and 1. It is the alpha "blending" parameter used in adaptive background subtraction algorithm.

gamma_parameter = number between 0 and 255. It is the gamma "decay" parameter used in the pseudocode for persistent frame differencing.

5 What Libraries Can I Use?

The intent is that you will implement the change detection algorithms using general processing Matlab functions (https://www.mathworks.com/help/matlab/functionlist.html) and functions in the image processing toolbox (https://www.mathworks.com/help/images/index.html).

DO NOT use anything from the computer vision toolbox, or any third-party libraries/packages. Don't plagiarize any code from anywhere or anyone else.

6 What to Submit

Half of your grade will be based on submitting a fully operational program and the other half will be based on a video report showing your results on the 3 "challenging" sequences that were given in DataSets.zip.

1) Turn in a running version of your main program **proj3main.m** along with all subroutines or helper function used in a single zip/tar archive file. Include enough comments in your functions so that we have a clear understanding of what each section of code does. The more thought and effort you put in to demonstrating / illustrating in your written report that your code works correctly, the less likely we feel the need to poke around in your code, but in case we do, make your code and comments readable.

2) Because of the shortness of time until the end of the semester, we are not going to ask you to submit a written report, but instead to upload a single MP4 VIDEO file showing the results your program achieves (for some reasonable choice of parameter settings) on the three sequences ArenaA, ArenaN and AShipDeck given in DataSets.zip. This can be done, for example, by making a zoom screen recording while you play through the results videos you produced in step 4) of section 2. Include an initial "credits" slide that contains all team member names and a short description of what each member contributed to the project.

7 Grading Criteria

Project 3 grade = (100 − Deductions) × (1− Late Penalty)

Deductions - Implementation

- Implementation of all four motion detection algorithms [20pts]: Each algorithm should be clearly defined and implemented to generate each of the four images needed to generate the four-panel results output images.
- Main routine and results [20pts]: A submitted **proj3main.m** function that works properly when we call it (see Section 4), and that produces output images sufficiently close in appearance to what we think the results should be

- Program structure and readability [10pts]: The code should have enough comments to explain the functions and procedures while being divided into small modules (function calls).

Deductions – Video Report

- Video Report file [45pts]: Uploaded MP4 file that shows results of your program on the three sequences ArenaA, ArenaN and AShipDeck given in DataSets.zip.
- Credits [5pts]: Include in the video report an initial "credits" slide documenting the names of each team member and their contribution to the project.

Late Penalty

Deduction	**Late Time**	**Time Stamp Submitted**
0%	Not Late	By 11:59**pm** 12/09/20
10%	Up to 12 hours	By 11:59**am** 12/10/20
20%	Up to 24 hours	By 11:59**pm** 12/10/20
40%	Up to 36 hours	By 11:59**am** 12/11/20
80%	Up to 48 hours	By 11:59**pm** 12/11/20
100%	More than 48 hours	After 11:59**pm** 12/11/20