Documentation on Human Detection Program

/\*

Note1: this doc will demonstrate the approach in file human\_detection\_video.py.

Details of human\_detection\_from\_image can easily be deducted after understanding human\_detection\_video.py, since the methods in use of both files are similar.

\*/

/\*

Note2: Explanation of the codes are given in the python file, in comment

\*/

# 1. Aims:

Given a video, we want to detect and calculate the number of people in each frame of that video.

# 2. Overview:

For each frame, we will do 2 things: maintain the existing people that were detected in the previous frame and do a partial search (means we only search in a small window of the frame) for detecting new people. These windows are overlapping sliding windows.

The above approach helps reduce processing time, so that the program can run in real-time.

# 3. How to search/detect people in a frame/window:

We use 2 methods to detect people: HOG pedestrian detection and Haar cascades face detector.

## HOG pedestrian detector:

### Pros:

This method is originally and best used for detecting pedestrians.

Relatively strong method.

### Cons:

Often, only standing (or walking) people those have all their bodies showed in the image can be detected.

Slow. We cannot apply this method to every frame.

## Haar cascades face detector:

### Pros:

Strong, detect faces relatively well

Lightweight, fast

### Cons:

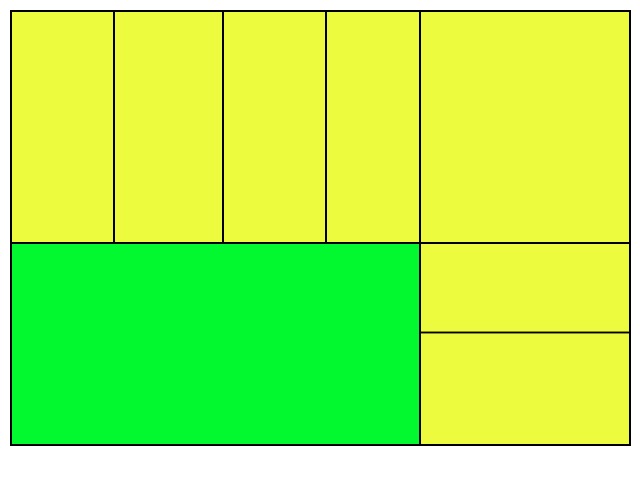
Can only detect upfront face

# 4. Define the overlapping sliding windows:

For each frame, we only try to detect people in a small window of the frame.

The sliding windows for pedestrian detection and face detection are separate (means, for each frame, we will do face detection on a window, and do pedestrian detection of an another window).

The overlapping degree is 50% (for each pair of adjacent windows, they will have 50% of their area intersect with each other).

The below figure demonstrates an example on sliding windows: The frame is depicted by green colour, the frames are depicted by yellow colour. The windows are sitting on top of the frame. There are 3 rows and 5 columns - meaning a total of 3x5 = 15 overlapping windows.

By default, in the code, I’m using 3x5 windows (similar to the above example) for face detection, and 2x7 windows for pedestrian detection.

The reason for this default setting is because of the shape of the windows. For 3x5 windows, each window will have a nearly square-shape, which is suitable for face detection. And for 2x7 windows, each window will have a standing rectangle shape, which is best for pedestrian detection.

# 5. The process:

LOOP:

Input the next frame.

Detect pedestrians and faces in the corresponding windows.

Consolidate the people detected in the last frame. [see Appendix A]

People detected in the several previous frames, if not detected in this frame, will still be counted. [see Appendix B]

For each in the detected faces, check if it belongs to one of the detected pedestrians. If it does, remove it. [see Appendix C]

Draw rectangle around each detected person.

# Appendix:

### Appendix A:

“Consolidate” means for a person that we detected in the last frame, we check to see whether this person is still in this frame anymore. This is done by considering the location and the size of this detected person in the frame.

For example, if in the last frame, a person is located at location (x, y), and have size (w, h). Then, in this frame, we perform detection in the sub-frame which located at the same location (x, y), but this frame has a bit bigger size, say (w\*1.1, h\*1.1). We will likely to detect that same person out, but the location of the person may change a bit (if he is moving).

This consolidate function is time-efficient, since by examining just a small sub-frame of the frame, we have the person we want. It can also work well for tracking task (but for now, we don’t need this).

### Appendix B:

Due to various reasons (the algorithm in use, the noise, etc.), a person may not be detected in all the frames he actually appear in.

For example, a person is detected in the 1st frame and the 3rd frame, but not in the 2nd frame. Do you think that he does or does not actually appear in the 2nd frame?, known that the time between 2 consecutive frames is just about several dozens of millisecond. Yes, if he appear in the 1st and the 3rd frame, it’s 99.99% that he does appear in the 2nd frame.

To overcome this problem, we use a memory variable. So that if this person appeared one of the last several (say, 5) frames, we assume that he also appear in this frame.

### Appendix C:

To check if 2 (faces and/or pedestrians) point to the same object, we perform an overlap-subtraction. That is, for example, if 2 pedestrians have overlapping area bigger than, say, 70% of their area, we consider them to be pointing to the same person, and remove 1 of them.