VIC – Assignment 2
Pedestrian Detection on Video

Gladys ROCH – gladys.roch@supelec.fr

General principle:

The pedestrians are detected through background subtraction and contours detection. No Machine learning is used as to not use external training dataset.

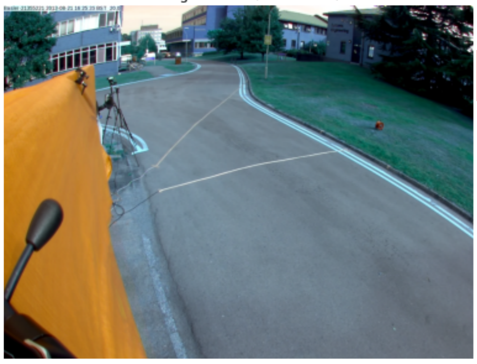
Background Removal:

Background subtraction takes advantage of the fact that the camera doesn't move throughout the video sequence. Because the background stays mostly unchanged across the frames, we can compare frames with each other and extract only the moving parts on each of them. This technique works is our case because the only moving objected are pedestrians (e.g. no cars).

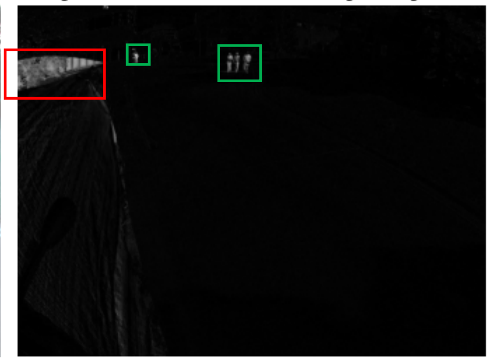
To extract the background from a frame, one option is to make the **average of all the frames** from the sequence. Because there isn't a continuous flow of pedestrians, on average any part of the image is more background than pedestrian. Therefore, the average of all frame outputs the background (see below). For computational efficiency I use only 20 random frames from the sequence, which proves to be enough.

However, the average method is very sensitive to noise in the background. In our case the yellow cover on the very left of the image flows with the wind, which causes bad background removal.

Average Frame (20 frames)



Background removal on Frame 70 using average frame



Instead of subtracting the average background, I **subtract a nearby frame from the target frame**. Because the background noise is slower than the pedestrians, the background stays constant across a few consecutive frames. Testing gave the best score using the frame at time +4.

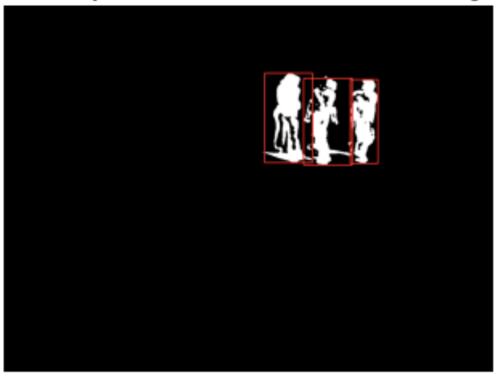
Thresholding and Contours detection:

After background subtraction, I use **Gaussian blur to smooth the delta frame** before thresholding. Testing give the best score for a gaussian kernel of size (11, 11). The threshold value to **convert the delta frame to black and white** is set to 20 (on a scale of [0, 255]).

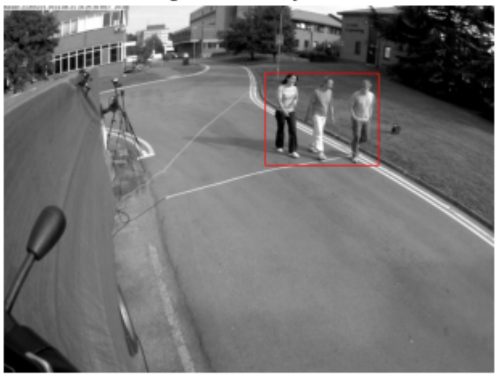
The contours detection and boxing are done with OpenCV. The boundary **boxes that are too small are removed** (the minimum length and width is 15 pixels).

The remaining boxes that are overlapping each other are **merged together**. (see bellow)

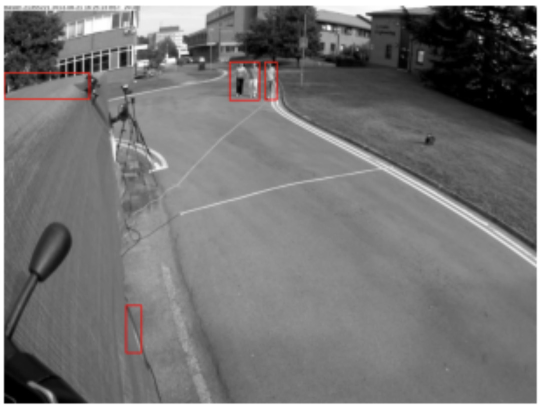
Boundary boxes on frame 340 after thresholding.



Merged boundary boxes



Failing cases:



Moving background:
When the background moves to quickly is it detected and artefact bounding boxes are added.

Shadows:
When the shadows of the pedestrians are very contrasted the bounding boxes are bigger than needed.

Non-Pedestrian moving objects:
The algorithms boxes cars, animals, etc. that moves across the frame it is not specific to pedestrians. It is specific to movement.

A method using ML:

Although, I chose to implement a method purely on image processing, another option would have been to use **Optical flow and a SCV classifier**.
Steps:

1- Find a pedestrian dataset for training. This should contain close ups of full-size pedestrians.
2- Compute the Optical flow on each instance of the dataset and train the SCV on it.
3- On the video, compute the optical flow of each frame, a high resolution is needed to take into account pedestrian that are far away from the camera and thus are small.
4- Create a window of the same ratio of the training dataset. Slide it over the target frame. At each step predicted using the trained SVC whether a pedestrian is in the window. If yes, create a box. I would use 2 different window sizes to catch small and big pedestrians.
5- If there are a lot of overlapping boxes I would either use a merge function of a heatmap function (keep the area where there are enough boxes overlapping).

Conclusion:

My algorithm yields a **score of 0.37** and runs in **38 seconds** on the 684-frame video.
The method I chose is appropriate for video detection on a calm background. It detects every moving object and therefore is not appropriate if we want to discriminate pedestrians from cars, birds, etc.