Documentation

How to

1. How to generate results after detection / NMS / tracking
   - Open Bgsubtrack.cpp
   - Uncomment one of the DET_RES, NMS_RES, TRK_RES flags depending of what kind of results you want to output
   - When launching the software, redirect the output to the file you want. Extension of the output file should be « .csv »

     Detection results :
     - Compile generate_results.cpp
     - Launch the script (the results are displayed in the console)
     - (optionnal) Uncomment TEST flag on top of the script to output a video showing ground truth, true positives and false positives. You should then modify the path in line 208 and line 217. Line 208 is the path to the output video you will create. Line 217 is the path to the dataset video used to produce the results.
     -

     Tracking results :
     - Compile generate_tracking_results.cpp
     - Launch the script (the results are displayed in the console)
     - (optionnal) Uncomment TEST flag to output a video showing ground truth, true_positives and false positives. Paths can be modified in line 702 and 711.
     - (optionnal) In BgsubTrack.cpp, uncomment VIZ flag to create a video showing tracklets with IDs.

2. How to generate results with background subtraction (MOG / frame difference)
   - Open Bgsubtrack.cpp
   - Uncomment the flag corresponding to the method you want to use (FRAME_DIFF flag or MOG flag) line 23 and 24
   - Follow part 1. to output the results you want

3. How to generate results without background subtraction (sliding windows over the whole image)
   - Open Bgsubtrack.cpp
   - Comment both FRAME_DIFF and MOG flags
   - In main(), comment from line 660 to line 719. Comment line 723 and 724. Comment line 728, uncomment line 729. Comment line 766
   - Comment « boundingLocations[i].x + » and « boundingLocations[i].y + » in line 740
   - Repeat the previous step in line 761
   - Follow part 1. to output the results you want

4. How to create an SVM model

- Split your samples in 4 groups : positive training samples, negative training samples, positive validation samples, negative validation samples.
- Compile TrainSVM
- Launch the script
- (optionnal) Modify the SVM's C coefficient, the size of the detection window and the way this window slides in line 208 of TrainSVM.cpp. You can also modify the histogram's number of bins.
- (optionnal) Modify SVM's kernel in line 12 of TrainSVM.cpp
- (optionnal) Uncomment HARD_TRAINING flag (line 6) to do hard_negative mining and retrain the SVM model with these new hard samples.

Main idea of the algorithm

1. Background subtraction

Given any image of the video we want to process, we first apply a background subtraction in order to extract foreground moving objects. We use this method because the UANDES dataset is composed of videos showing static scenes (neither the environment nor the lights are varying).
We allow to use 2 different methods to apply the background subtraction :

- by using a Mixture of Gaussians (MOG) which allows us to obtain a model of the background in the form of a probability map (each pixel is given a weight that represents its probability to belong to the background or the foreground). These weights are updated at each frame of the video. The way this background model evolves is defined by a learning rate coefficient which allows us to tune the weight's variation step.
  For example with a low learning rate, we probably will not be able to segment a moving object that happens in a regions that did not vary for hundreds of frame. The weights associated with these pixels are really low because nothing happened in this region for hundreds of frame and when something finally happens the learning rate is too low to update the pixels' weights enough to overpass the threshold segmenting foreground from background.
- by calculating the difference between two consecutive frames

Background subtraction has cons such as :

- sensitiveness to static objects of interest (non moving object will not be segmented)
- sensitiveness to pedestrian wearing clothes of the same color as the background
- sensitiveness to fast illumination changes

2. Selection of regions of interest

Background subtraction will return a binary image with multiple objects. Most of them are not of interest (noise etc). In order to reduce this noise, we first apply a closing that will reduce the size of small fragmented objects. Then we extract contours in this binary closed image in order to calculate the minimal bounding rectangle of each contour. These rectangles are used to approximate the area of each contour. Given these areas we select regions which are of interest following a size criteria (the bounding rectangle must be big enough to contain human's head and shoulders).
If this condition is validated we create a region of interest which is bigger by 20 pixels on x and y axis from top left and bottom right corners than the minimal bounding rectangle.

3. Detection

Then we apply a sliding window that will slide through each identified region of interest. Each time the sliding window is contained by the region of interest it will calculate an HOG (Histogram of Oriented Gradients) descriptor. All these descriptors will be fed to the trained classifier to decide whether the descriptor is a pedestrian or not. In our case the classifier is an SVM (Support Vector Machine) with linear kernel, but it could be any kind of classifier.
After this step we possibly have multiple detections of the same head, that is why we apply a Non Maxima Suppression (NMS) to keep the detection in which we can have the highest confidence. We defined the confidence as the distance to the hyperplane returned by the SVM for each descriptor.

4. Tracking

Once we have our detections it will be used to update existing trackers or create new ones.
Each detection will be used in 2 different ways depending on using KCF (Kernelized Correlation Filter) tracker of color histogram tracker.

KCF tracker :

Each detection is used to validate the filter's prediction. If the detection shares a common area with the predicted area then the corresponding tracker is updated according to its prediction.

Color histogram tracker :

The detection is used to calculate a color histogram which is compared with all the existing trackers. It is also used to validate location coherence. For an existing tracker we are calculating the common area between the detection and last tracked region. If the common area is sufficient enought then we calculate the distance between the detection's histogram and the last tracked region's histogram. If the distance is small enough, all the conditions are validated and we update the corresponding tracker with the new detection.

If the location coherence is not validated (KCF) or the distance between 2 histograms is too big (color histograms) then we use the last 2 tracked regions of the tracker to estimate a movement vector. Assuming the pedestrian is walking at a constant speed in a direction that remains the same as the one

calculated with the last 2 tracked regions, we define a new region of interest centered on the estimated point. In this region of interest we apply the sliding window in order to re-detect the head. With this method we should detect only one head. Even if it is really approximative this method is not penalizing tracking performances as shown by the « ID changes » mesure which is equal to 0 for the 15 videos. However it could be a problem in denser scenes. When re-detecting a single head, this detection is used to update the tracker which was missing a measure update.

In the case that the new detection is not sharing more than 10 % common area with all the already tracked regions, then the detection is assumed to be a new person and we use this detection to initialize a new tracker.

If a tracker has lost its target for more than 5 frames then we terminate it and save it if the tracklets lasted longer than 20 frames otherwise it is just deleted.