UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Master's degree in **Automatic Control and Robotics**

Short Project

Design and implementation of a technique to track moving objects based on optical flow.

Stéphane Maillot ▸ 12/12/2016

Short Project

Design and implementation of a technique to track moving objects based on optical flow.

Abstract

The optical flow is a computer vision technique that allow to detect contrasts not only with 2D gradients but also using changes in time from one frame to another. The purpose of this short project is to use this technique in order to track moving objects in an image.

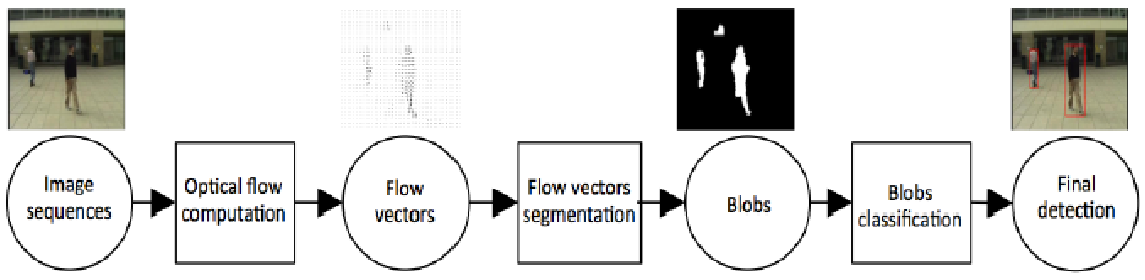
For this project, I focused on pedestrians. I thus choose to use mainly videos from street video surveillance. That's why I don't pay attention to the movement of the camera. Otherwise I should have deleted the velocity components due to the movement of the camera.

My main objective is to detect moving objects on the image and to track them while crossing the filmed area. I should also be able to estimate the number of moving object on the image. This could permit to detect abnormal reunion of people or rush ours for example. Tracking people can also permit to know about frequent itineraries.

Description of the method

The article I chose to base my project on is a thesis about pedestrian detection using optical flow. This article provides me a good starting point about the methodology to adopt in order to solve this tracking problem.

However, this thesis doesn't provide the integral code used because the writer use parts of code from a book. That's why I decided not to use the provided code and to program my own MATLAB code. Indeed, the computer vision toolbox already include a function that compute optical flow and the segmentation can be achieved by using simple morphologic operations we have seen.



Pedestrian detection method used by Núria Zoroa Rauet in his thesis.

I will then use the same method to detect moving object and will add some “tracking” steps to it. The steps are:

- Extracting frame images
- Compute optical flow from one image to another.
- Extract norm and angle from optical flow.
- Convert these data into a binary image.
- Clean and fill binary objects.
- Identify each object.
- Count objects and draw trajectories.

Explanation of the method

1. Extracting images from a video

The first step is to extract the images from video files. This is done using the `VideoReader` which can extract a frame as a 3-D length x width x 3 matrix corresponding to a RGB image.

To store the whole video I then use a 4-D matrix.

```
while hasFrame(vr)
    video = cat(4,video,readFrame(vr));
end
```

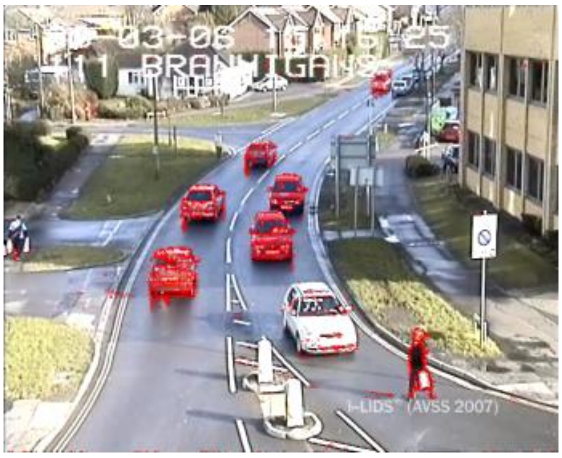
Same for saving the video, I use the `videoWrite` function.

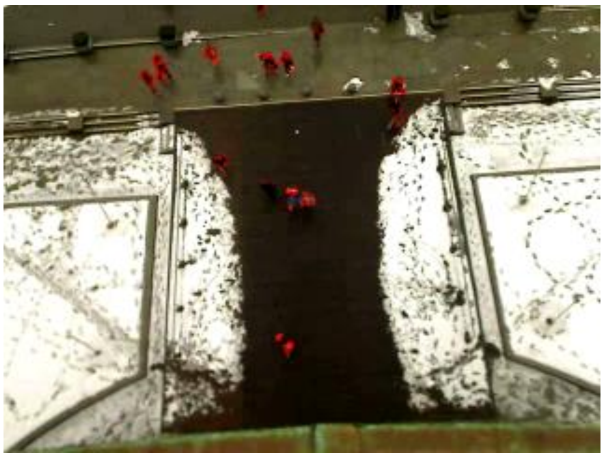
The videos used to test the algorithm is a security camera video installed in a road with cars and pedestrians and one installed on top of a hotel that film pedestrians only.

2. Compute optical flow

To compute the optical flow, I first need to convert the image in grayscale.

Then I can compute the optical flow from one image to another with the MATLAB `opticalFlowLK` object and `estimateFlow()` function. Then I have written a function to plot the vector field on the image `insertOF()`.



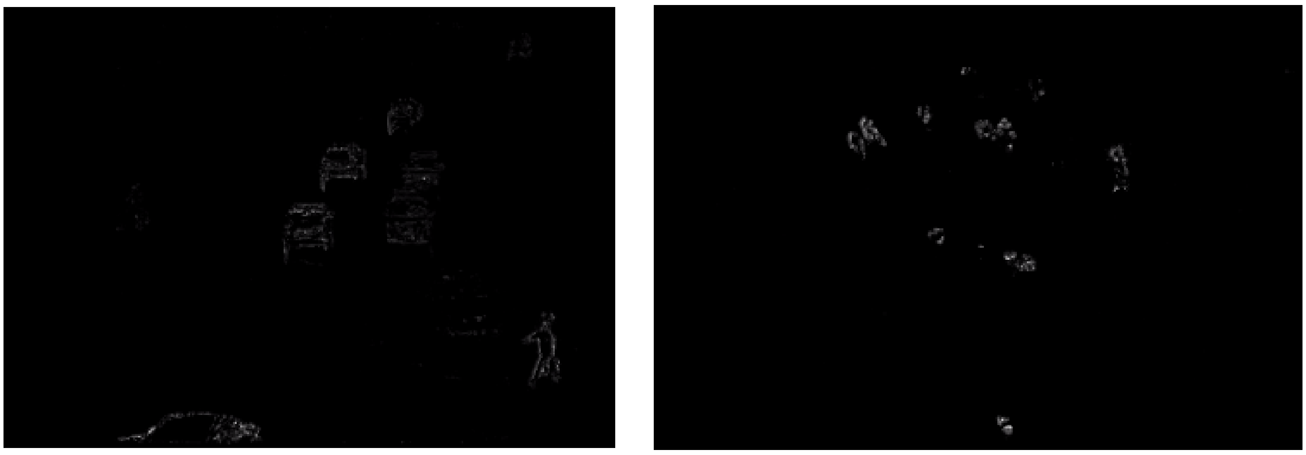


This representation is good to visualize the optical flow on the video. However, for the algorithm we need a binary image that contain information from the vector field and that permit to identify moving objects. I will then extract the norm and angle of each optical flow vector.

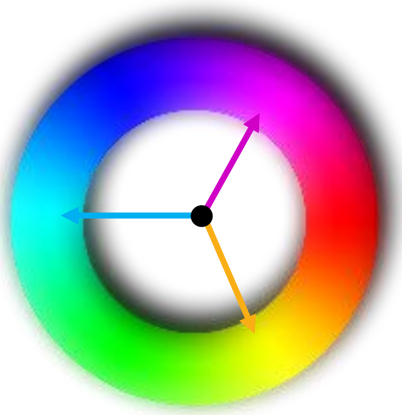
3. Extract optical flow norm and angle

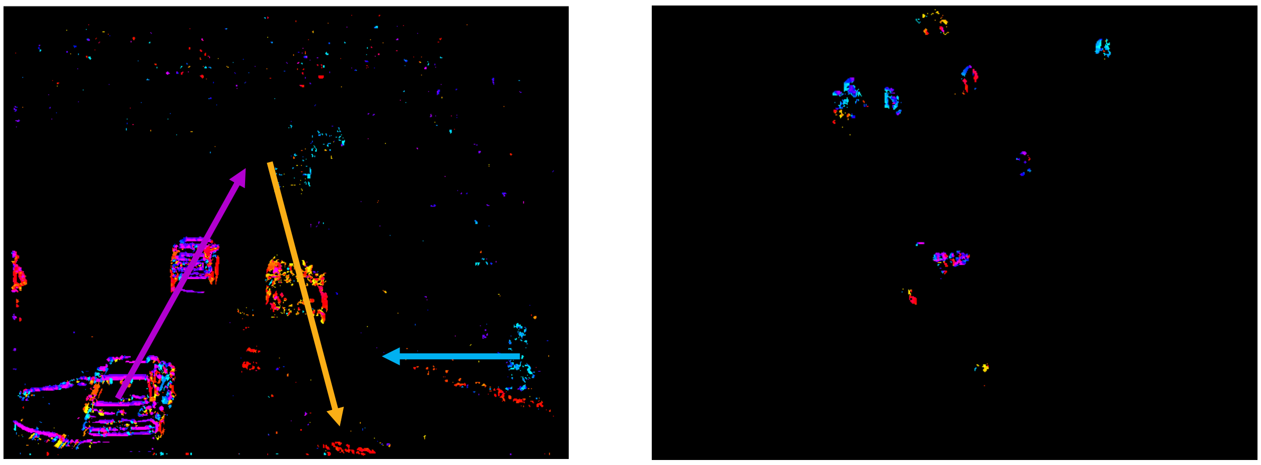
To extract this information from the image I have created 2 function `OFModMap()` and `OFModAngleMap()`.

The first one gives a grayscale image representing the norm of each optical flow vector.



The second one gives the same image but with colors representing the angle of the vector according to the following figure.



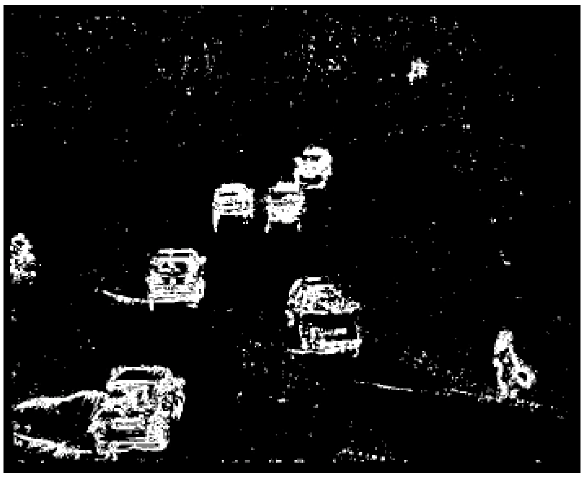
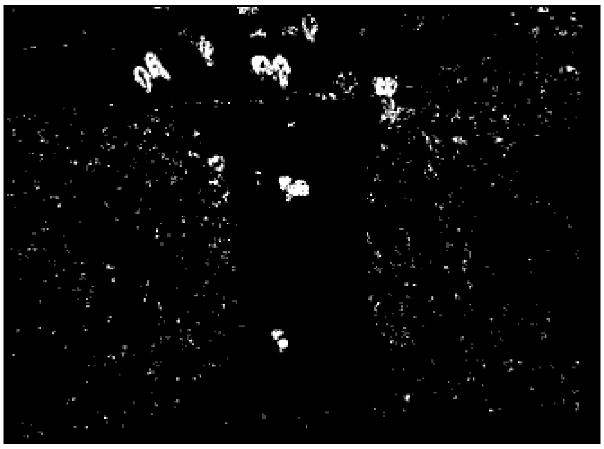


Although this is a really interesting information, I haven't used the angle in the rest of the project.

4. Convert the norm into a binary image

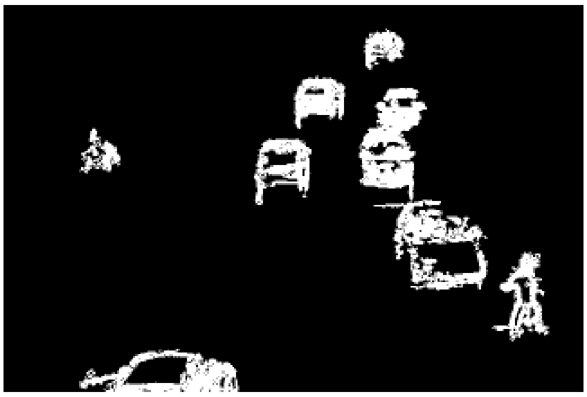
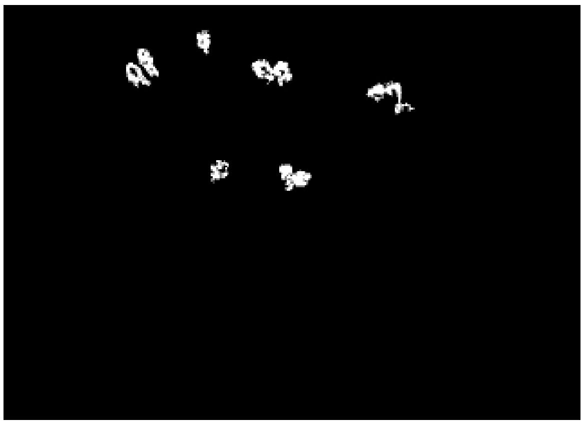
The easiest way to obtain a binary image of the scene is to use the norm of the optical flow previously computed.

To do that I used the `im2bw()` function with a given threshold. Instead of computing the "optimal" threshold as in practice 4, I realized that finding manually the best threshold is more accurate. We'll see after the next part how to find it.

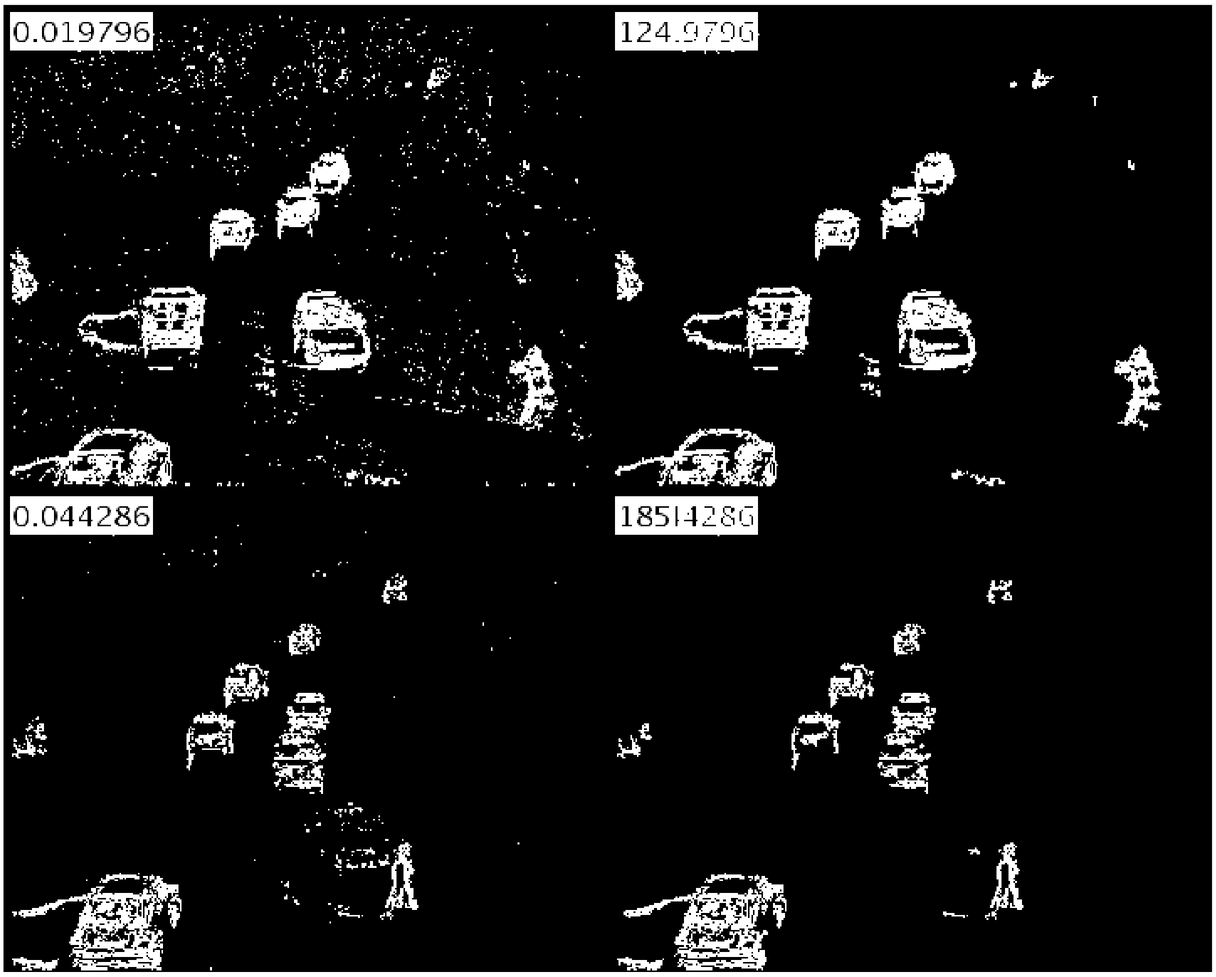
5. Cleaning the image

We can see that there is a lot of noise on these images. We now have to clean this binary image to keep only useful information. For this task, I use the MATLAB function `bwareaopen()`. This function need a cleaning size as parameter which is the maximum area to delete. We'll see below how to estimate it.

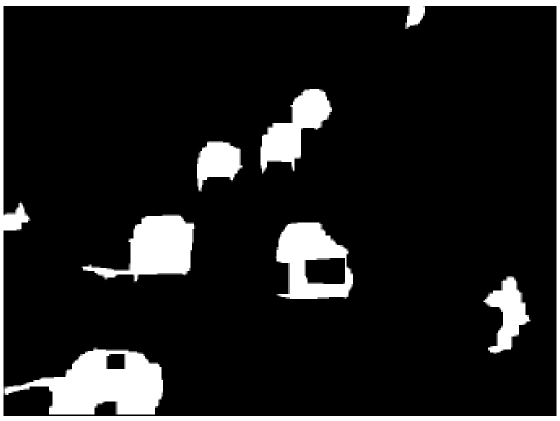
6. Estimating threshold and cleaning size

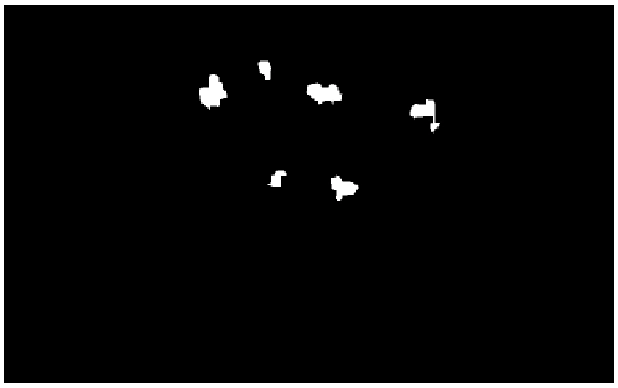
To estimate the previous parameters, I wrote the matlab file `binTester` that permit to test different parameters for binarizing the image. It creates a video of the results of binarization at different steps and make the parameters evolve to find values that work well. Here is the video at 2 different moment (2 rows), on the left the threshold and binarized image and on right the cleaning size and cleaned image.



7. Closing shapes

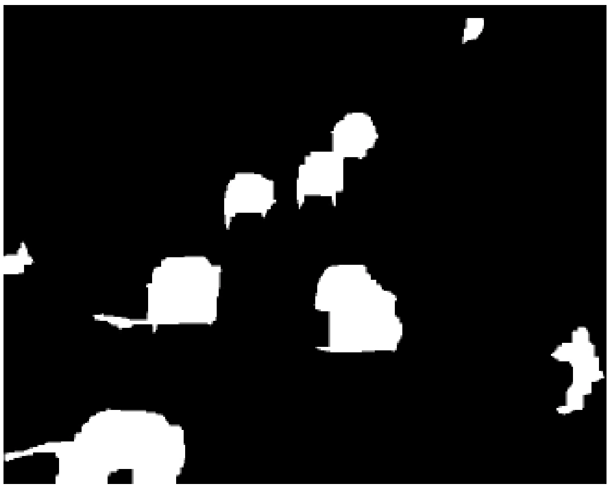
We can see that interesting shapes are now isolated. However, optical flow mainly detects contours (high contrasts). We now need to close the shapes with a morphologic operator.

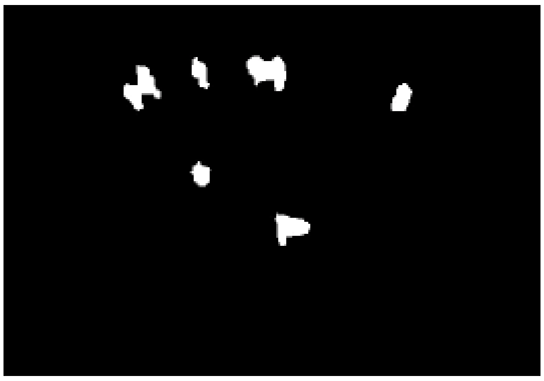




8. Filling holes

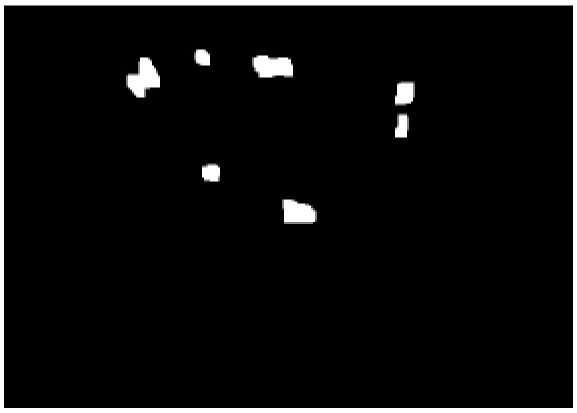
We now obtain closed shapes with some abnormalities, before eroding them a filling step could be necessary, especially for large shapes like cars. The `imfill()` function is used.





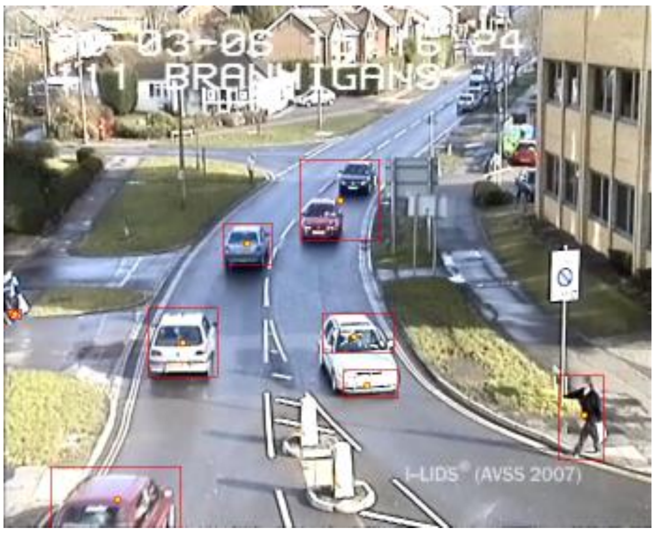
9. Deleting shadows

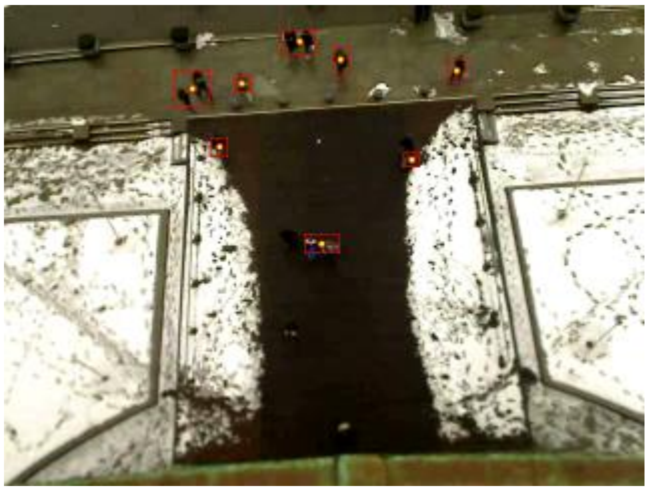
If some parts of shadows remain we can delete them with an erosion (opening if we don't want to lose the shape).



10. Detecting objects

Now we have a good separation of each shape. We can perform a blob analysis on these images. I use only centroids of each objects and bounding boxes.



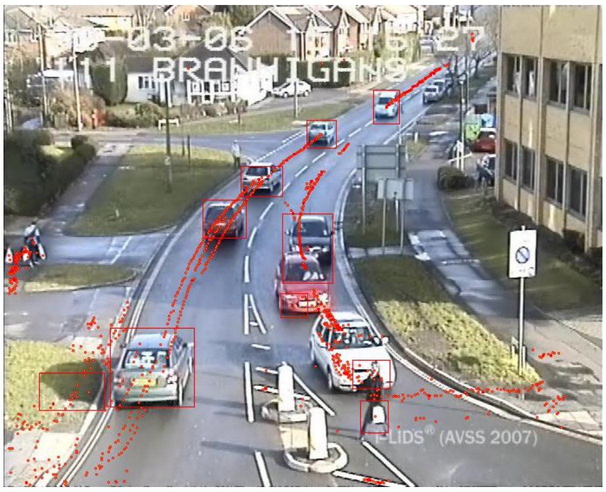


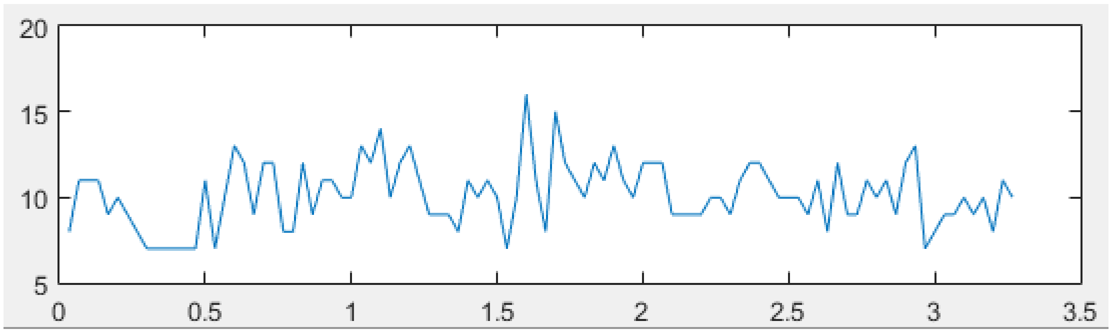
11. Tracking objects

Now we can detect the position of each object, to obtain tracking information on these moving objects I plot the trajectories to identify the flow characteristics or to identify each object itinerary.

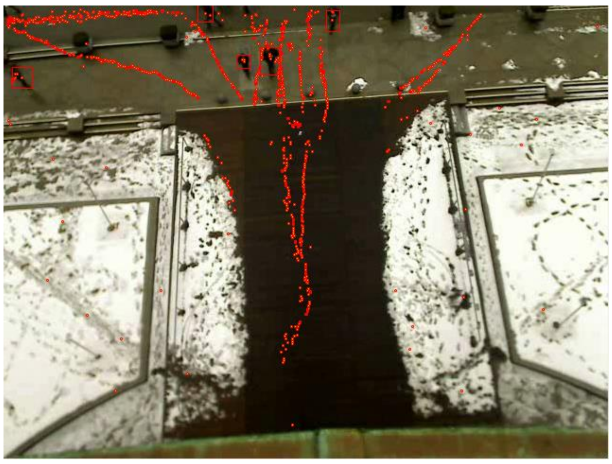
Another interesting information is the number of detected objects on the image, I plot this number over time. Even if it will probably not be accurate, it contains useful information for automatic detection of rush hours or abnormal reunions.

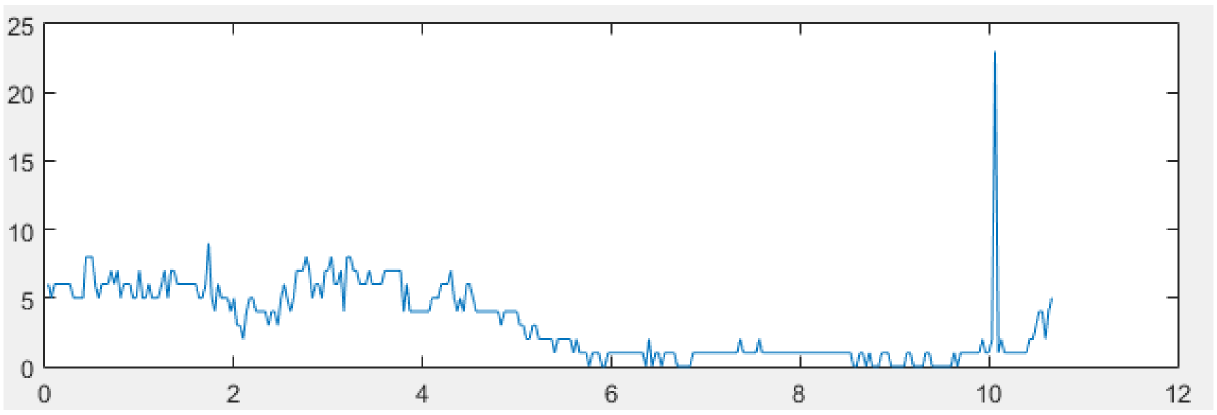
In the first example, the traffic obviously mainly follow the road and pedestrian routes. The number of moving object don't vary too much because the sequence is short (3sec).



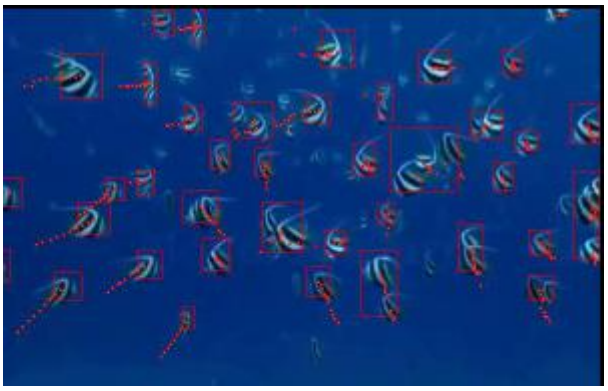


In the second example, the different trajectories are also really clear (people going left/right or crossing the road).





As the optical flow only detect changes on the luminosity, it can work in many different situations without a lot of modification in parameters. As here where it follows the trajectory of fishes.



The more important is to have moving objects with a constant size because the binarization parameters are based on the size of objects. It is also important to have separated objects, otherwise only one will be detected. A crowd movement is not well detected with this method while spread pedestrian flow is. This came from the fact that I only use the norm of the optical flow. Angle could help to distinguish close moving objects.

References

- Núria Zoroa Rauet's degree thesis –OPTICAL FLOW SEGMENTATION FOR PEDESTRIAN DETECTION

- Some videos of the dataset came from Saad Ali's personal page
- Blob analysis found on MathWorks website
- Bwareaopen on MathWorks website
- Insertshape on MathWorks website