Counting taxis with automatic tracking.

Manuel Felipe Pineda Loaiza, *MSc. Student, Universidad Tecnolgica de Pereira*

***Abstract*—This work covers the problem of detection and tracking of moving objects in a video with the specific goal of counting taxis. In this work we used computer vision techniques such background subtraction, segmentation and tracking with the help of OpenCV and python 3.**

***Keywords*—Computer vision, background subtraction, segmentation, kalman filter, OpenCV.**

I. INTRODUCTION

The video tracking is the process of locating a moving object over the time. It has a bast field of applications, some of which are [1]: video-surveillance, augmented reality, and traffic control.

The goal of the video tracking is to associate target objects in consecutive frames of the video. This is a complex task because it depends on several factors that change with the time, some of them are, for example: the trajectory of the objects and the illumination on the scene. For this reason the problem is divided in several stages that will be described later. The big picture of the process is:

1) Subtract background.
2) Image Segmentation / Classification.
3) Tracking.
4) Counting.

II. PROCESS

A. Color filter and background subtraction

In the background subtraction stage we take the original video as input and we classify each pixel as either background or foreground for further analysis (object detection and tracking). We initially used the OpenCV method `createBackgroundSubtractorMOG2` which internally uses a Mixture Of Gaussians. This worked well but we had to tune the `threshold` because the filter was very noisy and was not so stable with the illumination changes.

In order to improve that, we applied a color filter before the background subtraction task, this helped a lot with the noise in the background and it was also beneficial for the classification stage since in the video the only yellow vehicles were in fact the taxis. In the figure 2 we show the comparison between the background subtraction without the color filter and with the color filter.

To do this color filtering, we defined a color space for the "yellow" and filtered out all the pixels who where not in the color space. The figure 1 shows the result of the color filter.

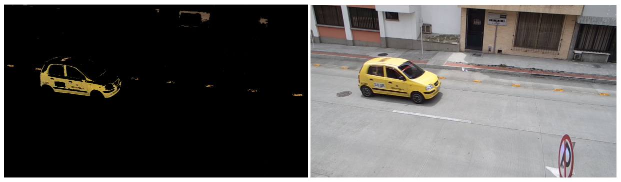
(a) Color filter. (b) Original image.

Fig. 1: Color filter applied to the original image

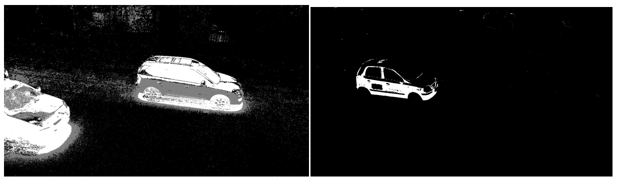
(a) Noisy background subtraction. (b) background subtraction after color filter.

Fig. 2: background subtraction comparison's

B. Segmentation

The goal of this stage is to separate the objects in the image, in our particular case, is in charge of detecting the taxis. To achieve this we found the contours in the image using the method `findContours` of OpenCV. For each image, we filtered out all the contours that were too small o too large based in the "normal" size of a car in the image. Each contour is described as a bounding box around the taxi, only the valid contours are passed to the next stage.

Usually, some machine-learning related techniques are applied at this point to improve the detection and classification of the objects in the image. However in our case that was not necessary because the color and the size were really good descriptors of the taxis.

C. Tracking

Given the contour (bounding box) of the taxi we computed the central points [2] and we used them as the position of the taxi. With the position of the taxi in each frame, we used a Kalman filter to estimate the real position of the vehicle.

The kalman filter is an optimal recursive Bayesian filter for linear functions subjected to Gaussian noise [3]. The goal of this filter is to generate precise estimations over the time. It is composed by two equations:

$$x_k = Ax_{k-1} + w_{k-1} \tag{1}$$

Where x_k is the state vector at time k, A is the transition matrix and w_{k-1} is the process noise.

$$z_k = Hx_k + v_k \tag{2}$$

Where z_k is the measurement itself, H is a matrix that maps the state to the measurement, and v_k is the measurement noise.

We modeled the Kalman filter for a 2D motion with constant speed.

$$px_k = px_{k-1} + \Delta t * vx \tag{3}$$

$$py_k = py_{k-1} + \Delta t * vy \tag{4}$$

Where px_k and py_k is the position in x, y at the time k and vx_k and vy_k are the components of the speed. Then we can define a state vector as:

$$x = \begin{bmatrix} px \\ py \\ vx \\ vy \end{bmatrix} \tag{5}$$

And the transition matrix is:

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

And finally the measure matrix is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{7}$$

For the implementation of this stage we used OpenCV with the model described above.

D. Counting

This stage is really simple and it was merged with the tracking in the code, basically when a predefined threshold is passed between two measurements of the position, we count the object as a new taxi and we reset all the parameters of the filter.

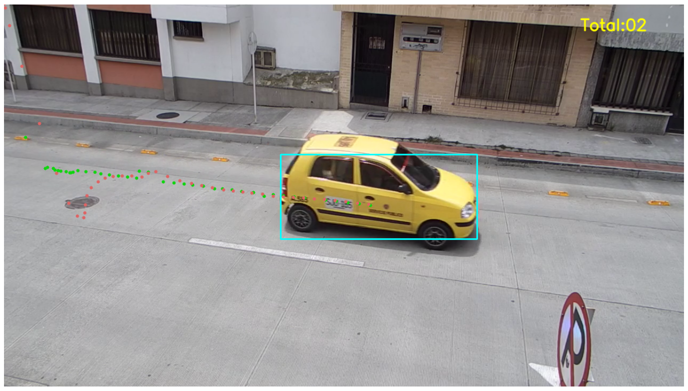

Fig. 3: Final output of the tracking and counting.

III. Experiments and Results.

All the images shown in the report were generated by our code. The test videos were taken using a Fujifilm Finepix SL1000 camera.

In the tested video, the accuracy of counting the taxis was of the 100% but we still have an unsolved problem which is the classification of yellow cars that are not taxis.

IV. Conclusions

The process of tracking and counting taxis in a video under normal conditions of illumination and traffic worked as expected.

All the tuning of the parameters was done manually, this is a good starting point for future work, merging this with more sophisticated techniques and machine learning.

The background subtraction stage is a really complicated process, in this case the color filter was a really good help but it not generalize to the process of counting any type of vehicles.

References

[1] Wikipedia. Video tracking — wikipedia, the free encyclopedia, 2017. [Online; accessed 6-November-2017].

[2] OpenCV. Contour features, 2017. [Online; accessed 6-November-2017].

[3] Wikipedia. Kalman filter — wikipedia, the free encyclopedia, 2017. [Online; accessed 7-November-2017].