# Counting and Tracking Multiple Objects: An Intelligent Transportation System Application

Cristopher Freitas

November 2, 2016

Instructor: Dr. Emma Regentova
Deparment: Electrical and Computer Engineering

**Abstract**

During this research project, we've been working with Computer Vision algorithms to create a real-time system capable of multi-object tracking and counting, for bicyclist and pedestrians. The system is implemented using OpenCV and C++ libraries for UNIX-based OS. The algorithm starts with a Background Subtraction and its following processing for reducing noise, increasing the BLOB quality, which will be detected and used as input to the tracker according to its features.

## 1 Objectives

This report is part of an Independent Study class, which the goal is to get involved with field of research, in this case, Computer Vision and Image Processing, trying to solve problems and contribute with the project. The main problem was to develop a counting and tracking system for the application of safety studies in Intelligent Transportation Systems (ITS). On this problem we handle images recorded by NDOT cameras and samples taken by the research group, comparing the performance of the application, the images taken by NDOT has a low resolution, turning this problem quite challenging. Researching through the best methods of solving this problem, we came up with a solution applying Background/Foreground Subtraction for motion based detection of the objects, for each object creating an instance of the tracker and counting.

## 2 Methodology

Researching through the best methods of solving this problem, we came up with a solution divided into three problems: **(i) Motion-Based Detection**, since we cannot have manual inputs for this system, we are detecting the objects

moving in video and using as first input; **(ii) Object Tracking**, after getting the input, we can start tracking; **(iii) and Object Detection**, this last process has to recognize the object and return if it is a bicycle, pedestrian, car or noise, this a very important problem to increase the performance and accuracy of the system.

## 2.1 Motion-Based Detection

For motion-based detection we have two very known methods that can be used: Optical Flow, which analyzes the relative motion in the image, returning a vector with the moving points; and Background/Foreground Subtraction, which gets an image and keeps subtracting the next frames to obtain a Foreground image. In our application, the Background Subtraction method shows better results, since we are not interested in specific locations of moving points, but in moving objects instead, and calculating each point of a moving object would be a waste of processing power.
The Background Subtraction method used was a Gaussian Mixture Model, which in OpenCV 3.1 was implemented as BackgroundSubtractorMOG2[4][5], this implementation contains the common parameters for this general method such as number of mixtures, frame history and threshold. We change these parameters according to the image requirements. After getting the foreground, we need to apply some pre-processing before getting the blobs, the image came up with a bit of noise that was removed with a Median Filter, creating a cleaner image, however the objects still have sparse points which needed to be filled, for this, we have used a simple Morphological Transformation for closing the objects. The last step was to remove the shadows, which are detected by the GMM and depends upon the image color distribution to make a more accurate prediction of what is shadow or actual object.
After having a good foreground image, we have used a BLOB detection method, this method was implemented into OpenCV 3.1 and takes many different parameters such as size, minimum distance between blobs, its basic implementation contains a connected components algorithm for getting the BLOBs. For each BLOB returned by the detector, we create a different bounding box, which are going to be the input for the tracking method.

## 2.2 Object Tracking

With the bounding box for each object, we can now start to track these objects, however, if we tried to track every single object, that would require a huge amount of processing power and it'd be out of our objectives, which are to analyze bicycles and pedestrian through intersections. Therefore, we create regions of interest on these intersections and at the moment one bounding box overlaps this ROI, an instance of the tracker is called. The tracking method utilized is a Kernelized Correlation Filter method[3], which according to its results is the best tracking method existing by now. We extend this method for

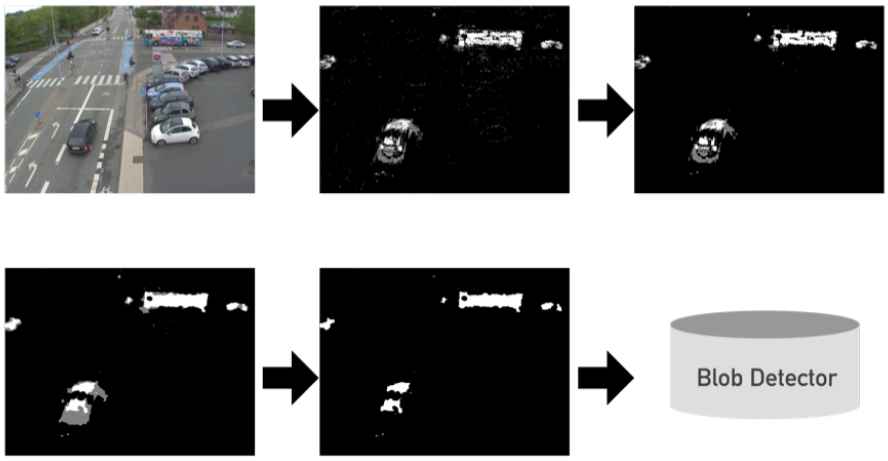


Figure 1: Background Subtraction and preprocessing Foreground result.

multi-object tracking, creating one thread for each tracker instance and counting each object.


Figure 2: Multi-Object tracking and counting sample image.

## 2.3 Object Detection

The purpose of having object detection is to improve the system accuracy and add the ability of counting the bicycles and pedestrians separated. The tracker used, when loses its object, is not able to recover after this and doesn't destroy the tracker instance, which can clutter the system with a lot of objects that are not valid anymore. Hence, the object detection can help the

tracker to analyze the object confidence and destroy if the tracker gets lost. Until now, we've worked with three different methods of object detection, which are Haar-like, HOG and the DPM method[2][1], which is a hybrid implementation of Haar and HOG. The DPM is very accurate to detecting object without any previous input, but we don't need this concern, since we already have the object, we can use Haar or HOG to verify its confidence and take any further decision. Testing the DPM detection had a huge impact in the frame rate of the system, then we are now working with Haar-like features to be implemented.

### 2.4 Results and Conclusion

As results, we analyzed the output videos generated by the systems, lacking of some statistical analysis yet. But as we can see in the videos samples available at this repository, there's two different videos, first one (denmark1-paths.avi), with lower resolution though with a more clear image and more organized scenario, where bicycles and cars are moving without a lot of disturbance, this scenario is a very good one for our application, as shown in the results videos, we achieved a good performance ( 17 fps running in MBPr 13' i5 2015) of getting the object (bicycle or person) tracked. However, the difficult increases when we decide to differentiate bicycles and pedestrians, and to develop a new method, it requires a lot of time that wasn't available for our work. This work intended to build an application with the best existing techiniques of tracking and detecting objects, due to limitations, we didn't exhaust testing scenarios at this application.

## References


[1] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE, 2010.

[2] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[3] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2015.

[4] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[5] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.