Moving Pedestrian and Vehicle Detection and Tracking

Kiarash Torkian

***Abstract*—This paper focuses on highlighting two algorithms for pedestrian and vehicle detection and tracking, its constraints, and how it could be improved upon in order to make the algorithms work in a lot more scenarios. The aspects explored include what makes a tracking and detection software successful at what it does, how could it be achieved, and how could the results be used in our society. Upon completion of the assignment it was found that the process can be difficult because of the huge number of the variables at play and that to create a perfect algorithm, it would takes years of research.**

I. Introduction

Automation has become an important part of our daily lives ranging from processes in factories to aircraft. Another important automation is visual recognition that is widely used in street surveillance or self-driving cars. One that is capable of detecting its surroundings and matching it with an object[1]. Although you can take those data and perform various routes, this paper will only focus on the detection and tracking part of a visual recognition software. Further, the process of pedestrian and vehicle detection and tracking can be considered as identifying objects that are not part of an environment and continuing to track their position as they move from a place to another. Creating a pedestrian and vehicle detection and tracking is a difficult process, and making one that is just perfect every time is exponentially harder. There are various variables at play, including light, angle of camera, and quality. So it would be difficult to have an algorithms that fits them all. During this project, various algorithms were tried out in order for the results to be compared. Using different benchmarks, it was found out that the two algorithms, as expected, perform differently where one was more appropriate than the other, while on other results, it was vice versa. The two algorithms that were implemented are "Absolute Difference" and "Background Subtractor."

A. Absolute Difference

The first algorithm that will be covered here is Absolute Difference. The method was implemented to experiment with detecting and tracking moving objects without the use of prebuilt tools such as Histograms of Oriented Gradients (HOG)[2][5] The algorithm starts by asking the user for a video input[3] which will then be traversed through frame by frame. This will allow it to learn the environment and differentiate between moving and static objects. By the end of the traversal, the method will create an image[3] that includes only the background and not any of the moving objects.

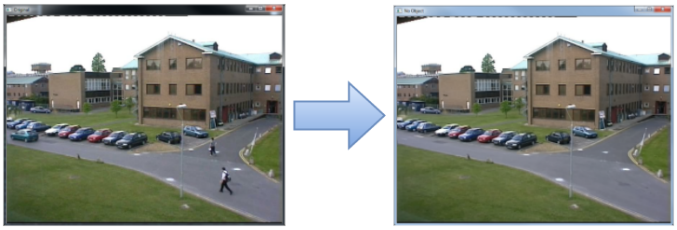

Figure 1: *A video with vehicles and pedestrians is converted into an image containing only non moving objects.*

Once the environment image is created, the method will once again loop though the video. This time finding the absolute difference between the current frame from the static environment. This will allow us to find objects that are not part of the static environment. However, the image will be very noise, therefore, to optimize, applying effects such blur is almost necessary. It will smooth out the foreground mask and also helps in avoiding the detection of the smallest movements since the small ones will disappear. The detection part is done using the findContours() method provided by OpenCV[1] which will then outline those objects. Since each frame will be detected using these exact steps, a tracking feel is given as well.

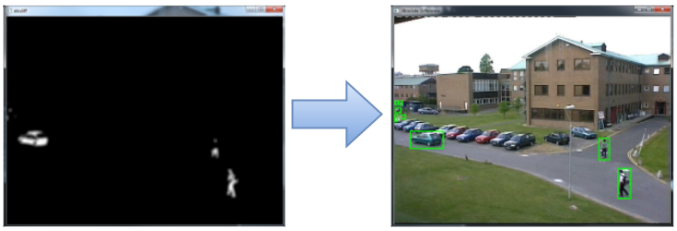

Figure 2: *Foreground mask is found using the absdiff() and then detection and tracking is done frame by frame using findContours() and threshold()[2].*

B. Background Subtractor

The second algorithm that was looked into was the Background Subtractor. This part mainly makes use of

[1]For the purpose of this paper, objects include vehicles and pedestrians

[2]HOG is a tool used in computer vision for the purpose of object detection

[3]For the duration of this paper, this image will be refereed as "static environment"

the BackgroundSubtractorMOG2 class that specialized into creating a background model. It can be used if for any reason we are unable to find a static environment. A lot of the methods used here by the class are done in the background and are not fully visible. For the most part, a developer will just have to play around with some of its constructor parameters such as history, backgroundratio, and detectShadows. For example, according to the OpenCV documents, the history field is defined as "Sets the number of last frames that affect the background model."[7] So although a shortcut that allows us to get to the foreground mask, to get a good results, one needs to play around with its parameters and possibly change them from one video to another.

Using the BackgroundSubtractorMOG2 class we are able to get the foreground mask without the need of getting a static environment first. But to optimize our results the use of blurring is still required.

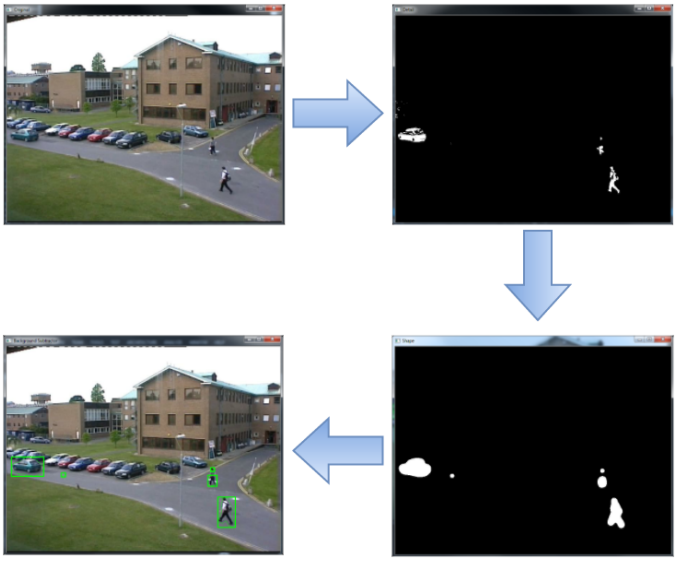

Figure 3: *A simple four steps of achieving detection/tracking using Background Subtractor.*

II. Conclusion

The program overall does a great job of detecting big movements and tracking them efficiently and accurately. Obviously there is always room for more improvements and that the algorithms are far from a perfect real life example. Couple of issues that rises in either Absolute difference(AB), Background Subtractor(BS) or both:

1) If two objects overlay, they are counted as one. (BOTH)
2) If an object has the same color as the background, its not detected. (BOTH)
3) False positives (AB)
4) Can not differentiate between a pedestrian and a vehicle. (BOTH)

References

[1] "Creating Bounding boxes and circles for contours OpenCV 2.4.12.0 documentation", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/bounding_rects_circles/bounding_rects_circles.html

[2] "Basic Thresholding Operations OpenCV 2.4.12.0 documentation", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html

[3] "Reading and Writing Images and Video OpenCV 2.4.12.0 documentation", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html

[4] "People Detection Sample from OpenCV", Magicandlove.com, 2016. [Online]. Available: http://www.magicandlove.com/blog/2011/12/04/people-detection-sample-from-opencv/

[5] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", Lear, 2016. [Online]. Available: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[6] M. Nishida, "mmnishid/MotionTrackingOpenCV", GitHub, 2015. [Online]. Available: https://github.com/mmnishid/MotionTrackingOpenCV

[7] "OpenCV: cv::BackgroundSubtractorMOG2 Class Reference", Docs.opencv.org, 2016. [Online]. Available: http://docs.opencv.org/3.1.0/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html#a5e8b40fef89a582ce42d99d2453db67a