Method for Counting Traffic Participants on Highways and at Intersections

Venkat Sai Charan Bandi

Introduction

With the rapid development in driverless and assisted driving technologies, automated vehicles are improving to have more functionality within computer vision, detecting other vehicles of multiple types in addition to pedestrians. Advances in computer vision capabilities have many potential safety upsides. To reach potential safety improvements, vehicle sensors must be able to detect when a threat is present as well as classify the threat. In order to classify the threat, it is also useful to be able to discriminate between different types of road objects (i.e. cars, busses, bikes, and pedestrians). Having the ability to do such object detection can lead to life-saving safety improvements in new cars.

In this experiment, we used four classifiers to detect and classify the different vehicles and pedestrians at highways and intersections during day, night and multiple weather scenarios including rain and snow. We annotated the frames manually and ran our models using OpenCV to check for the accuracy of our method. We ran our code on both a set of images and a set of videos.

Experimental Methods

In this project, we are using classifiers for cars, pedestrians, two-wheeled vehicles, and busses that were premade by *Aditya Pai* [1]. These classifiers were made through using tools provided by opencv. To begin training a classifier, the user must give annotated positive and negative images to opencv for learning. The positive images contain the object we are trying to learn to detect, and the negative images do *not* contain the object we want to learn to detect. Instead, negative images contain background images instead or similar objects that do not fit the class in order to learn how to determine whether an object fits the supposed class. Opencv then produces an xml file after training a number of iterations (just enough to learn the data, but not too many to overfit to the training data), which is used to create the cascade classifier that looks for the objects in our video or image.

All data was taken as either the screenshots or the raw videos downloaded from two different sites: *videvo* and *videezy*. The links to each video are provided in our citations. In general, videos were gathered to represent different environmental conditions so that we could use each set of results to see how the performance of our classifier changes based on different conditions. The various environmental conditions included night/day as well as clear weather / rain / snow. We also realized the camera angle makes a difference depending on how the classifier is trained, too, which is why two different angles were used for comparisons of the independent variables.

To implement road object counting for images with opencv, the pre-trained cascade classifiers were read in and converted to Cascade Classifiers, which were applied to each image we wanted to classify. We annotated each image to count the number of each road object (car, bus, bike, pedestrian), and then compared those findings to the number of objects that were classified by the classifier within each type of road object. Another pre-trained classifier, from cvlib, was also used for comparison on our images. This classifier is part of the cvlib library and has the capability to detect many common objects, including road objects. The hope here was to have a comparison of the widely known tool to the individual classifiers for each object that are trained with opencv.

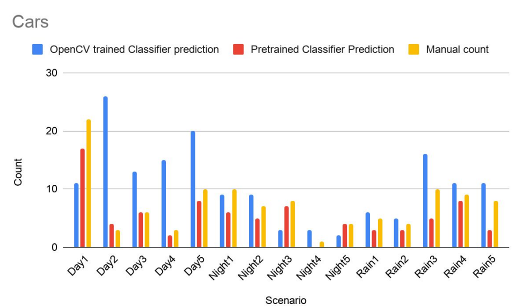
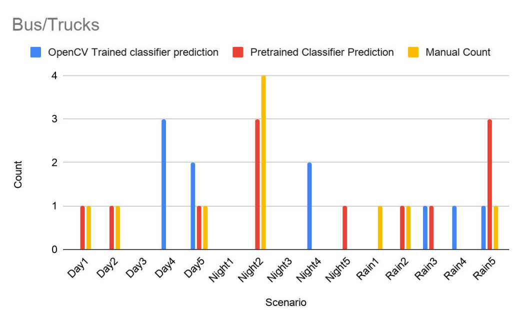
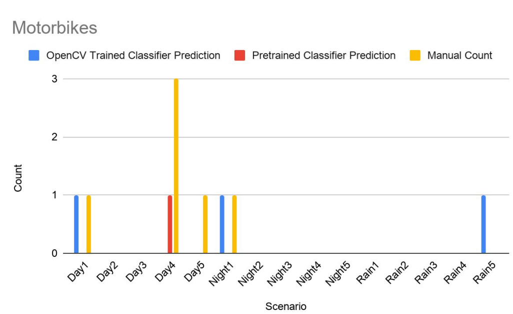
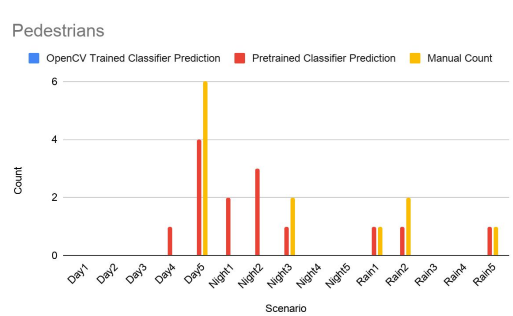
To implement road object counting for video, the pre-trained cascade classifiers were read in (*cars.xml, pedestrian.xml, two_wheeler.xml,* and *bus_front.xml*) and converted into Cascade Classifiers, just as in the image analysis. Next, a Centroid Tracker and a dictionary to hold the unique objects found was initialized for each class of object (4 total of each). Then, in a while loop which loops for each frame in our videos, the current frame was converted to black and white for faster computations by the Cascade Classifiers. New objects were detected on the first frame, as well as every 6 frames rather than every frame due to the performance cost that detecting in every frame would have. If objects weren't being detected, their corresponding trackers were being updated. After all of the corresponding trackers were updated, the centroid tracker for each class of objects was updated in order to associate old object centroids with the newly computed centroids. Each unique object's centroid is shown with a colored circle: green for vehicle, blue for pedestrian, red for bicycle, and purple for bus. This process continues as the frames are looped through, and once the last frame is reached the total number of dictionary entries for each class is returned as the count of objects for that given class.

To measure success of this method, a small sampling of videos were chosen to be annotated by hand for the number of cars, pedestrians, bikes, and busses they contained. After the ground truth was found, the number of objects counted by the program was compared. Due to the fact that we expected the classifiers to miss some objects as well as wrongly classify others, we also took count of the number of successful labels by the classifiers within each class.

Experimental Results

Part I: Image Link to the results : https://bit.ly/2J2Z1zz

The test results are based on the classifier we have trained, using a pre-trained classifier and manually counting the traffic participants in various images.

Cars : Pre trained classifier predictions have given much closer results to the actual predictions during day time while there is a considerable variation in the results during night time and during rain as well.	
Bus/Trucks : Pre-trained classifier has almost consistently predicted the number of large vehicles to the actual count during day,night and rain. The trained classifier seemed to give a larger number of false positives and false negatives throughout the experiment.	
Motorbikes : Trained classifier predictions have performed much better than pre-trained classifiers in all scenarios and gave the prediction regarding the number of motorbikes	
Pedestrians: The trained file classifiers have performed rather poorly in detecting the number of pedestrians while the pre-trained classifier, in spite of having various false positives performed better.	

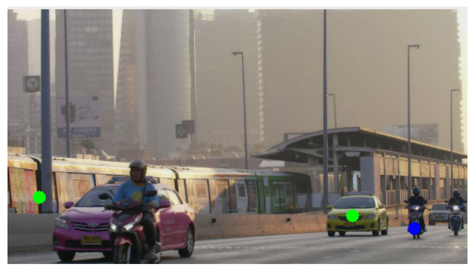
Part II: Video

The following table summarizes each video's angle, time of day, weather conditions, ground-truth counts as annotated by us, the counts from our computer vision program, as well as the number of objects that our computer vision program detected correctly.

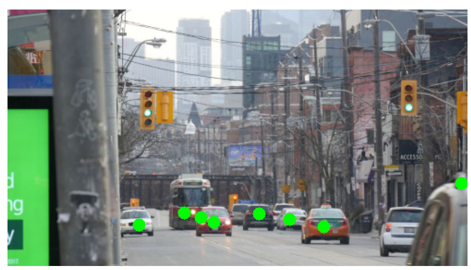
Perspective	Time of Day	Conditions	Filename	Actual Counts				CV Counts				# Correct CV Detections			
				Cars	Pedestrians	Bikes	Bus	Cars	Pedestrians	Bikes	Bus	Cars	Pedestrians	Bikes	Bus
Front	Day	Clear	front-day-3.mp4	7	0	24	1	23	14	0	1	6	9	0	1
		Snow	front-snow-2.mp4	16	2	0	1	28	9	0	0	11	0	NA	0
	Night	Clear	front-night.mp4	16	2	0	1	67	2	0	2	12	0	NA	1
		Rain	front-night-rain.mp4	14	0	0	3	1	2	0	0	0	0	NA	0
Side	Day	Clear	side-day-2.mp4	24	7	1	3	74	27	0	0	20	6	0	0
		Rain	side-rain-2.mp4	11	0	0	3	12	0	0	0	4	NA	NA	0
	Night	Clear	side-night-4.mp4	7	7	0	1	2	11	0	0	1	7	NA	0

We noticed from the results and videos is that while the tracking worked when the view of the object was uninterrupted, as soon as another object passed in front of it and then cleared the way, that same object was marked as a new object. This, combined with general false positives of other objects, is why most of the counts achieved by our computer vision code are greater than the actual count, often by a solid number. Many objects are properly identified, but over-counted, which is why the last section, *number of correct cv detections,* is useful.

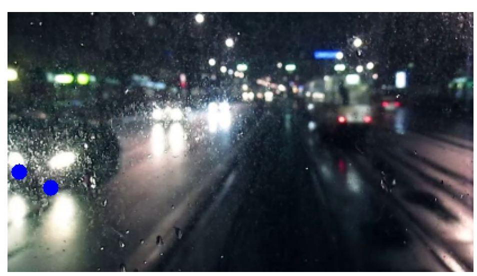
Front-Daytime-Clear: This video sample, from a highway, had a large number of motorcyclists (24) seen from the front, as well as several cars and one bus. Interestingly, the pedestrian classifier caught and identified several of the motorcyclists, which makes sense because there are humans (i.e. pedestrians) on the motorcycles. The two-wheeled classifier did not catch any of these bikes, however, likely because of the front facing perspective of this video and the fact that the classifier was likely trained on side facing motorcycles.



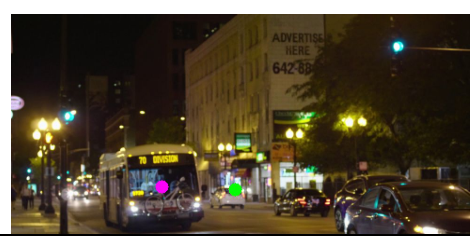
Front-Daytime-Snow: This video sample, from an urban area, had majority cars to be detected, as well as a couple pedestrians very small in the distance, and a bus. The interesting things to notice here were that the bus was detected, but as a car and not a bus for some reason. The "pedestrians" that the pedestrian classifier noted were actually the traffic light -- an interesting mis-classification.



Front-Night-Clear: This video is taken from another urban area. The interesting part of this run was that the pedestrian classifier actually classified the people on the bus, but not in the distance on the sidewalk.



Front-Night-Rain: The classifiers actually did not detect anything correctly in this video. A car and two pedestrians were detected, but they were complete misclassifications. This also likely has to do with the general quality of this video, since rain is physically on the camera lens.



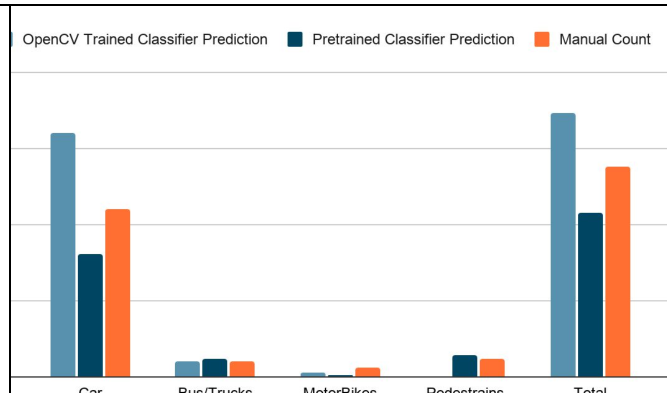
Side-Day-Clear: This video was taken in an urban setting, from a mostly side view of roadway participants. Here, in the clear conditions, the classifier did a great job of detecting cars from far away (from a front view as they waited for the through traffic to pass and their light to change). None of the busses were detected, since the classifier for busses was trained on the bus front sides. Only one pedestrian was detected until they began crossing the road and got some more sunlight.	
Side-Day-Rain: While there were multiple busses present in this video, none of them were detected again, as the bus classifier was trained only on the fronts of busses. Not too many cars were detected either, contrasting with the clear weather where cars were able to be easily detected from the sides.	
***Side-Night-Clear*:** In this urban image, the pedestrians were for the most part still, and were all able to be detected by the pedestrian tracker. Neither the cars, nor the bus here were able to be tracked by the car or bus classifiers.	

Research Questions

Part I: Images

The following graph speaks to the *counts* of the detections made by our computer vision program for images. We have cumulated the data from the experimental results in the previous section. Between the OpenCV trained classifier prediction and pre-trained classifier predictions, the latter gives better results across the weather and time.

After calculating the results we found that out of all identifications of the images we found that the openCV common objects could successfully identify all objects but had an mislabel or missed identification for about 21.74% of the data. For the OpenCV trained classifiers we had a high rate of error around 60.19%.



Part II: Video

The following table speaks to the precision of the detections made by our computer vision program for video. The calculations were made from using information from the table in the previous section, and dividing the number of objects that are correctly identified in the video at any point, by the total number of each type of object that is identified in the video. This table of calculations allowed us to make a judgment on the success of these classifiers and this method of counting.

Perspective	Time of Day	Conditions	Filename	Precision - Relevant Items Selected			
				Cars	Pedestrians	Bikes	Bus
Front	Day	Clear	front-day-3.mp4	0.26086957	0.64285714	0	1
		Snow	front-snow-2.mp4	0.39285714	0	NA	NA
	Night	Clear	front-night.mp4	0.17910448	0	NA	0.5
		Rain	front-night-rain.mp4	0	0	NA	NA
Side	Day	Clear	side-day-2.mp4	0.27027027	0.22222222	0	NA
		Rain	side-rain-2.mp4	0.33333333	NA	NA	NA
	Night	Clear	side-night-4.mp4	0.5	0.63636364	NA	NA

The percentages in this table can be read as, “of all the cars detected, 26% were properly selected”. The selections outside of those 26 were either duplicate selections of the same object, or misclassifications. These numbers are pretty low due to the number of items that were double counted due to being lost by their tracker. In all, though, these numbers indicate that there are improvements that can be made to this model, whether it be through improved tracking, or improved classifiers.

The next table was also calculated through the result table in the previous section. This table represents the number of objects detected properly at some point as the code runs over the video sample. It can be read as, “85% of the cars from file *front-day-3.mp4* were able to be identified at some point in the video”. Having this information, and the knowledge that a lot of double counting happened, we can see that with improved tracking, our code would produce better results.

Perspective	Time of Day	Conditions	Filename	Detection Accuracy			
				Cars	Pedestrians	Bikes	Bus
Front	Day	Clear	front-day-3.mp4	0.857143	0.375	0	1
		Snow	front-snow-2.mp4	0.6875	0	NA	0
	Night	Clear	front-night.mp4	0.75	0	NA	1
		Rain	front-night-rain.mp4	0	NA	NA	0
Side	Day	Clear	side-day-2.mp4	0.833333	0.85714286	0	0
		Rain	side-rain-2.mp4	0.363636	NA	NA	0
	Night	Clear	side-night-4.mp4	0.142857	1	NA	0

In all, the best results came from clear weather during the daytime in both the front and side perspectives (as seen with cars, pedestrians, and busses). Clear weather conditions at night also aided in better road object detection when compared to rainy conditions at night from these particular samples. In general, performance of these classifiers took a hit when any type of weather conditions such as rain or snow were presented. The cells containing *NA* in the table represent scenarios where there were none of that object to detect.

These video samples were taken from cameras at eye-level on the side of the road, meaning our method only requires a video camera of readable quality. This method that we have implemented for video data often runs slower than real time due to the number of calculations necessary with each frame. If a similar tool to ours were needed in real time, it would have to have more processing power than ours currently does in order to compute faster. Distributing calculations to multiple cores / threads / machines would greatly improve speed over the current method which is computing everything sequentially.

Conclusion

From our experiments, we could conclude that clear weather conditions during day time aided better results with recognizing the traffic participants. Clear weather conditions during the night time also gave good results when compared to rain and snowy days/nights. The camera angles also made a difference depending on how the classifiers were trained. In all, our method has a lot of potential, but needs improvement in scalability and accuracy in order to make a difference in real-time.

For future work, we could train our own classifiers with opencv gathering a large number of training images to make sure our classifier picks up the information we want it to from multiple angles. We can also improve the tracking of the vehicles so that the video classifier doesn't count the same vehicle multiple times as it gets lost and found around other vehicles.

Citations

Classifiers

1. https://github.com/AdityaPai2398/Vehicle-And-Pedestrian-Detection-Using-Haar-Cascades/tree/master/Main%20Project/Main%20Project

Videos / Image Screenshots

1. https://www.videvo.net/video/cars-driving-through-heavy-rain-in-taipei/457750/
2. https://www.videvo.net/video/tram-rear-view/4839/
3. https://www.videezy.com/abstract/20738-night-shot-of-cars-and-buses-in-chicago-streets
4. https://www.videvo.net/video/train-and-traffic-on-taksin-bridge/457173/
5. https://www.videvo.net/video/busy-berlin-street-at-night/6594/
6. https://www.videvo.net/video/seoul-street-at-sunrise/456496/
7. https://www.videvo.net/video/melbourne---tram-in-fed-square/1294/
8. https://www.videvo.net/video/busy-street-in-london-at-night/5433/
9. https://www.videvo.net/video/lions-gate-bridge/4713/
10. https://www.videvo.net/video/city-street/4390/
11. https://www.videvo.net/video/financial-district-streets-singapore/458687/
12. https://www.videvo.net/video/snowy-toronto-street/5973/

Other

1. https://github.com/peter-moran/vehicle-detector
2. https://drive.google.com/folderview?id=1cQXRv5lLxG3JoGd3KsBwIFb3QyIXdoma
3. https://www.pyimagesearch.com/2019/12/02/opencv-vehicle-detection-tracking-and-speed-estimation/
4. https://towardsdatascience.com/object-detection-with-less-than-10-lines-of-code-using-python-2d28eebc5b11