Robust and Fast Side View Vehicle Detection and Counting

Meng Han Vivek Ramadoss

Abstract

Traffic sensing is important for traffic researchers, but there is limited infrastructure systems related to it. This project proposed a robust and fast computer vision based traffic sensing approach. Static cameras were installed on the sidewalk and captured the side view of vehicles. The detection algorithm is based on HOG features and is trained with an SVM. Combined with fast motion detection and tracking algorithms, the approach is able to achieve good vehicle counting accuracy with fast computation.

1. Introduction

In the United States, Federal Highway Administration estimates that pre-planned special events are responsible for 93187 million hours of traffic delay annually, with direct costs ranging between $1.7 and $3.4 billion [2]. Data obtained traffic counting can be used to efficiently route traffic. Traffic count results can also be used in traffic signal phase design, leading to optimum time schedule for traffic signals and reduced traffic congestion. However, traffic sensing infrastructure is very limited in most urban infrastructure systems.

Currently, most researchers collect data from GPS data or personal navigation devices. Recently, a smartphone based turning movement counter called TrafficTurk is deployed for wide applications [2]. However, it requires intensive manual efforts and costs.

To reduce manual counting efforts and human errors, we developed a computer vision based vehicle counting system that is based on a robust and fast side view vehicle detection algorithm. This algorithm is a hybrid approach by combining machine learning based object detection and fast motion detection.

2. Literature Review

Literature review was mainly focused on two areas: current practices of traffic sensing, and computer vision-based systems that are being developed for traffic sensing.

2.1. Current Traffic Sensing Practices

Accurate counts of moving vehicles is the basis of traffic analysis. It provides necessary data for calculating traffic flow in a certain time slot so that the result can be utilized as a reference to predict the future traffic volume changes. Based on the analysis, one can further evaluate traffic impacts with regards to various traffic control measures, geometric modifications, or maintenance practices. To carry out such kind of analysis, one needs to either manually, or with the help of various measuring approaches, track the vehicles path through streets from enter to exit.

For manual count of turning movement volumes, data collectors face two specific difficulties as they are conducting field observations. The first one would be the fact that it requires the data collectors to observe both the entry and exit legs simultaneously, but distances between the two are typically too long to check in a short period. The next challenge is that multiple vehicles run through a traffic intersection concurrently will make the collectors get lost during the tracing, or they can simply not trace several moving vehicles together. This will lead to miscounting and wrong statistic data of the turning movement [4].

In a recent project conducted by URS, both manual turning movement counts, which were performed by technicians in the field, and automated turning movement counts, using video collection unit (VCU) developed by Miovision Technologies, are implemented for a TMC task [7]. The VCU is a digital camera recording all vehicle and pedestrian passes, and the recording is further processed to glean the two passing volume counts. In this project, the manual turning movement gathering process is said to be a labor intensive work. For the manual count, each spot for the field data collection is visited in advance of the work so as to confirm the feasibility for a manual counting. It is also required to get approval of various organizations, only for getting access to good views of the intersections. The VCU method requires the camera being mounted on an existing pole, which could be either utility poles, or wood posts.

Nearly 700 hours are used in total for the VCU technology in that project. This project, along with the counting outcomes, verified that the VCU data collection method is sufficiently accurate. It is also capable to count passing objects during an extended period of time at remote locations, saving substantial time and cost for this project. Besides the evident advantage regarding convenience, it only needs one technician to deploy multiple VCU units at many intersection in a relatively short time. Therefore, use of VCU reduces labor cost significantly. According to the report from URS, this new approach has an accuracy that is higher manual counting for the TMC analysis [7].

2.2. Current computer vision based traffic monitoring systems

Peiris and Sonnadara [6] used a single digital camera to extract various traffic parameters, including vehicle count, density, and type at a three-way junction. To minimize the occlusion between vehicles and cover the three junction, the camera was placed high on a building. This method could overcome the limitation of most previous work that the camera is required to be placed in the way of vehicles moving towards the camera. After gray-scaling and noise filtering the images are converted from the video clip, frame by frame differences and background subtraction are used to identify vehicles. A morphological operation is used to build a bounding box corresponding to a identified moving vehicle. Blob tracking is performed using euclidean distance of the vehicles between two consecutive frames. However, vehicles can only be counted when a vehicle enters the selected region from the recorded video, which is shown in Figure 1. In addition, motion detection cannot always guarantee detected objects are vehicles.

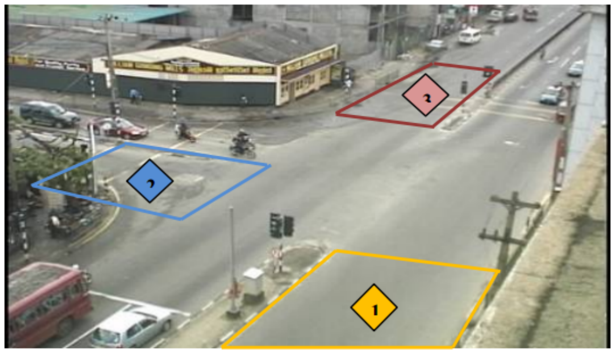


Figure 1: Selected regions for vehicle tracking [6].

Messelodi et al. [5] also proposed a real-time vision system to detect and classify vehicles at urban road intersections. An updated Kalman filter method and a feature-based tracking method were adopted. Relatively less effort was spent focusing on monitoring intersections of urban areas since compared to highways, intersections may incorporate problems including highly variable structure of intersections, the multiple flows of vehicles with different turning movement, and vehicles stopping at traffic lights. Although the used surveillance cameras can be remotely controlled to pan, tilt, or zoom, the boundary of the selected area that is monitored is required to identify through a set of points which defines the entry and exit gates of interest for the turning movements detection.

3. Methodology

The following steps were taken to achieve the intended results.

1. Data Acquisition. A video of a street segment is captured on the ground level. This is to capture the side view of each passing vehicle.

2. HOG-SVM based vehicle detector. To train a vehicle detector, Histogram of Orientated Gradients (HOG) is deployed as feature detector. Support Vector Machines (SVMs) is used to train a classifier that is based on HOG features. The vehicle is then tested with same-scale and multi-scale scenarios.

3. Vehicle Detection. Moving objects were detected via performing background subtraction, morphological closing, rectangle fitting, and filtering. Trained vehicle detector is then used to verify whether each of the moving objects is a vehicle or not.

4. Vehicle Tracking. The tracking of vehicles was accomplished by using Kalman filtering. Each vehicle was given a label and its properties were recorded. Efforts were made to ensure that vehicles that become occluded (i.e. vehicles traveling in opposite directions) would not be re-assigned to a new label in later stage.

5. Vehicle Counting. This is based on vehicle tracking results. The algorithm keeps track of the number of vehicles that have currenty entered the frame. Whenever a new vehicle comes, it would be assigned to an incremented label.

3.1. Data Acquisition

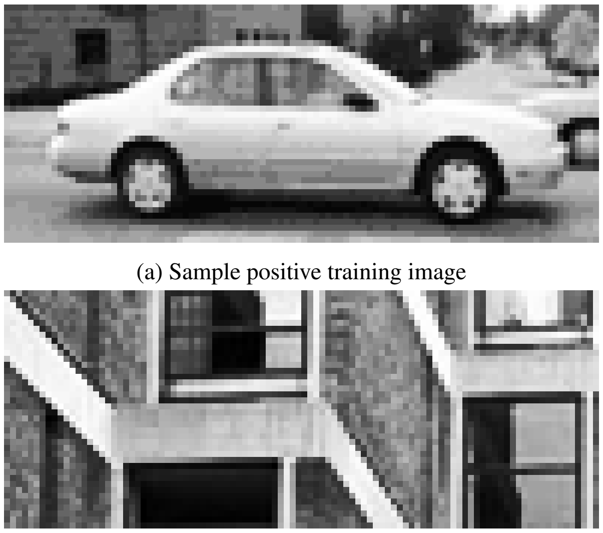
A video clip on a street segment was captured in 30 frames per second (FPS). A digital camera was installed on a tripod to stay static while capturing. The video duration is 7:08 min. It is located on 1st street between Kirby and St Marys. This is a two-way street and there are pedestrians and motorcycles. Also it's flurry and drizzling weather conditions which brings slight camera shake and environment noises. A sample frame from the recorded video is shown in Figure 2.

3.2. HOG-SVM based Vehicle Detector

The dataset that is used for training in this project is from UIUC Image Database for Car Detection (`https://cogcomp.cs.illinois.edu/Data/Car/`). This

Figure 2: Sample frame captured on 1st street.

dataset provides 550 positive images (side views of vehicles), 500 negative images (objects or scenes that are not side views of vehicles), 170 testing images with same scale, and 108 images with different scales. Samples of training dataset are shown in Figure 3.



(a) Sample positive training image

(b) Sample negative training image

Figure 3: Sample training images

Since images are ready, we need to figure out what features we can extract to feed into Machine learning algorithms. Histograms of Oriented Gradients [3] is a widely-used feature descriptor used in computer vision for object detection. It counts occurrences of gradient orientation in localized portions of an image. It transforms any given image into a one-dimension vector. For our implementation, we deployed OpenCV [1] built-in HOG function and the vector length we have is 7656. Note that HOG extracts features from grayscale images.

Now we have the features of training images, we can use feed features into SVM to train a classifier. As mentioned earlier, the feature length is 7656 and thus the trained SVM is of dimension of 7656. We deployed SVMs from Scikit Learn library, which is a Python Machine Learning Library.

To test how this classifier works, we apply the HOG-SVM detector into two testing datasets. The first dataset contains 170 images, of which the vehicles are approximately the same size with the training images. The second dataset contains 108 images, of which the vehicles are in different scales compared with the training images (either smaller or bigger).

For first dataset, a sliding window algorithm would be sufficient. The basic idea is to iterate through the whole image with a window size which is the same with training data, and determine one by one whether it is a vehicle or not. As shown in Figure 5a, two vehicles are correctly detected. Overall, the HOG-SVM detector achieves a 92% accuracy with the same-scale dataset.

For second dataset, a sliding window algorithm would not be enough. Given the vehicle in unknown size, we cannot simply detect with only one scale. Consequently, a multi-scale sliding window algorithm is applied. More specifically, each image is scaled to multiple sizes and each of them would apply sliding window algorithm. Note that the sliding window would remain the same size despite of resized image size. This algorithm is illustrated in Figure 4.

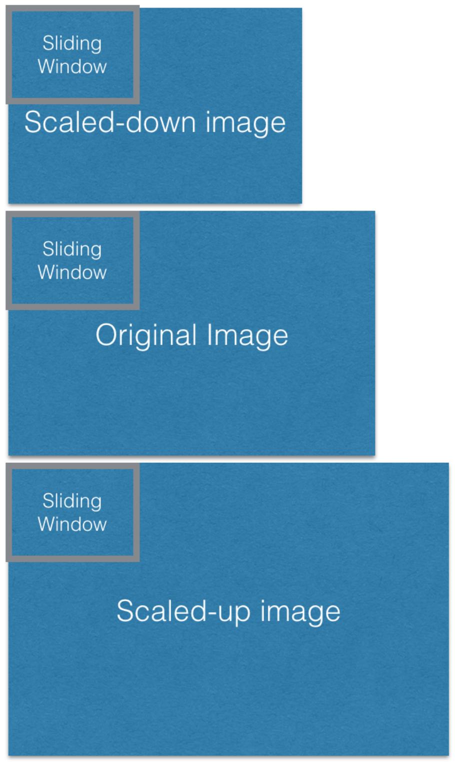
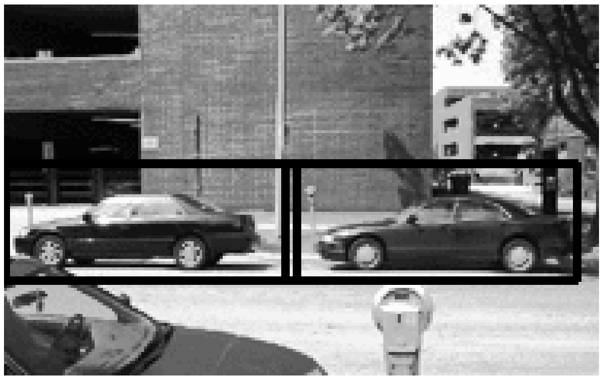


Figure 4: Multi-scale sliding window algorithm illustration

This is much slower than the first dataset, but this is what we should use for the real-world dataset given unknown vehicle sizes. Overall the HOG-SVM detector achieves a 85% accuracy with the multi-scale dataset. One sample result is shown in Figure 5b.


(a) Sample same-scale test image

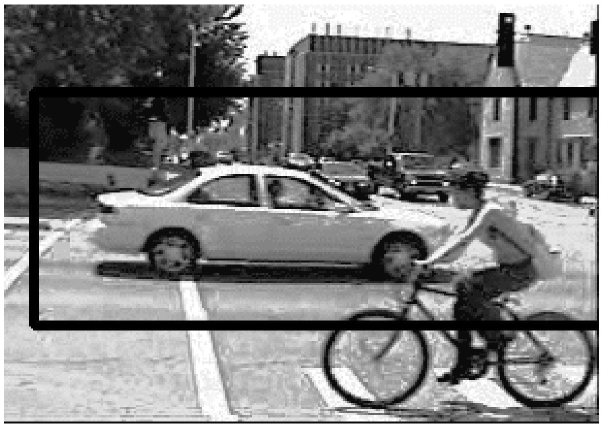

(b) Sample multi-scale test image

Figure 5: Sample multi-scale test image results

3.3. Vehicle Detection

Although we have trained a HOG-SVM vehicle detector, we cannot directly apply it to the whole video frame by frame because of the inefficiency. Since our goal is to achieve processing in real-time, we need to combine Machine Learning based detection with fast motion detection.

Vehicle detection is processed frame by frame of the video, i.e. vehicle detection results are independent as of each frame. To help illustrate the vehicle detection process, one sample frame is included in Figure 6, as well as all intermediate results of it in following subsections.

3.3.1 Background Subtraction

Background subtraction provides the very first step for vehicle detection process. We deployed BGS Library [8] to ac-

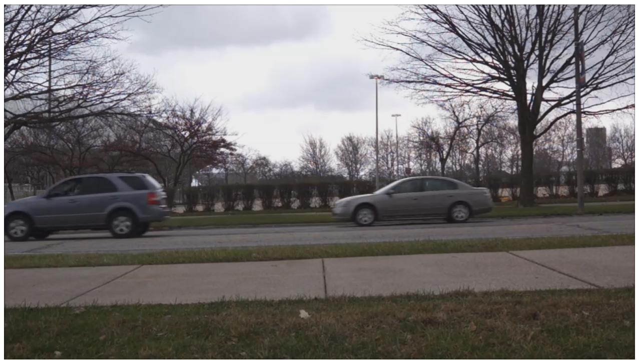


Figure 6: Sample source frame.

complish this task. BGS is a OpenCV based library that provides more than 30 different background subtraction methods. After the evaluation of accuracy and computational complexity, Frame Difference algorithm is selected. The resulting frame after background subtraction is shown in Figure 7.

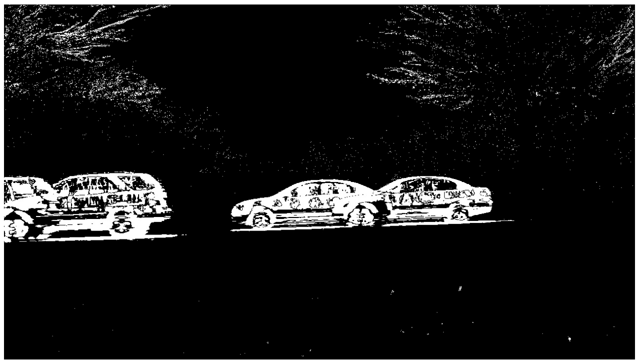


Figure 7: Sample frame after background subtraction.

3.3.2 Morphological Closing

The result after background subtraction still requires improvements due to the voids inside detected regions. Also there are lots of environment noises, e.g. moving leaves on the tree blown by strong wind. A morphological closing transformation was applied to fill in the voids to improve accuracy. An OpenCV built-in function was used to accomplish this. A sample result is shown in Figure 8.

3.3.3 Fitting Rectangles

Bounding rectangles were drawn around the contours of the detected regions. Due to the presence of noise, some of the drawn rectangles needed to be filtered. Rectangles with sizes that deviated too much from that of a regular vehicle

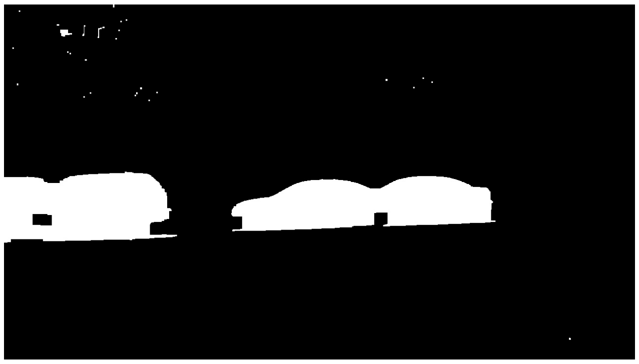


Figure 8: Sample frame after morphological closing.

were first excluded. An example of this procedure is shown in Figure 9.

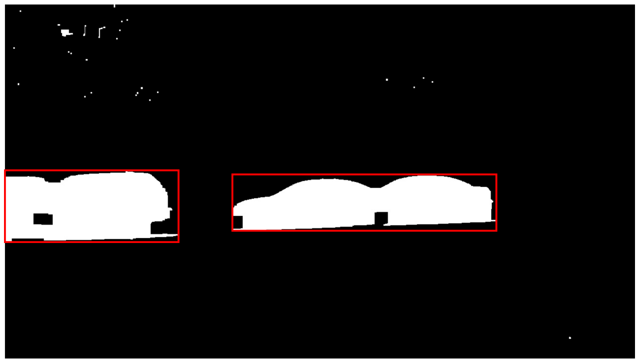


Figure 9: Minimum area rectangles that are fit to contours.

3.3.4 HOG-SVM Vehicle Detector

Instead of applying the HOG-SVM vehicle detector to the whole image with multi-scale sliding window algorithm, it only has to be applied to the much smaller potential vehicle boxes passed by the above step. This is based on the assumption that vehicles on the street are moving, so that only moving objects are potentially vehicles. This step dramatically decreases the computational time of HOG-SVM based vehicle detector.

For each of the bounding rectangles passed in from the last step, HOG-SVM multi-scale sliding window algorithm would be applied one by one, and determines whether it is a vehicle or not. In this example shown in Figure 9, two boxes that pass in are both vehicles, detected as shown in Figure 10.

Comparing Figure 10 with Figure 6, the detected vehicles can be validated.

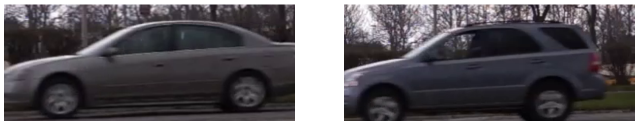

(a) Detected vehicle 1 (b) Detected vehicle 2

Figure 10: Detected vehicles

3.4. Vehicle Tracking

There are two tracking scenarios. Firstly, tracking is by Kalman-filter predictions. This is based on the assumption that detections are correct. In other words, detected vehicles are the actual vehicles in the current frame. Although the whole detection process is trying to accomplish this goal, correctness cannot always be guaranteed. Missing detection due to occlusion is very likely to happen. This leads to the second tracking scenarios, where tracking is trying to correct detection. There are several assumptions made, but parameters are not tuned for specific videos for generality.

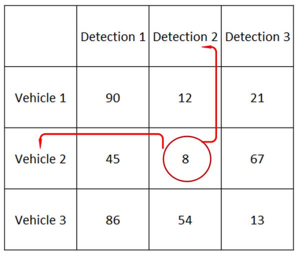
3.4.1 Kalman-Filter Based Tracking

Kalman-filter based tracking was applied to the detected regions in Figure 9. Based on the true centers of a bounding rectangle, which is the current location of a vehicle, Kalman filtering predicts a position for where the vehicle is most likely going to be. The position of each bounding box in the next frame was then compared against each predicted position so vehicles that had already been labeled would maintain their label and new vehicles that had just entered frame would receive a new label.

The process is explained in detail as follows. The very first frame of the video is treated differently from the remaining frames of the video. Nothing has been labeled prior to the first frame, thus the frame requires initialization. Using the first two frames, potential vehicles appeared during the first frame are detected and labeled. All of the subsequent frames then attempts to label vehicles based on frames that immediately precede them. Existing vehicles should maintain their labels and vehicles that had just entered the frame should receive new labels.

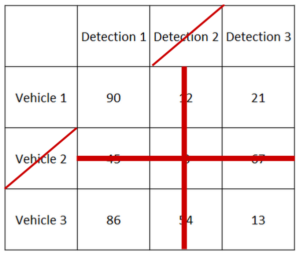
There are three possible labeling scenarios. The first scenario, also the most straightforward, is when the number of vehicles in the preceding frame is equal to the number of detection within the current frame. The Euclidean distances of the center points of the detected regions are calculated between two consecutive frames. An example is shown in Figure 11. Detection 1-3 are regions that have just been detected in the current frame, while Vehicle 1-3 are vehicles that had been labeled from the preceding frame.

Shown in Figure 11a, the minimum of the 9 calculated Euclidean distances is found. The labeled vehicle and the associated newly detected region are then believed to be the same vehicle. Shown in Figure 11b, the entire row and column can be eliminated. The procedure is repeated by finding the next closest Euclidean distance to match labeled vehicles with newly detected regions until only one possible match remains. This is shown in Figure 11c.



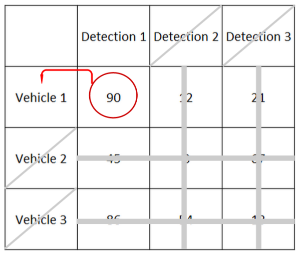
	Detection 1	Detection 2	Detection 3
Vehicle 1	90	12	21
Vehicle 2	45	8	67
Vehicle 3	86	54	13

(a) Detected vehicles are pairing by minimum Euclidean distance.



	Detection 1	Detection 2	Detection 3
Vehicle 1	90	12	21
Vehicle 2	45	8	67
Vehicle 3	86	54	13

(b) The associated vehicle and detection are struck out after pairing.



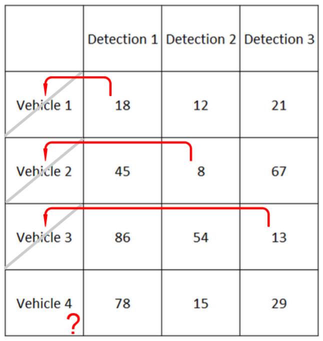
	Detection 1	Detection 2	Detection 3
Vehicle 1	90	12	21
Vehicle 2	45	8	67
Vehicle 3	86	54	13

(c) The remaining vehicle and detection are paired.

Figure 11: Sample vehicle tracking procedure when detected vehicles and existing vehicles share the same count.

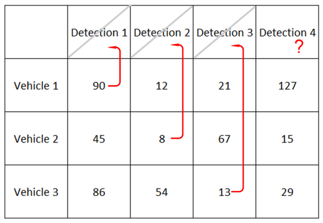
The second scenario involves having more vehicles labeled in the previous frame than the number of regions detected in the current frame. An example is shown in Figure 12. The aforementioned procedure is applied to match up the first 3 vehicles. Based on the where the remaining vehicle was last detected in the previous frame, one of two actions is taken. If the vehicle was last detected somewhere in the middle of the intersection, the vehicle is potentially stopping temporarily and its position is recorded. If in a later frame a vehicle appears with the same position, tracking of the vehicle is resumed. The second action labels the vehicle as having left the intersection, which is taken when the previously labeled vehicle exits the frame of the video.

The third scenario, shown in Figure 13, occurs when there are more regions detected in the current frame than the number of vehicles labeled in the preceding frame. Once again, the previously described procedure is followed to associate newly detected regions with labeled vehicles. A new label is assigned to the final remaining newly detected region.



	Detection 1	Detection 2	Detection 3
Vehicle 1	18	12	21
Vehicle 2	45	8	67
Vehicle 3	86	54	13
Vehicle 4 ?	78	15	29

Figure 12: Sample tracking procedure for the second scenario.



	Detection 1	Detection 2	Detection 3	Detection 4 ?
Vehicle 1	90	12	21	127
Vehicle 2	45	8	67	15
Vehicle 3	86	54	13	29

Figure 13: Sample tracking procedure for the third scenario.

3.4.2 Tracking Corrects Detection

This mainly deals with the scenario where detection is not correct, in particular, missing detection due to occlusion. To deal with this scenario, a buffer zone of two frames is allowed for each detected vehicle. That means, whenever a vehicle is not matched with a detection, it is not deleted until two frames later. There is advantages and disadvantages by doing this. For the good part, this helps handle occlusions. Vehicles travel in opposite directions can easily get occluded by each other. By retaining the vehicle for two frames, we can expect they are still assigned with the same label after occlusion. For the bad part, this delays the deletion for vehicles that actually leave the scene.

3.4.3 Tracking Sample Result

The result of the tracking process is shown in Figure 14. The blue dot is the true center location while the green dot is the predicted location by Kalman filter. By finding the nearest prediction-true location pair in case of multiple vehicle detection, correct labels are assigned.

4. Results [1] & Discussion

The results of the project is shown in several aspects. First, the detection robustness is discussed as of avoiding false positives. Second, vehicle counting is compared be-

[1] Video demos are available at `https://goo.gl/LLaUj0`

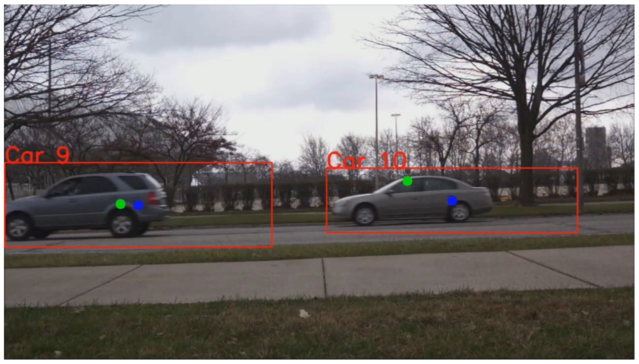
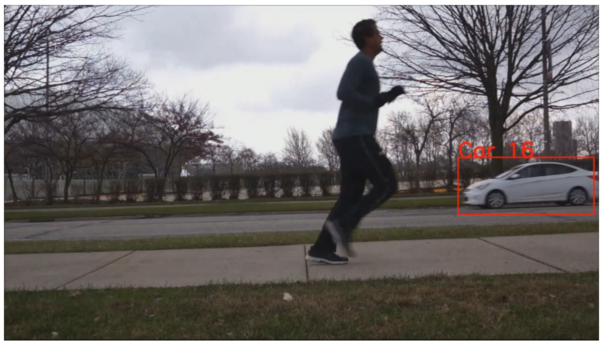


Figure 14: Vehicles are tracked with their centered positions using Kalman Filtering.

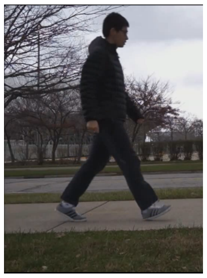
tween human-labeled and machine-labeled. Last but not the least, computational complexity is evaluated on machines.

4.1. Detection Robustness

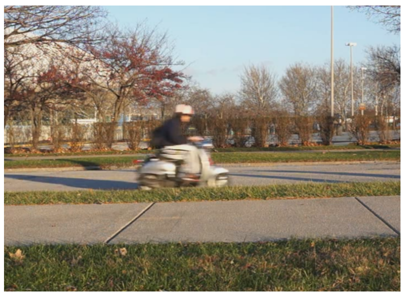
The introduction of HOG-SVM vehicle detector greatly increases the robustness of vehicle detection. It helps to avoid detecting moving objects that are not vehicles, as shown in Figure 15. In our observation, the proposed algorithm is robust to, and not limited to pedestrians, motorcycles, bicycles, flurry weather and drizzling weather, etc.



(a) Running person is not detected while vehicle gets correctly detected.



(b) Pedestrian is not detected.



(c) Motorcycle is not detected.

Figure 15: Detection robustness of our approach.

Elimination of false positives sometimes eliminate true positives as well. Since the training dataset only contains sedans and small SUVs, it does not have consistent detection results to trucks, which are popular in Midwest. This problem can be solved by introducing trucks in positive training dataset.

4.2. Counting Accuracy

In our testing video, the ground-truth (counted by group members) count of traffic is 64 vehicles, while the algorithm counts 67. This gives us a 95% counting accuracy.

As for error analysis, the extra counts are mostly due to occlusion. Once occluded, detected vehicles that have already been assigned a label, is regarded as moving out of the scene. After occlusion, the same vehicle might be detected once again. However, in this case it would be regarded as a new vehicle. This results in a situation where the same car was counted twice. That explains why the algorithm counts more than the ground-truth.

Although we try to handle this in the tracking algorithm, it's a trade-off of deleting vehicles that have actually left the scene and deleting those who are temporarily occluded. We allow only 2 frames of buffer zone to wait for any vehicle to be detected once again to avoid re-assigning a new label. However, that becomes a parameter tuning game and we did not specifically tune for testing video.

4.3. Computational Time Evaluation

The system is capable of processing videos at an accelerated pace. The duration of the input video was 7 minutes and 8 seconds. At a processing rate of 5 FPS, the system took about 190 seconds to process the entire video on a standard computer laptop, which is only 44% of the video duration. The performance is expected to improve even further if computers with better hardware are used for processing.

5. Conclusions

By analyzing recorded videos of traffic, the computer vision based monitoring system discussed here is capable of automatically counting the number of vehicles that enter the scene, with decent accuracy and fast computational time.

Currently, there does not exist any commercially available automated systems that are capable of monitoring traffic with exceptional accuracy. The system developed provides many benefits compared to counting vehicles manually and other systems. The benefits include, but are not limited to:

1. Robust detection. Currently most computer vision traffic sensing systems rely on only motion detection, which is not robust enough to complex road conditions (pedestrians, bicycles, motorcycles, weather). Our approach contains a trained Machine Learning based ve-

hicle detector, which provides much more confidence in identifying moving objects as vehicles.

2. Reduced cost. Counters are no longer required for counting vehicles. The amount of savings varies with the size of the project. The larger the project, the greater the amount of savings.

3. Easier deployment. A static camera on ground level is enough to capture videos that can be handled by our approach.

Errors arose when the system attempts to detect vehicles that were partially occluded. This error directly influenced the accuracy of vehicle counting. Despite the presence of the limitations, the system was able to produce good results when analyzing the test data. The accuracy of the vehicle counting capability was 95%.

The limitations of this approach is that the trained detector is only able to handle vehicle side views. Consequently, this algorithm cannot be directly applied to traffic intersections, where different vehicle views would be present.

References

[1] G. Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[2] M. Carvaja. Trafficturk: a smartphone based turning movementcounter for monitoring extreme congestion events, 2013.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[4] M. P. Dixon, A. Abdel-Rahim, M. Kyte, P. Rust, H. Cooley, and L. Rodegerdts. Field evaluation of roundabout turning movement estimation procedures. *Journal of Transportation engineering*, 133(2):138–146, 2007.

[5] S. Messelodi, C. M. Modena, and M. Zanin. A computer vision system for the detection and classification of vehicles at urban road intersections. *Pattern analysis and applications*, 8(1-2):17–31, 2005.

[6] K. Peiris and D. Sonnadara. Extracting traffic parameters at intersections through computer vision. In *Proceedings of the Technical Sessions*, volume 27, pages 68–75, 2011.

[7] R. Schneider. Comparison of turning movement count data collection methods for signal optimization study. *URS Corporation*, 2011.

[8] A. Sobral. BGSLibrary: An opencv c++ background subtraction library. In *IX Workshop de Viso Computacional (WVC'2013)*, Rio de Janeiro, Brazil, Jun 2013.