A YOLO-based Traffic Counting System

Jia-Ping Lin and Min-Te Sun
Department of Computer Science and Information Engineering
National Central University
Taoyuan 320, Taiwan
Email: 105522018@cc.ncu.edu.tw, msun@csie.ncu.edu.tw

***Abstract*—Image recognition can be applied in many applications of Intelligent Transportation System. Through automated traffic flow counting, the traffic information can be presented effectively for a given area. After the existing image recognition model process the monitoring video, the coordinates of objects in each frame can be easily extracted. The extracted object coordinates are then filtered to obtain the required vehicle coordinates. To achieve the function of vehicle counting, it is necessary to identify the relationship of vehicles in different frames, i.e., whether or not they represent the same vehicle. Although the vehicle counting can be achieved by using the tracking algorithm, a short period of recognition failure may cause wrong tracking, which will lead to incorrect traffic counting. In this paper, we propose a system that utilizes the YOLO framework for traffic flow counting. The system architecture consists of three blocks, including the Detector that generates the bounding box of vehicles, the Buffer which stores coordinates of vehicles, and the Counter which is responsible for vehicle counting. The proposed system requires only to utilize simple distance calculations to achieve the purpose of vehicle counting. In addition, by adding checkpoints, the system is able to alleviate the consequence of false detection. The videos from different locations and angles are used to verify and analyze the correctness and overall efficiency of the proposed system, and the results indicate that our system achieves high counting accuracy under the environment with sufficient ambient light.**

Keywords*—*YOLO, traffic flow, image recognition.

I. INTRODUCTION

To effectively control traffic conditions and solve problems such as traffic congestion and traffic accidents, many developed countries, such as US, Japan, and Germany, have started to develop the Intelligent Transportation System (ITS) [1]. ITS is a way to integrate many advanced technologies, such as Car Navigation [2], Traffic Signal Control Systems [3], and Automatic Number-plate Recognition [4], to a single transportation management and control system. One of the fundamental building blocks for these technologies is traffic flow identification [5], i.e., to count the number of passing-by vehicles at a given point.

The mainstream methods to count and classify vehicles can be roughly segmented into hardware solutions and software solutions. Inductive loops [6] and piezoelectric sensors [7] are the two most widely used systems in the ITS hardware solution. Although the hardware solutions have higher accuracy than the software solutions, it cost more to maintain and in pavement destruction. With the rapid improvement of computer computing performance and the development of image recognition technology in recent years, the software solutions use the technique of image recognition to calculate the vehicle passing through the surveillance screen. After all the vehicles in the video are identified by the trained model, the system needs to find out the relevance of the vehicles which detected in different frames to achieve the purpose of vehicle counting. Although using the tracking algorithm to process the coordinates of the detected vehicle in each frame can achieve the purpose of vehicle counting, recognition failure in a short period of time may cause a wrong tracking. It will lead to a bad performance of traffic counting.

In order to have a better understanding of the traffic condition in National Central University, we propose a counting system which is able to calculate the traffic flow in real-time through the camera at the entrance and exit. This research uses YOLO as the infrastructure for image recognition of our system. After YOLO identifies a vehicle in a frame, our counting system has to know whether the vehicles recognized across different frames are the same vehicle or not. If the counting system is capable of such a task, it can then counts how many vehicles passing through a specific checkpoint. The experiments show that, with sufficient ambient light, the proposed system can reach high counting accuracy.

The rest of this paper is organized as follows. In Section II, the main structure of the system and the operation of each function blocks are explained. In Section III, videos which are recorded at entrance and exit are used to evaluate the performance of the system. Finally, Section IV concludes this paper.

II. DESIGN

A. Design Goals

In this section, we will introduce our design goals.

1) *Real-time Counting:* A robust Vehicle Counting System needs to accomplish the job in real-time.
2) *Multiple-object Sensing:* To accurately count the traffic flow, the Vehicle Counting System needs to be capable of sensing multiple objects.
3) *Object Classification:* In the real world scenario, there will be many types of objects picked up by the camera. Consequently, the Vehicle Counting System needs to be able to extract vehicles from other types of moving objects.

B. System Model

As illustrated in Figure 1, our system has three major function blocks, include Detector that generates the bounding box of vehicles, the Buffer which stores coordinates of vehicles, and the Counter which is responsible for vehicle counting. The operation flow of this system is described as follows. The

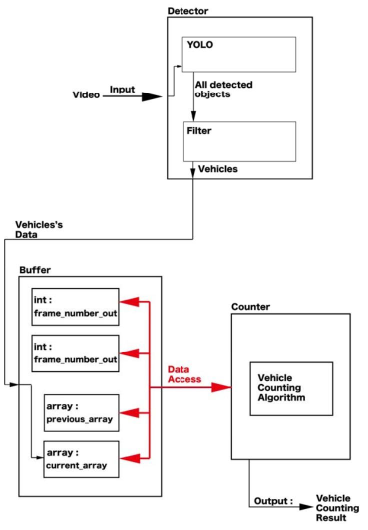


Fig. 1: System Architecture.

user must determine the source of the data entry first, whether it be a recorded video or a webcam. Once our system receives the video, the detector will process the video and output the bounding boxes as the input for the buffer. Finally, the counter will process the data that stores in the buffer and output the result of the vehicle counting. The actual operation of these blocks in our system will be further elaborated in the following subsections.

1) Detector: The detector is the first block after the video is fed. The detector must do preliminary extraction and processing of the video for subsequent uses. Our system uses the trained YOLO model as the basic architecture of the detector. Note that YOLO is modified to suit the situation we want to apply. In reality, there will be other moving objects picked up by the camera, such as bicycles, people, and animals, so we must extract only the vehicles for counting. YOLO can identify 80 items using the weights trained by the COCO [8] training dataset. However, our Vehicle Counting System only needs to deal with the following items: car, bus, and truck. Hence, we modified the original YOLO code to only output the coordinates of the above three objects and mask out other moving objects in the screen. However, in our experimental films, vehicles are sometimes detected as trains when passing through the gate. Therefore, in the context that has a gate, we will list trains as items of interest in the system. The operation of the detector is shown in Figure 2.

2) Buffer: After the film is processed by the detector, the bounding boxes required by the system are extracted and the data of these bounding boxes is stored in the buffer in the system. Our buffer maintains two arrays and two ints for each checkpoint. The details of checkpoint will be further described in the next section.

The four data structures stored in the buffer are introduced below:

1) *previous_array:* The previous_array is used to store the bounding boxes of all vehicles in the previous frame.
2) *current_array:* The current_array is used to store the bounding boxes of all vehicles in the current frame.
3) *frame_number_out:* The number of the last frame that has a vehicle leaving the screen.
4) *frame_number_in:* The number of the last frame that has a vehicle entering the screen.

3) Counter: Before using our system, the user must decide a checkpoint. After the checkpoint is selected, the counter will check if there is a vehicle passing the checkpoint frame by frame. For each vehicle coordinate in the current frame, we need to find out the pair which has the shortest distance between this coordinate and all vehicle coordinates in the previous frame. Then we use a threshold to remove the following cases.

1) *Newly appearing vehicles should not find the corresponding point in the previous frame.*
2) *Those vehicles that have not been detected by YOLO in the previous frame should not find the corresponding coordinate.*
3) *When two vehicles are in close proximity and one of them is not detected by YOLO in the previous frame, it will trick the counter to consider the two different vehicles as the same vehicle.*

To avoid aforementioned issues, the threshold should not be greater than the average length of vehicles. After we remove the exceptions, the counter then uses whether the pair of coordinates that are close to each other in consecutive frames crosses the checkpoint to calculate the traffic flow.

In the process of implementation, two additional issues were identified.

1) *Duplicate Counting* When YOLO generates a bounding box, the variance in the size of the boundary is quite large, which may cause difficulties for Counter. Therefore, the center coordinates of the bounding boxes are used to do the counting. In order to facilitate the debugging, the vehicle counting process is recorded into a file. Through this log file, we find out that the bounding box may also cause a vehicle to be counted twice due to the fluctuation of the bounding box. Sometimes the coordinate of a vehicle will fluctuate rapidly, which can lead to the vehicle being counted more than once. A particular example is illustrated in Figure 3. In frame 7412, the

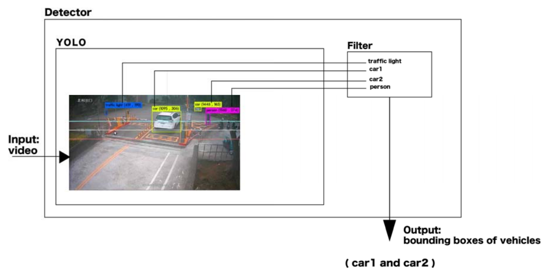
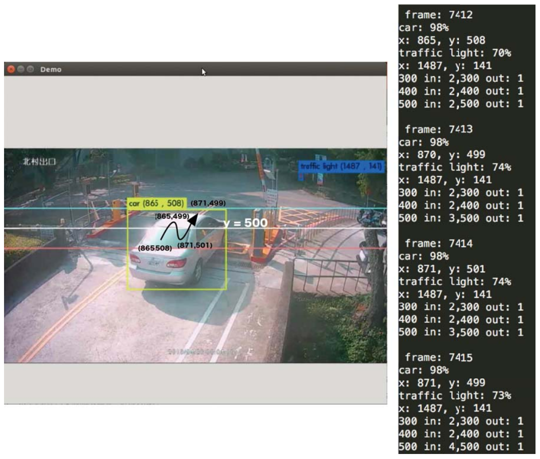


Fig. 2: Detector.

coordinate of the vehicle is (865, 508), and in the next frame, it moves to (870, 499). Apparently, this vehicle crosses the checkpoint y = 500, therefore it is counted as a single entering event. However, the coordinate of the same vehicle somehow fluctuates back to (871, 501) in frame 7414 and move to (871, 499) in the next frame. As a result, this vehicle is counted twice. This issue is called Duplicate Counting.



```
 frame: 7412
car: 98%
x: 865, y: 508
traffic light: 70%
x: 1487, y: 141
300 in: 2,300 out: 1
400 in: 2,400 out: 1
500 in: 2,500 out: 1

 frame: 7413
car: 98%
x: 870, y: 499
traffic light: 74%
x: 1487, y: 141
300 in: 2,300 out: 1
400 in: 2,400 out: 1
500 in: 3,500 out: 1

 frame: 7414
car: 98%
x: 871, y: 501
traffic light: 74%
x: 1487, y: 141
300 in: 2,300 out: 1
400 in: 2,400 out: 1
500 in: 3,500 out: 1

 frame: 7415
car: 98%
x: 871, y: 499
traffic light: 73%
x: 1487, y: 141
300 in: 2,300 out: 1
400 in: 2,400 out: 1
500 in: 4,500 out: 1
```

Fig. 3: Fluctuation Example.

2) *False Detection* In rare cases, such as changes in light and shadow, special colors of the vehicles, and special shapes of the vehicles, YOLO does not detect the vehicles in a number of consecutive frames. This issue comes from the image detection model used in YOLO. Unless the image detection model in YOLO can be improved to have 100% accuracy (which is out of the scope of this paper), this issue is not likely to solve. A particular example is illustrated in Figure 4, where the vehicle detected in frame 6428 and frame 6461 was not detected between frame 6449 and frame 6453.

The following efforts have been made to address two aforementioned issues.

1) *Duplicate Counting:* In order to solve the problem of duplicate counting, we maintain two arrays for each checkpoint. A frame number will be saved when a vehicle enters or exits the checkpoint. If the two incoming or outgoing frame numbers are too close to each other, the later one will be discarded, as it is not possible for two vehicles to enter or exit the gate within such a short period of time. By doing so, the problem of duplicate counting can be solved. The size of the frame gap can be adjusted by the average speed of the road. For example, in the video we used to make experiments, the vehicle needs to take more than 3 seconds to pass through the gate. And the FPS of this video is 30fps. Therefore the frame gap should not be less than 90 in this case.
2) *False Detection:* So far, it is still impossible to train a model with a recognition rate of 100%. Therefore, we can only achieve more optimal results by adding a few more checkpoints and using a funtion among these checkpoints to take the best result. Three checkpoints are utilized to achieve more optimal results.

III. Performance

In this section, we will present and analyze the experiment results.

A. *Experimental environment*

There are two test videos for this experiment. The first one covers the side entrance to National Central University. It has two lanes, one for entering and the other for exiting the university. The second one covers the backdoor exit for National Central University. It covers only two lanes, both for leaving the university. In both videos, our system counts the amount of traffic during a period of time and compares the amount of traffic with the ground truth obtained by human observation to demonstrate the performance. We divide the day into three time periods: morning, afternoon, evening, and each time period take two hours for the experiment.

B. *Finding the best function*

As mentioned in Section II, YOLO may experience False Detection in some cases. In order to improve system performance, the system counts the number of vehicles among three checkpoints for further processing. As illustrated in Figures 5, the number in each bar represents the counting result for the video at a different time interval, and the line in figure is the ground truth. It is clear that the result of the $y = 400$ is the worst. The reason leads to this bad performance is that the vehicle is in the closest proximity to the fence among three checkpoints. In such a case, the fence affects YOLO to do the detection. However, regardless of $y = 300$ or $y = 500$, no one consistently performs the best. Consequently, a good function needs to be identified to get the best result among these checkpoints. The performance of the maximum function and average function are shown in Figure 6. As mentioned in Section II, the system has solved the duplicate problem, and the probability of occurring false positive in YOLO is also extremely low. Without a doubt, the experimental results when using the max function is better than using the average function. After the above analysis, the system adopts the max function to extract the best results from these three checkpoints.

The accuracy of experiment is shown in Table I. After using the max function, the result of vehicle counting during daytime reaches 100% accuracy. However, the counting result is unacceptable when the vehicle exits at night. As shown in Figure 8, when the vehicle exits at night time, the vehicle headlights will cause overexposure, which will in turn cause YOLO unable to recognize the vehicle. To deal with this issue, a night vision camera and additional training of YOLO model for night vision vehicle images may be a solution.

TABLE I: The accuracy after using max function.

	morning	afternoon	night
inbound Accuracy:	100%	100%	100%
outbound Accuracy:	100%	100%	80%

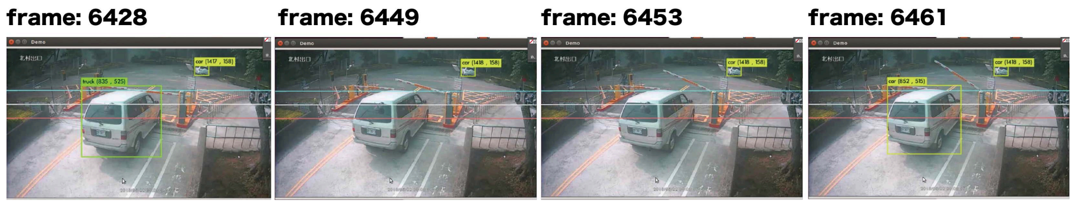


Fig. 4: An Example of False Detection.

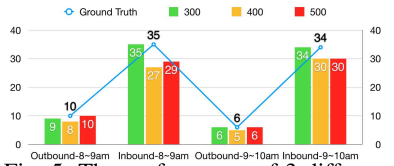


Fig. 5: The performance of 3 different checkpoints(morning).

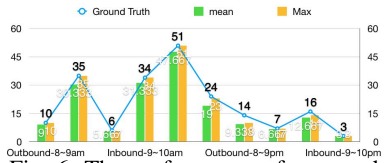


Fig. 6: The performance of max and mean functions.

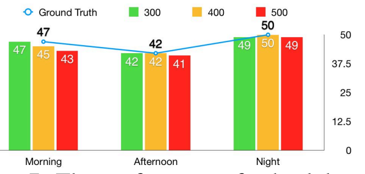


Fig. 7: The performance for backdoor video.

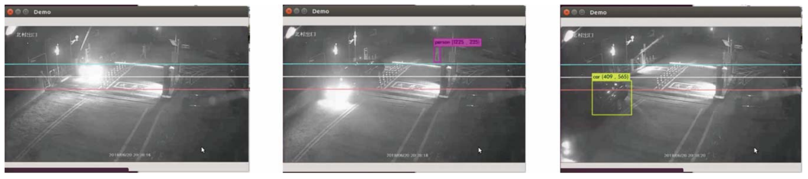


Fig. 8: A vehicle exits at night.

C. *Applying to the backdoor video*

In addition to the side door video, the experiment is also conducted using the backdoor video. As shown in Figure 7, the overall accuracy is higher than that of the side door video. This is because the vehicle captured in this video is clearer and the angle is more inline to the vehicle. Hence, the error rate of YOLO recognition is also reduced. As shown in Table II, it is worth mentioning that because this video has better ambient light, the accuracy of the vehicle counting at night is better than that of the side door video.

TABLE II: The accuracy of backdoor video

	morning	afternoon	night
inbound Accuracy:	100%	100%	100%

In all kinds of vehicles, the hatchback is most likely to produce false detection. This may be due to the fact that the training set of YOLO model had fewer picture for such vehicles in the past. It can be addressed by retraining the YOLO model with more pictures of hatchback vehicles.

IV. Conclusions

Intelligent Transportation System promises a safe transportation environment to us. Benefitting from the rapid development of image recognition, it can achieve the comprehensive task like traffic flow counting easily. By modifying the existing YOLO model, the coordinates of the detected objects are filtered to obtain the required vehicle coordinates. In this paper, we proposed a system to count vehicles by utilizing simple distance calculation with the modified YOLO model. The system architecture consists of the Detector to generate the bounding box of vehicles, the Buffer to store coordinates of vehicles, and the Counter responsible for vehicle counting. By experimenting with two videos taken from different locations and angles, the correctness and overall efficiency of the system are demonstrated.

References

[1] G. Dimitrakopoulos and P. Demestichas, "Intelligent transportation systems," *IEEE Vehicular Technology Magazine*, vol. 5, no. 1, pp. 77–84, 2010.

[2] K. Morinaga, "Car navigation system," May 16 1995, uS Patent 5,416,478.

[3] K. H. Molloy, J. P. Ward, and V. M. Benson, "Traffic signal control system," Jul. 4 1972, uS Patent 3,675,196.

[4] R. Lotufo, A. Morgan, and A. Johnson, "Automatic number-plate recognition," in *Image Analysis for Transport Applications, IEE Colloquium on*. IET, 1990, pp. 6–1.

[5] M. Cremer and M. Papageorgiou, "Parameter identification for a traffic flow model," *Automatica*, vol. 17, no. 6, pp. 837–843, 1981.

[6] J. Gajda, R. Sroka, M. Stencel, A. Wajda, and T. Zeglen, "A vehicle classification based on inductive loop detectors," in *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, vol. 1. IEEE, 2001, pp. 460–464.

[7] D. M. Merhar, "Piezoelectric vehicle impact sensor," Oct. 31 1972, uS Patent 3,701,903.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.