ICCSPA20 1570627532

An Efficient Multi-Object Tracking and Counting Framework Using Video Streaming in Urban Vehicular Environments

Ahmed Dirir, Mohammed Adib, Anas Mahmoud, Moatasem Al-Gunaid, Hesham El-Sayed
College of Information Technology,
UAE University,
Al Ain, Abu Dhabi, UAE
e-mail: {helsayed}@uaeu.ac.ae

Abstract

Object counting is an active research area that gained more attention in the last few years. Since deep learning methods outperformed all other object detection algorithms, the design of efficient object counting algorithms became more realistic and achievable. Numerous algorithms targeting various challenges associated with object counting have been introduced. In a smart transportation system, vehicle counting plays a crucial role as it helps in creating autonomous systems, and better planning for roads. In this paper, we present an efficient object counting system and assess its performance using a dataset of 20 different videos. The proposed system leverage an efficient object detector, and object tracker to perform the counting. This paper combines different approaches to count objects by tracking them, but performs the tracking operation efficiently. Therefore, the proposed systems achieve high accuracy values with low processing time.

Keywords

Counting; Object Detection & Tracking; YOLOv2; KCF.

I. INTRODUCTION

Object detection algorithms based on deep learning methods completely outperformed others. They can be divided into two main categories, the one-stage approach, and the two-stage approach [1] [2] [3]. The one-stage approach uses a fixed number of predictions on a grid to define the bounding boxes around the object, then it tries to classify each bounding box and map it into one of the different classes defined by the neural network. YOLO (You only look once) [4] [5] [6], and SSD (Single Shot Detector) [7] [8] are one stage approach detection algorithms that are highly prioritized.

The two-stage approaches calculate the position and size of the bounding boxes using a neural network before classification. Therefore, the two-stage approach is more accurate, but it involves more computational power because two neural networks are implemented, one for forming the bounding boxes, and one for classification. R-CNN (Region-based Convolutional Neural Networks) is a two-stage approach algorithm that is widely used. There are two other versions of R-CNN, Faster R-CNN, and Mask R-CNN [9] [10].

Object tracking algorithms are extremely useful in many applications, they can be divided into four main categories [11], matching based tracking; filtering based tracking, class-based tracking, and fusion-based tracking. More recently, several deep learning methods are combined with other algorithms to improve tracking performance [12].

Correlations filters correlate two samples to find the similarity between them. KCF (Kernelised Correlation filter) got more acceptance because it has shown higher accuracy and speed compared to other correlation filters based trackers [13].

Object Counting based on deep learning methods is an active research area that attracted many scholars in recent years. Some tried to establish an object counting quantitative comparison between background subtraction, Viola-Jones, and Deep Learning Methods on four different datasets [14]. Others investigated YOLO potential in object counting. One group build a traffic counting system based on YOLO. They have utilized simple distance calculations to achieve the purpose of vehicle counting, and they added checkpoints to alleviate the consequence of false detection [15] [17].

Similarly, another group used YOLO as a primary object detector, and they have combined it with correlation filters to build an object counting system [16]. Once an object is detected, they start tracking it until it gets out of the frame or disappears. However, this is a computationally expensive algorithm since it tracks every object in the frame. Moreover, it is more vulnerable to tracking failures as objects that disappear and reappear might be counted twice.

Finally, to avoid counting the same object twice, and to reduce the computational complexity, [18] suggested detecting objects every N frames, then using Kanade–Lucas–Tomasi feature tracker (KLT) to track counted objects.

In this paper, we propose an efficient object counting system that integrate deep learning (YOLO) for object detection, and Kernelised Correlation Filter (KCF) for object tracking. The proposed system utilizes a simple distance calculation, and KCF tracker to count objects once. The tracking is implemented in a narrow region instead of the whole frame to reduce errors and time complexity. The proposed object counting algorithm is fast so that there is no need to detect objects every N frames. It works frame by frame as will be explained later in the paper. The rest of this paper is organized as follows. Section II describes the proposed system architecture. Section III presents the datasets preparations, and Section IV shoes the performance evaluation of the proposed system. Finally, Section V concludes the paper. The system has been tested against a dataset of 20 different videos. These videos varied in the view angle, speed of motion for the objects, density of objects and the image quality.

978-1-7281-6535-6/20/$31.00 ©2020 IEEE

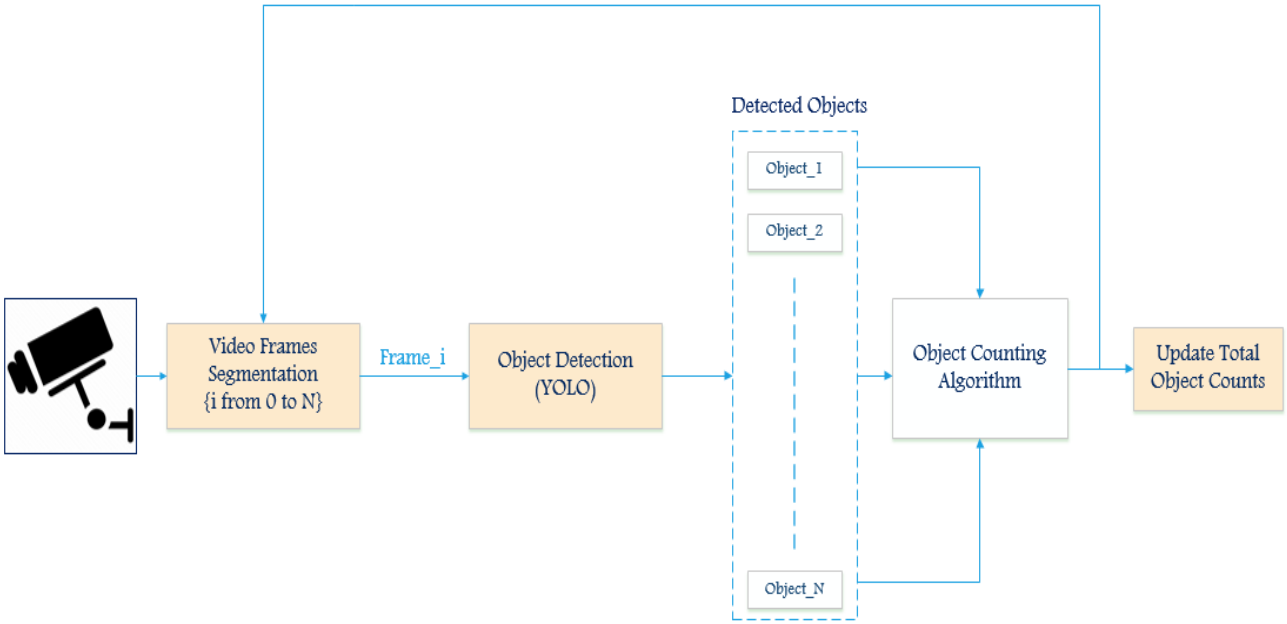


Fig.1 Proposed Object Counting System Archeticture

II. System Archeticture

The High-level Architecture of the system is shown in Fig 1. The system starts with the video segmentation stage where the upcoming video stream is segmented into **N** frames. The system feeds these frames sequentially to the object detection algorithm (YOLO). YOLO output a list of detected objects in each frame. Then, these detected objects are the input for the object counting algorithm where the actual counting is implemented. The object counting stage updates the total objects count if the input to the object counting stage satisfies the conditions as will be explained in the next sections. This process will be repeated for **N** frames.

A. Initial Setup

The initial setup required is the positioning of the crossing line in the best location such that the maximum accuracy is achieved. The crossing line is the line in which objects are counted once they crossed it. Since YOLO is the primary and only object detection in the system, the position of the line will affect the system accuracy as YOLO performance changes with objects sizes, shapes, and orientation. Fig 2 shows two different positions of the crossing line. The red line is clearly in a better position as the number of objects detected by YOLO is more than the number detected in the yellow line region. The second disadvantage of the yellow line is the blind spot region where the cars in the most left lane might be missed if large objects (Trucks) are passing in the adjacent lane (Center lane). In our system, we choose the position of the crossing line manually by inspection. It is worth to note that throughout all experiments we conducted, we can conclude that the best position for the crossing line should be within the center of the frame, as YOLO tends to be more accurate.

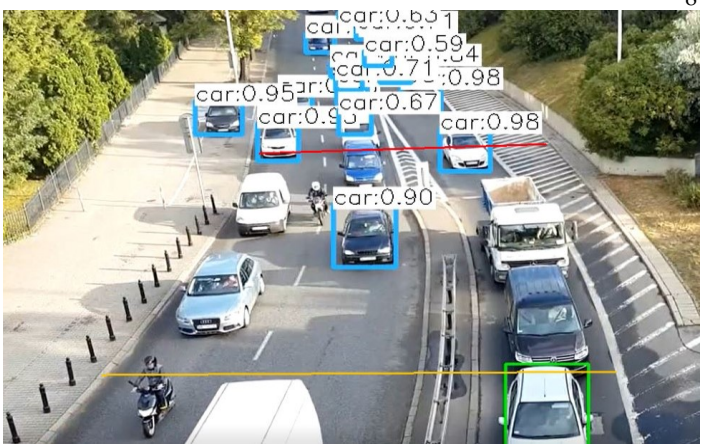


Fig.2 Different positions of the crossing line

B. Object Detection Stage

YOLO defines a collection of bounding boxes with different aspect ratios. The aspect ratios are calculated by applying the k-means clustering algorithm on all bounding boxes in the training dataset [7]. The bounding boxes are fed to the neural network to classify each bounding box and map it to one of the predefined classes. The detection network has 24 convolutional layers and two fully connected layers at the end. In this stage, each frame is possessed with YOLO to find out the bounding boxes for all objects in the current frame. These objects alongside with the output from the tracking stage to is used to find the target object. YOLO processes each frame independently from previous frames; thus, it can recover from object loss in one frame. These features make YOLO excellent candidate for object detection, unfortunately, not for object tracking. Another advantage of YOLO is the ability to detect scale change, where some of tracking algorithms suffer the most.

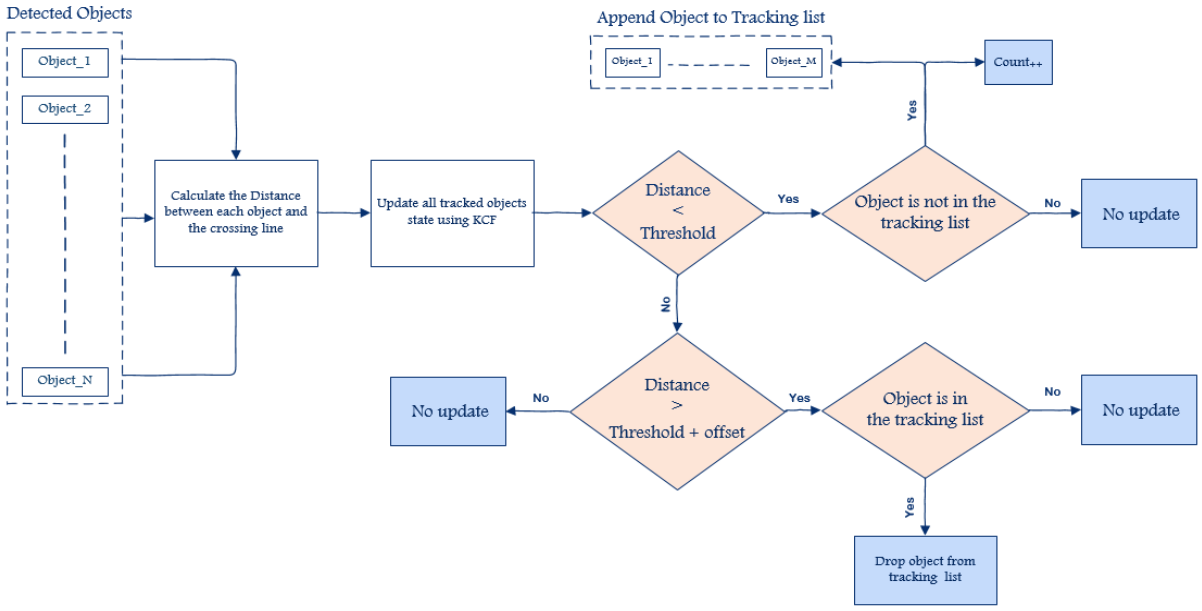


Fig.3 Object Counting algorithmand flow diagram

C. *Object Counting stage*

The object counting stage work flow is shown in Fig 3. For all detected objects after being filtered in the object detection stage, the distance between each object and the crossing line is calculated. If the distance between an object and the crossing line is less than a threshold τ, and if the object is not in the tracked objects list, the system starts tracking this object using KCF tracker, increment the total objects count by one, and add this object to the tracked objects list. The system continuously updates the state of each tracked object in the tracked objects list using the KCF tracker. Once an object is added to the tracked objects list as shown in the green box in Fig 4, the system will keep tracking it until it gets far from the crossing line by an offset α. Finally, the system will discard any tracked object, which gets far from the crossing line by more than the predefined offset α. We have chosen the threshold τ and the offset α experimentally; therefore, they might vary slightly from one street environment to another. In our system, the threshold is 10 and the offset is 20-pixel values.

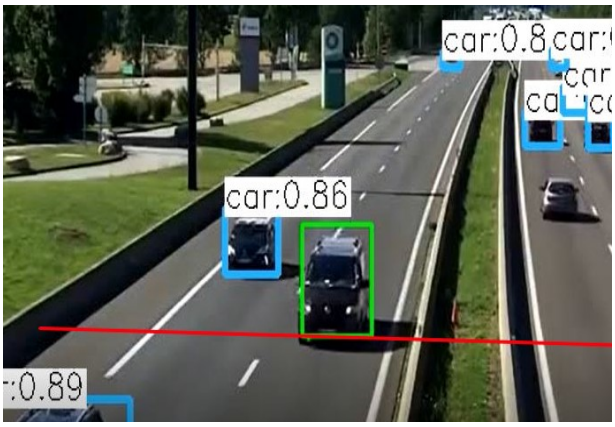


Fig.4 Object being added to the tracked objects list

To make sure that any counted object is counted once, the system will track objects that have been counted until they get far the crossing line by the offset α. The intersection over union (IoU) defined by equation [1] between the YOLO bounding box and the KCF tracker bound box is used to find if the system counted an object or not. If the IoU between two objects is greater than a margin μ (0.8 in our system), then the system will ignore this object, and it will not count it. The main motivation for this architecture is the fact that an object might stand within a distance less than the threshold for more than one frame due to congestion or other reasons, which will result in counting this object multiple times. We have to make sure that any counted object is defined to the system in a way that the system will be aware that this object is counted. The only possible way is to track every counted object until it become far from the crossing line by the offset α safe enough to conclude that this object has gone. Fig 5 shows an object that the system is tracking. The green box is for the KCF, and the blue box is for YOLO. In this case, the IoU is more than 0.9.

$$IoU = \frac{\textbf{area of overlap}}{\textbf{Area of union}} \qquad (1)$$

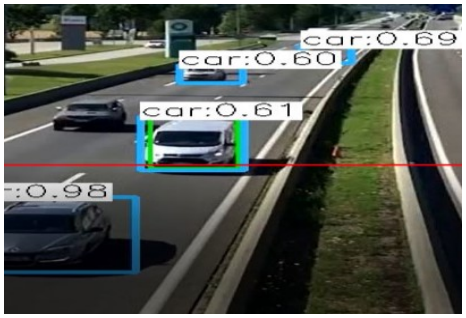


Fig. 5 Intersection over Union

The proposed framework has the advantage over other counting by tracking algorithms because the tracking is done for short period of time. Instead of tracking the object from the point it enters the frame to the point it leaves the frame, our system is tracking the object only from the moment it crosses the crossing line to the moment the object get far from the crossing line by the predefined offset. The proposed algorithm is more robust to tracking failure compare to other algorithms that track the object through the whole frame since the probability of tracking failure is less when the tracking region is small. Most of the cases, the system tracks any given object for 10 frames at most with approximately no change in the background. Recall that we have mentioned that the position of the crossing line affects the accuracy; therefore, it is better to choose the position of the crossing line such that there is no change in the background. Furthermore, the proposed system has less computational complexity as it tracks few objects at the same time; the system is continuously removing old objects from the tracked objects list. Finally, as the system is continuously removing objects from the tracked objects list, the KCF tracker will have less templates to search for, imagine the difference between the KCF tracker searching for two objects (templates) with the KCF tracker searching for 10 objects (templates). In addition, the search area is limited to area around the crossing line rather than the whole frame. Thus, the accuracy of tracking is improved. The pseudocode of the proposed object-counting algorithm is shown below

Algorithm 1: Objects Counting Algorithm

```
Input:
    Image Frames Sequence (N size),
    Crossing line position in the frame
Output
    Number of objects in all frames (Whole Video)

Initialize
    tracked_Objects_List to empty list
    Total_Count => 0, Threshold => 10, Offset => 20, IoU_Min => 0.7

For frame in the frames sequence do
    Detected_Objects => Pass frame to YOLO Object Detection

   If tracked_Objects_List is not empty
        Update tracked objects status using KCF tracker
        Remove any tracked object at distance >= Threshold + Offset
    End If

    For object in Detected_Objects do
        Distance => Calculate distance between object and crossing line
        If Distance < Threshold
            IoU => Intersection over Union with all tracked objects
            If no iou in IoU > IoU_Min
                add object to tracked_Objects_List
                Total_Count = Total_Count + 1
            End If
        End If

    End For
End For

Return Total_Count
```

III. Datasets Preparation

To test the proposed algorithm for object counting, we have prepared a dataset that consist of 20 videos from different sources. We have chosen the videos so that we have a diversified set of environments and scenarios. The four main parameters chosen to reflect this diversity are, object density, image quality, angle of view, and speed of motion.

A. Objects Density

Since object counting algorithms performance change with the number of objects in the frame, the density of objects is one of the key factors to validate the proposed algorithm. The dataset has several videos that have high objects density, medium (normal) objects density, and low objects density. The YOLO performance degrades slightly with the scenes that have large number of objects, as YOLO will miss some of the objects. Moreover, KCF tracker will have difficult times finding the tracked objects as the number of similar objects increases with high-density scenes. The low objects density videos have been chosen to compare the performance with high-density frames.

B. Image Quality

The proposed object-counting algorithm depends on YOLO for object detection as well as KCF tracker for object tracking, therefore, the proposed algorithm performance will degrade with low-resolution images. To find out the reduction in performance compared to high-resolution images, the dataset has high resolution, medium resolution, and low-resolution videos. Moreover, the dataset almost has different videos from different times of the day starting from the early morning to midnight.

C. Image View Angle and Road Shape

The third parameters is view angle that considers the angle between the camera and the tracked objects. This parameter is important because the ability to detect objects using YOLO varies with the angle that YOLO perceives the object. Moreover, since the primary objective of the proposed algorithm is to count cars, view angle that are not facing the road directly might miss some objects because of the blind spot that occur due to the tilt in the camera view as well be shown in the next section

D. Speed of Motion

Finally, the speed of motion is considered too when preparing the dataset. As the speed of motion is indirectly affecting the object density. If objects are moving with fast speed, the probability for congestion is low. In addition, the number of frames for tracking the objects that crosses the crossing line is less, as the object will get far from the crossing line within few frames. On the other side, slow motion is a real test for the proposed algorithm as it test the tracking ability, which means that the ability of the KCF tracker to track objects for long time is challenged. However, this might not be the case as objects moving slowly might be easier to track, as there is no huge change in the background. The results are shown in the next section.

IV. Performance Evaluation

This section shows the performance evaluation results and the effect of each of the predefined parameters on the accuracy of the algorithm. The reasons for the accuracy value after the tests are discussed, the full results are shown in the next page.

A. Object Density

Based on the test results summarized in the next page, the object density is not a key factor affecting the accuracy of the proposed algorithm. The main reason in our opinion is the good choice of the position of the crossing line in the initial setup. The proposed algorithm is detecting objects in the whole frame, but it tracks the counted objects for predefined region only. Therefore, whether the video frame has large number of objects or not, the algorithm is ignoring all objects except the objects in the search region. Well, the number of the objects in the search region is at most equal to number of lanes the road has as the algorithm is continuously updating the tracked objects list and removing old objects. In rare cases, YOLO might detect one object as two objects as shown in the left image Fig 6. However, this is an object detection problem, and cannot be linked to high object density environment.

It is worth to point out to the fact that it is true that in high-density environments, the proposed algorithm will miss more objects compared to low density environments, but the percentage of error remains almost the same. In the dataset, video number 14 and 20 reflect the low object density environments, the accuracy is higher compared to video number 3, 5, 7, and video number 16 which reflect the high object density environments. However, the change in the accuracy is small and the overall accuracy is more than 90%, which is the target of the proposed algorithm.

B. Image Quality

The image quality or the frame resolution is the main cause of performance degradation and higher error rates. Since the proposed algorithm depends on YOLO object detection and KCF tracker at the same time. The low-resolution frames affect the YOLO and KCF tracker at the same time. Hence, the accuracy of the proposed algorithm will degrade. These low-resolution frames cause the YOLO to miss more objects, and based on the architecture we proposed, if YOLO is not detecting the object, the algorithm will never be able to start tracking it, nor counting it. Furthermore, if the YOLO can detect these objects in low-resolution environments, the KCF tracker failure is much more compared to high or even medium resolution frames. Consequently, if YOLO is detecting and KCF tracker has more failure rates, the total count will be more than the ground truth as shown in the right image in Fig 6 since every frame YOLO is detecting an object with no tracking bounding box, the IoU is 0 and this object will be counted again and again. So, low image quality affect the accuracy by two means, either YOLO is not detecting in the first place, which results in total count value less than the ground truth, or by YOLO being able to detect, but the KCF tracker failing to track which results in total count value more than the ground truth. Video number 8 shows a low-resolution video, the accuracy result is the worst in the whole dataset.

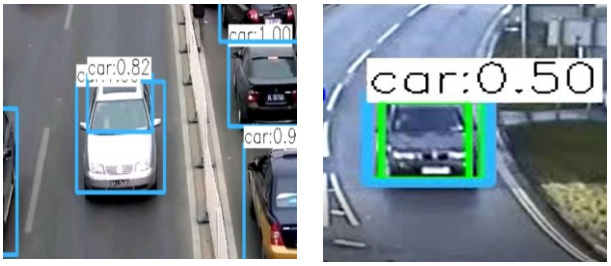


Fig. 6 Object density (Left) and Image Quality (right)

C. Camera View Angle

The view angle is for sure one of the most important factors that determine the accuracy of the proposed algorithm. Since a slightly tilted view as shown in the left image in Fig 7 might results in missing the detection of the object because of the blind spot. The blind spot occurs when a large object is moving in the center, and blocking some parts of the objects in the most right or most left side. Of Course, the large object will not block smaller objects from being scene in the camera, but the remaining parts that are visible to the camera are not enough for the YOLO object detection to detect the class of the object. Therefore, we conclude that the camera view affects the accuracy, but depending on the number of large objects that might cause the blind spot problem. The left image in Fig 7 shows a showcase of the blind spot problem. Video number 2 and video number 3 shows the same environment at different times of the day. The first video has more large objects than the second one, which explains the difference in accuracy value even though the environment has not changed.

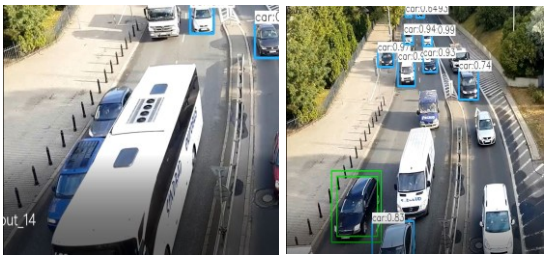
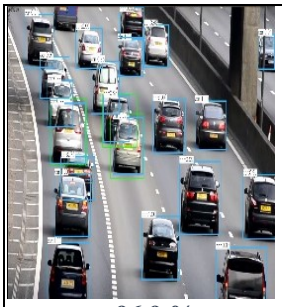
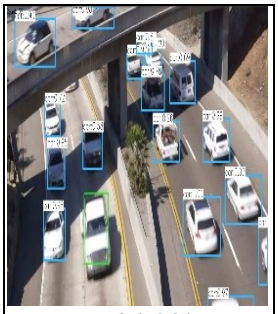
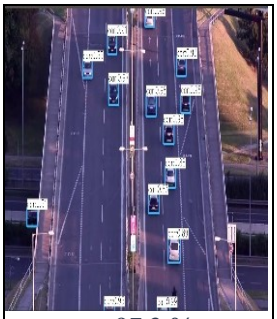
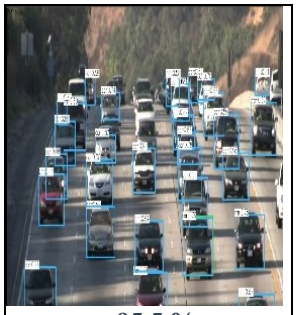
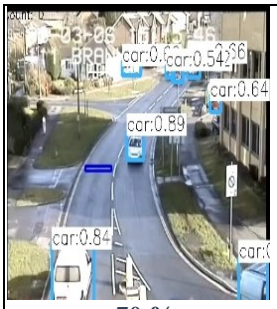
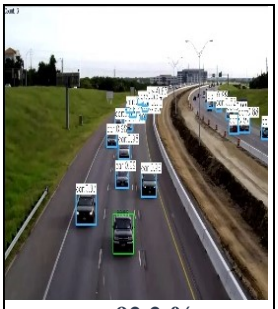
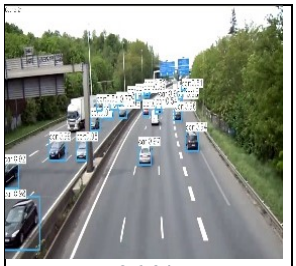
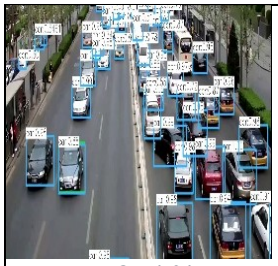
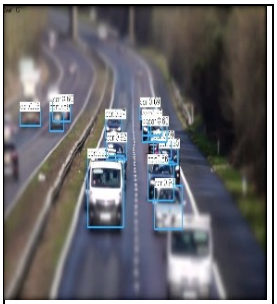
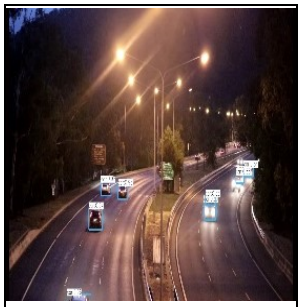


Fig. 7 View angle of the camera

D. Speed and Direction of Motion

Finally, the speed of motion. Based on the results from the performance evaluation, we can conclude that high-speed object motion in general has no effect on the accuracy of the proposed algorithm. Video number 5, and video number 16 reflect high-speed environments, the accuracy is above 90% as expected, and have one of the best accuracy values. However, high object density with low speed environments resembled in video number 13, and video number 19 have one of the worst accuracy results. The main reason for that is the increased rate of tracking failure in the KCF tracker, as objects are move slowly and existence of large number of similar objects in the frame. The KCF should track each counted object in the tracked objects list for longer time compare to high-speed environments. If tracking failed at one frame, YOLO detection of the object will result in the creation of a new tracked object, then, if the KCF tracker retain the old object again, we will have two tracking bounding boxes as shown the right image in Fig 7.

Normal traffic with trucks and perfect image view	Curved road with normal trafic and more trucks	Curved road with high trafic density and no trucks	Normal car density with buses and perfect image view	High car density with high speed and curved road
 94.3 %	 93.6 %	 96.9 %	 94.2 %	 96.6 %
Aerial image view with traffic density	High car density with perfect image view	Curved road with low image rosultion	Curved road with normal traffic and blind spot	Normal traffic and perfect image view
 97.3 %	 95.5 %	 70 %	 83.7 %	 92.2 %
Normal traffic with trucks and blind spot	Normal traffic with outgoing cars and perfect image view	High traffic density with low speed	Low traffic density and perfect image view	Normal traffic and perfect image view
 97.8 %	 96 %	 85.4 %	 97.2 %	 92.3 %
High traffic density and speed with occlusion and perfect image view	Normal traffic with trucks and blind spot	Normal traffic at night	High traffic with distraction and noise and low speed	Extra Low traffic and perfect image view
 93.8 %	 92.3 %	 97%	 85.7 %	 98 .1%

CONCLUSION

This paper presented an innovative efficient object counting system. The system general architecture is explained. Then, the challenges that we encountered using YOLO and KCF are addressed. Finally, the system performance is evaluated using 20 different videos with different characteristics. The performance evaluation showed promising results in terms of accuracy, and speed.

ACKNOWLEDGMENT

This research was supported by the Research Office at the United Arab Emirates University (grant number G00003133) and the Roadway, Transportation, and Traffic Safety Research Center (RTTSRC) of the United Arab Emirates University (grant number 31R116).

REFERANCES

[1] K., Z. and R., M. (2018). A Review: Object Detection using Deep Learning. International Journal of Computer Applications, 180(29), pp.46-48.

[2] Anusha, C. and S., P. (2018). Object Detection using Deep Learning. International Journal of Computer Applications, 182(32), pp.18-22.

[3] Pathak, A., Pandey, M. and Rautaray, S. (2018). Application of Deep Learning for Object Detection. Procedia Computer Science, 132, pp.1706-1717.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proceedings of CVPR,2016, pp. 779–788.

[5] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 6517–6525. IEEE, 2017. 1, 2, 3

[6] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv, 2018. 4

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37.Springer, 2016. 3

[8] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg.Dssd: Deconvolutional single shots detector. arXiv preprintarXiv:1701.06659, 2017. 3

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Proceedings of NIPS, 2015, pp. 91–99.

[10] K. He, G. Gkioxari, P. Doll´ar, and R. Girshick, "Mask R-CNN," in Proceedings of ICCV, 2017, pp. 2961–2969.

[11] Jaya Sunkara, M Santhosh, Suresh Cherukuri, and L. Gopi Krishna, "Object Tracking Techniques and Performance Measures – A Conceptual Survey" IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI-2017)

[12] D. Zhang, H. Maei, X. Wang, Y. Wang, "Deep reinforcement learning for visual object tracking in videos", Comput. Res. Repository, 2017.

[13] M. George, B. Jose and J. Mathew, "Performance Evaluation of KCF based Trackers using VOT Dataset", Procedia Computer Science, vol. 125, pp. 560-567, 2018. Available: 10.1016/j.procs.2017.12.072.

[14] Benny Hardjono , Hendra Tjahyadi, "Vehicle Counting Quantitative Comparison Using Background Subtraction, Viola Jones and Deep Learning Methods", 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).

[15] Jia-Ping Lin, Min-Te Sun, "A YOLO-Based Traffic Counting System", 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI).

[16] C S Asha ; A V Narasimhadhan, "Vehicle Counting for Traffic Management System using YOLO and Correlation Filter", 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT).

[17] Alejandro Forero ; Francisco Calderon, "Vehicle and pedestrian video-tracking with classification based on deep convolutional neural networks", 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA).

[18] Mohamed A. Abdelwahab, "Accurate Vehicle Counting Approach Based on Deep Neural Networks", 2019 International Conference on Innovative Trends in Computer Engineering (ITCE).