Journal of Engineering Science and Technology
Vol. 16, No. 1 (2021) 194 - 208

REAL-TIME MOVING OBJECTS DETECTION AND TRACKING USING DEEP-STREAM TECHNOLOGY

NUHA H. ABDULGHAFOOR, HADEEL N. ABDULLAH*

Department of Electrical Engineering, University of Technology,
Street No. 52, Al-Wehda Neighbourhood, Baghdad, Iraq
*Corresponding Author: 30002@uotechnology.edu.iq

Abstract

Recently, the deep learning strategy has outperformed the rest of the algorithms related to object detection and tracking in terms of performance and efficiency. It has demonstrated the adaptation of many scientific and practical applications to artificial intelligence (AI) systems. The aim of this paper is primarily to build and develop a real-time object detection and tracking algorithm embedded in an AI computing device known as Nvidia Jetson TX2. It improves the performance of the proposed algorithm. It brings its work closer to reality. DeepStream-Software Development Kit (DS-SDK) was used to achieve high performance and interact with multiple video sources at the same time as well. Many convolutional neural networks were used inside the proposed algorithm, such as those based on Fast Region-Convolution Neural Network (Fast R-CNN), Single Shot Detector (SSD), and You Only Look Once (YOLO) network. Its performance in treating different video clips with deep streams of piping compared. The experimental results showed the excellent accuracy and speed of the proposed algorithm in achieving the desired goal.

Keywords: CNN, Deep learning, DeepStream, Object detection, Object tracking.

1. Introduction

During the previous decades, the development of image processing techniques and video sequencing analysis to include a wide range of applications, computer vision, most of these developments are due to the use of artificial intelligence systems in general and the way of deep learning in particular [1]. In recent years, many active Convolution Neural Networks (CNNs) have been proposed for various applications. Such as Computer Vision. Now, these networks have smart software systems that make them suitable for many applications in real-time such as monitoring systems. The process of object detection and tracking, and automated navigation and monitoring systems present an essential challenge in real-time [2]. In this paper, an algorithm to detect and track objects in a complex dynamic video environment has proposed.

Figure 1 illustrates the structure of object detection and tracking through video sequencing. First, the video clip is captured by a camera, for example, and then inserted into the system. These clips may need to be pre-treated to get rid of the noise that may accompany the video sequence capture, as well as the operations of resizing the video frame to the required size—the representations of the video frames in the form of matrices to facilitate their mathematical handling. Then the object detection stage represents the identification of objects and their location within the video frame using, for example, deep learning algorithms. Where you need a data set divided into two parts, the first part is for training on the proposed neural network, and the other is for the network test to detect organisms. CNN's have gained the ability to discover and locate living organisms within the video frame. Where objects are selected, they are included in the surrounding boxes, which are the primary reference for objects tracking in frames, the object tracking stage has moved to find the path for each object and follow it through subsequent video frames.

Finally, there are several measurement parameters and metrics to determine the efficiency and ability to perform the algorithm in achieving the required. Many of the deep learning templates or models that are difficult to apply to small programming devices or platforms have been developed [3]. There are still some restrictions in the case of bottlenecks in input and output pipelines, which in turn may cause complexity when dealing with several streams of inputs and outputs. To improve performance and increase its efficiency, we suggest a new algorithm that provides seamless methods from start to finish. It can also convert original flow data into executable applications, as it consists of a DeepStream-based framework. Usually, the input data flow is composed of continuous and multiple video sequences, and the outputs are actionable visions of the test model or some algorithm analysis. The main contribution of this work is:

- The implement detection and tracking system for multiple objects (such as cars, pedestrians, etc.) in real-time monitoring system applications.
- Utilize DS-SDK technology to improve the performance of the algorithm to make it able to process more than 30 serial videos simultaneously. And then extract useful information with high speed and accuracy.
- Use a broad dataset from different environments to test the performance of the proposed algorithm.

The research document organized as follows: The second section explains similar works, such as research, articles, and others. The third section describes the basic principles and methodology of this article. The fourth section shows the structure of the proposed algorithm and all details of implementation. Section 5 presents laboratory experiments, simulation results, and analytical comparison for all the networks used. As for the sixth and final section, it clarifies the research findings and the future vision for developing the work.

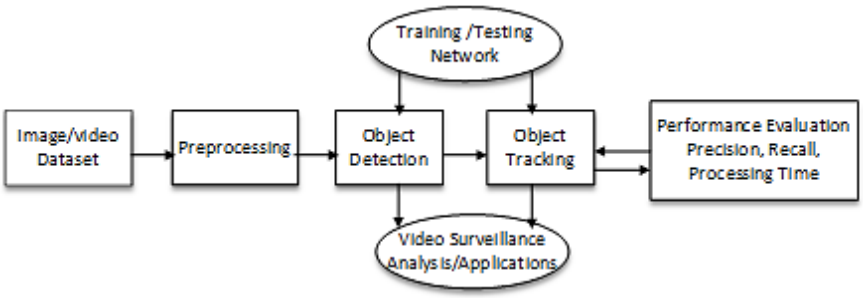

Fig. 1. Object detection and tracking framework [2].

2. Related Work

Several detection algorithms have emerged from the structure of the bypass neural networks, the most famous of which is the Region of Convolution Neural Network (R-CNN) [4], which marks the beginning of the application of these networks in the detection of objects. After that, new systems developed or proposed that had an exceptional performance in detecting objects such as the SSD algorithm [5] and YOLO [6]. Improvements are made to the core networks, such as adding features and a better structural choice such as MobileNet [7]. New versions of these networks have appeared, such as Fast-RCNN [8], YOLO version 2 (YOLOV2) [9], and others. Recently. Widely used datasets are available for training and testing these networks. Object detection or tracking algorithms used on similar embedded devices presented research work based on traffic monitoring, vehicle classification, pedestrians, and traffic lights. The proposed method showed intermediate resolution results and accurate classification of all objects in the video files captured from fixed cameras. Redmon et al. [10] The YOLOV3 and the small YOLOV2 compare a few other classic things in terms of speed and accuracy on the Jetson TX2 platform on a drone.

Howard et al. [11] suggested a compact multi-purpose real-time tracker based on the front-backlight (MobileNets) detector. It has accuracy trade-offs and shows strong performance with large size and latency. Tijtgat et al. [12] proposed an approach by using the drone's automatic learning algorithm to detect and track objects in real-time using the integrated camera or low-power computer system. The predecessors of TX2, TX1, and TK1 are evaluated, where they combine a small version of Faster-RCNN with Kernelized Correlation Filters (KCF) tracker to track one object on a drone. This article shows an accurate detection algorithm that is slow and mathematically expensive. While tracking algorithms are very fast but not careful, especially when there are fast-moving objects. Raj et al. [13] performed detection by using an SSD detector and made a comparison with other network

models like Fast R-CNN and YOLO and knowing the strengths and weaknesses of each system. It shows good accuracy and a high apparent positive rate and a minor false positive. It has drawbacks in Bad weather, noise, and non-standard vehicle.

Seidaliyeva et al. [14] proposed an installed algorithm done. It has a low computing power system and weight constraints. It shows accurate detection but slow and computationally expensive. Then, fast-tracking with less accuracy. Artamonov and Yakimov [15] provided a new method to classify traffic signals using a convolutional neural network called a YOLO implemented on a mobile platform using NVIDIA Jetson where freedom of movement appears, in addition to excellent accuracy and speed in the classification of traffic signals. The proposed algorithm drawbacks in the dark, bad weather, and appearance noise. Blanco-Filgueira et al. [16] implemented a visual tracing of multiple objects based on deep real-time learning using the NVIDIA Jetson TX2 development device. It has a built-in camera and wireless connectivity and is battery-powered for both mobile and external applications. The results highlighted the effectiveness of the algorithm under real challenging scenarios in different environmental conditions, such as low light and high contrast in the tracking phase and not consider in the detection phase.

Hossain and Lee [17] proposed an application based on a deep learning technique, implemented on a computer system integrated with a drone, to track the objects in real-time. Experiments with the proposed algorithms demonstrated good real-time efficacy using a multi-rotor aircraft. It counted a target of similar features. But it lost track of a computed function and considered it as a new target. Vadersteegen et al. [18] suggested a strategy to achieve object detection accuracy based on a multi-data algorithm embedded on a drone. Remote or invisible objects are detected, taking into consideration different heights and angles of shooting. The results showed that the proposed strategy performed well using two different Nvidia platforms (Jetson TX2 and Xavier) as the results showed that using this platform provides a straightforward methodology for discovering and tracking algorithms in addition to saving energy consumed.

3. Methodologies

DeepStream technology has many advantages, including it works in an ideal and integrated way to handle input and output processing. In addition to the possibility of working more than a deep learning algorithm to process data independently and asynchronously, which leads to increased performance efficiency and productivity without the need to manually manage the processing system. The most important task for some supported applications such as semantic fragmentation, object detection, classification, or object tracking operations.

3.1. Fast region convolution neural network (Fast-RCNN)

It is the developed version of the R-CNN network proposed by Ross Gershik and others [4]. It is one of the most popular systems in the field of object detection and classification. It consists of a group of convoluted neural networks, as shown in Fig. 2. It consists of two Region Proposal Networks (RPN) to generate several proposed regions that are expected to contain objects within the video frame. The main difference to a Fast R-CNN check over its predecessor is the presence of a selective search of these proposed areas rather than a full-frame scan. It is thus reducing the mathematical and

time cost of detection as much smaller regions are created. The RPN then sets squares surrounding areas that can contain objects, usually called anchors [12].

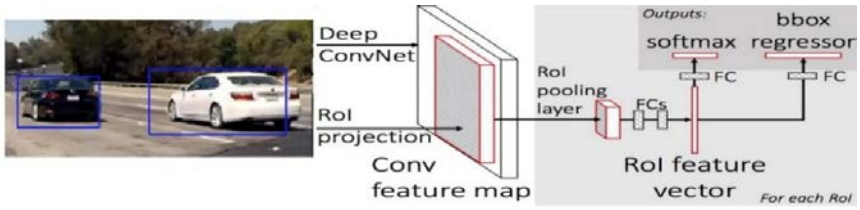

Fig. 2. The framework of Fast R-CNN [4].

3.2. Single shot detector (SSD)

SSD [5] is very common for having a valid number of frames per second (FPS) using lower resolution images at an acceptable cost of resolution. SSD is one of the few methods that constitute detection as a regression problem. Whereas SSD manages a convolutional network on the input image only once and calculates the feature map. Later on, we ran a 3x3 small convolutional nucleus on this feature map to predict surrounding squares and possible classification [13]. SSD also uses square bindings with different widths and learns to scale rather than getting the box. To manipulate the scale, the SSD expects the surrounding squares after multiple convolutional layers. Since each convolutional layer operates on a different size, it can discover objects of various sizes, as shown in Fig. 3.

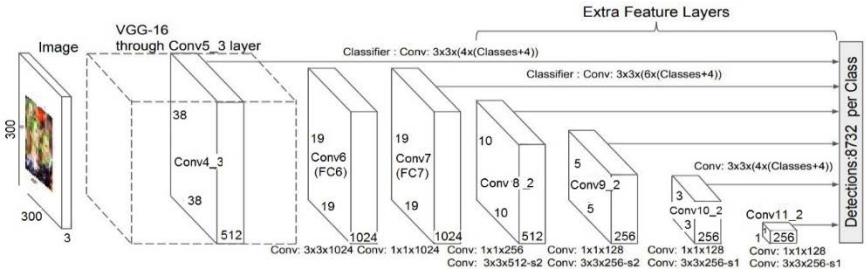

Fig. 3. The framework of SSD network [5].

The above graph is an example of an SSD with the Visual Geometry Group (VGG-16) as a feature extractor network model. VGG16 is a CNN model where the model has achieved high test accuracy, this network is improved by replacing large core size filters with multiple 3x3 filters one core after another. The SSD model adds many feature layers to the end of the underlying grid, which predicts offsets to default squares of different scales and widths and associated confidence.

3.3. You only lock once (YOLO)

For the YOLO frame [6], the detection is a direct slope that takes an input image and identifies the separation capabilities along with the bounding box coordinates. YOLO divides each image into a grid of S x S, and each network N predicts the bounding squares and confidence values for each, as shown in Fig. 4. Confidence reflects the accuracy of the bounding box and whether the bounding box contains

an object despite the specified category. YOLO also predicts a rating score per box per category [9]. You can combine both classes by working on the opportunity to attend each chapter in an expected square. So, the sum of the funds expected to be SxSxN. On the other hand, most of these funds have lower trusts, and if we set a threshold (for example, 30% confidence), we can get rid of most of them. It is not a good idea to compare the results of different detectors directly from their papers to get a clear view of the best. Many factors affect it, such as various basic feature snippets (for example, VGG, MobileNet, Residual neural Network (ResNet), and Inception) [7], different default image resolutions, as well as different hardware and software platforms. We mostly decide the appropriate trade-off between speed and accuracy based on our requirements.

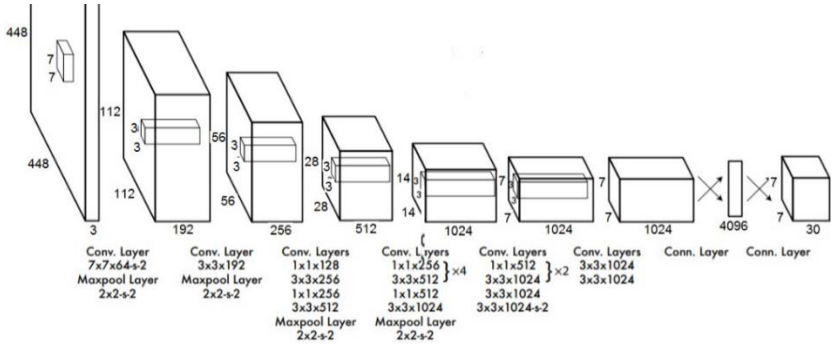

Fig. 4. The framework of YOLO network [8].

The main difference between these network models is that YOLO only makes predictions for one feature map, while SSD combines forecasts across multiple feature maps of different sizes. Also, SSD is a better option than YOLO as we can play it on video, and barter accuracy is very modest. While handling large volumes, SSD appears to work fine, but when we look at precision when the object size is small, performance decreases slightly. Although YOLO is faster, it loses relatively in its marks. Area-based detectors such as Faster R-CNN [12] (which use RPNs to create boundary boxes and object classification) demonstrate a small resolution feature if real-time speed is not needed (running at seven frames per second only). But in our case, the restrictions have to be met in real-time, so let's stick to the hard drives.

3.4. DeepStream software development kit (DS-SDK)

DeepStream SDK [19] is a complete suite of flow analysis tools for AI-based video and image processing. And the use of many sensors and data processing at the same time. DS-SDK features hardware acceleration building blocks, called add-ons that bring deep neural networks and other complex processing tasks into the stream processing line. This article shows the use of DS-SDK in the app to handle multiple flows and multiple systems of inference. The application is connected to real-time object detection and tracking runtime [20]. In addition to dealing with a lot of video streaming of surveillance systems, object detection, and tracking application. DS-SDK has been applied to platforms that include JetPack software as an operating system and also includes several additional components supporting high-resolution image and video processing applications. These include Compute Unified Device Architecture (CUDA), TensorRT, Multimedia Application Programming Interface

(APIs,) and more, as well as many libraries. It has been developed using a business strategy called Gstreamer that enables researchers and programmers to find quick and flexible applications that can transform large-size video into useful, flexible insights into processing. DS-SDK [20] simplifies the development of scalable, intelligent video, and Developers can now use this to build new applications to transform video quickly for different computer vision applications.

4. The Proposed System Architecture

In recent decades, artificial intelligence systems have entered into all areas of beading, including computer vision applications. It includes image processing and digital videos to infer some understanding of the data in these images and videos, for example, the detection and tracking of objects, which describes a smart system for determining the presence and location of an object or objects within an image. In this article, a DeepStream technology-based proposed algorithm is designed and implemented to create a real-time object detection and tracking system and test its features for various models of deep learning, as shown in Fig. 5. Many CNN models such as the Fast R-CNN, SSD, and YOLO network have been used on artificial intelligence (AI) computing algorithms. These models have been trained and tested as object detectors on a different data set, as these networks operate with DeepStream technology and other models nvosd and nvvidconv, and other supporting parameters in building the proposed algorithm as shown in Fig. 6. The DS-SDK-based algorithm achieved high throughput for applications, which includes detection, tracking, and categorizing objects. So, we design and implement an object detection and tracking system application across NVIDIA platforms that have many features like a large number of video streams processed on one platform at the same time. The NVidia Jetson TX2 [21] platform has been used to create a practical application to detect and track objects in real-time, as shown in Fig. 7.

All significant processing is done in the processing platform, which is NVIDIA Jetson TX2, as shown in Fig. 6. It represents a miniature computer that has a powerful processing unit and fast performance, especially in applications of artificial intelligence, also that it has a graphics processing unit (GPU) [22], type NVIDIA PascalTM 256 Compute Unified Device Architecture (CUDA) Core, which makes it suitable for to that it works at low capacity. From technical specifications for this platform [21], we noted applications based on the neural network. It is also supported by modern software, including NVidia Jetpack and SDK development kit, including comprehensive libraries and deep-learning software and GPUs and multimedia. DS-SDK is an easy-to-use technology that can build an ideal deep-learning model that is compatible with all platforms. It is done by defining specific parameters in a text frame that is easy to understand and apply.

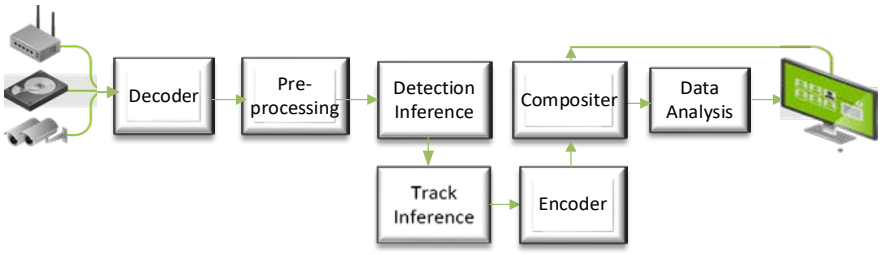

Fig. 5. The end to end of artificial intelligent object detection and tracking.

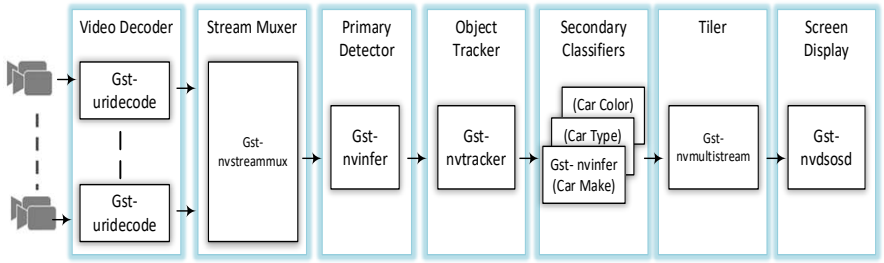

Fig. 6. The general building block of proposed algorithm based on DS-SDK.

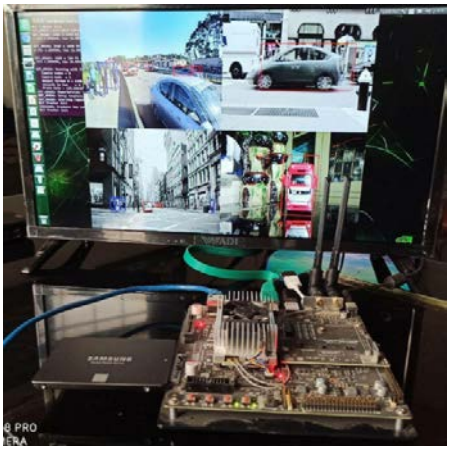

Fig. 7. The hardware implementation of object detection and tracking system.

The model has a relatively lower configuration compared to the computers used, and it is also small and lightweight that can perform in real-time with reasonably good accuracy and software and technology researchers. Nvidia Jetson TX2 launched with Ubuntu 18.04 and its board packages along with several modules like OpenCV, TensorFlow, CUDA Toolkit, cuDNN, and many other computer vision dependencies. The video frames are received using OpenCV and added to the input queue. Subsequent chains that run the detection mechanism run concurrently, add bounding boxes to the detected objects in frames, and add them to the output queue. Then, the tracking stage is based on the Kanade – Lucas – Tomasi (KLT) [23] as a feature tracker for proper synchronization and to complete the proposed essential algorithm task. Several equations can be used to develop a proposed algorithm, the grid model predicts the box updates, which are (x_p, y_p, w_p, h_p). Then the object displacement is made from the upper-left side of the image by the amount of *xi*, *yi* with prior knowledge of the height and width of the bounding box, as shown in Fig. 8. Therefore, the new dimensions are calculated, as shown below:

$$x_n = \sigma(x_p) + x_i \tag{1}$$

$$y_n = \sigma(y_p) + y_i \tag{2}$$

$$w_n = w_b e^{w_p} \tag{3}$$

$$h_n = h_b e^{h_p} \tag{4}$$

The second critical factor is the loss function, which represents a mixture of the value of the confidence in the predicted degrees and the accuracy of the location. Where the sum of classification and confidence losses represents one measure, as shown in the equation below, it is also an indication of the extent of correlation between the hypothetical square and the ground truth square of an item.

$$L(x, y, bp, bg) = \frac{1}{N}(L_{con}(x, y) + \alpha L_{loc}(x, bp, bg) \quad (5)$$

where N equals the number of matching boxes, α represents the balance parameter between the two losses. The second term represents the soft L_1 loss between the shift value of prediction (b_p) and ground truth (b_g) boxes. x and y is the center of the box. Then,

$$L_{loc}(x, bp, bg) = \sum_{i \in Pos(x,y,h,w)} \sum x_{ij} Smooth_{L1}(bp_i - bg_j) \quad (6)$$

where, $$Smooth_{L1}(k) = \begin{cases} 0.5k^2 & k > 0 \\ |k| - 0.5 & k < 0 \end{cases} \quad (7)$$

and

$$L_{con}(x, y) = -(\sum_i^N x_i \, log(y_i) + \sum_i^N Log(y_i)) \quad (8)$$

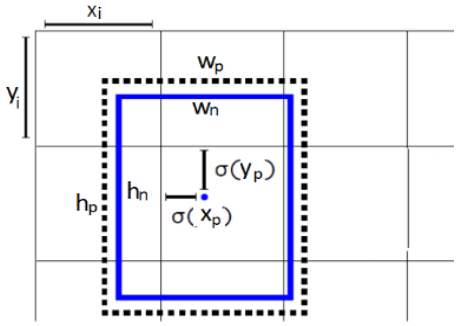

Fig. 8. The bounding box transformation between the predication and prior evaluation.

5. Experiments and Results

Based on our algorithm proposed above, a program was developed based on DS-SDK Technique and then implemented on the NVIDIA TX2 platform with Ubuntu 18.4 LTS operating system. The proposed algorithm has been tested on many different video data simultaneously, as shown in Fig. 9. It is primarily for traffic management systems, pedestrian tracking, and highway monitoring applications, as well as in sports applications. It shows that the proposed algorithm can be received the different video streams from multiple sources such as offline, online broadcasting, real-time monitoring source, and others. It was able to process more than 30 serial videos simultaneously. Then extract useful information with high smoothness, speed, and accuracy where good results were obtained compared to previous software systems which can process one video stream only.

Various deep learning network models have also been tried to test the effectiveness and efficiency of the proposed algorithm. To obtain the optimal performance for the proposed algorithm by choosing the efficient deep-learning

detector model. So that an SSD, Fast R-CNN, and YOLO object detector network was used, as shown in Fig. 10. The results obtained showed the SSD network's ability to detect objects with high accuracy. So, it can also be used in real-time applications and low-resolution video data. Then the Fast R-CNN architecture also chose as examples of region-based detection. The arithmetic time for SSD is higher than YOLO algorithms. Table 1 summarizes the comparative results for using different deep learning models. We test the efficiency of the proposed algorithm in different circumstances and challenges; it was tested on various datasets that include different environments, whether internal or external scenes, as shown in Fig. 11.

Fig. 9. The experimental result of the proposed algorithm for different applications.

Fig. 10. The experimental results using different deep-learning network model models: (a) SSD, (b) Fast R-CNN, and (c) YOLO.

Table 1. Performance comparison result for different Deep-learning network models.

Framework	Proposal	Backbone	Clarification	benefits	Limitations
Fast R-CNN	Selective Search - RPN	ResNetV2	No	More Accurate	Slow, high computation
YOLOV3	Joint Grid regression	DarkNet35	Yes	Less Accurate	Fast processing
SSD	RPN regression	VGG16	Yes	Good Accurate	Mid fast processing

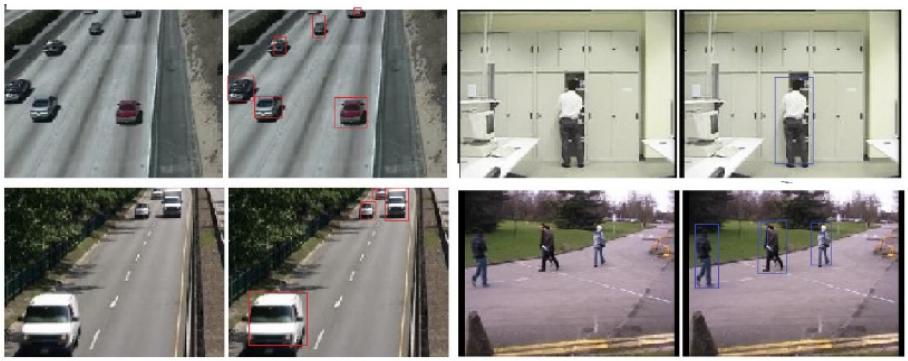

Fig. 11. Visual experimental results using different datasets in the internal and external environment.

Then, these improved by using the CPU and GPU in proposed algorithm testing, which in turn leads to a reduction in the computational load, as shown in Fig. 12. By computing the time processing for each detector models in terms of Frame per Second (fps). Table 2 shows the accuracy of the proposed algorithm in the detection and classification of various objects. It can be seen that SSD model usage is reasonably fast, along with good detection accuracy for real-time scenes. While Table 3 shows the results (mean average precision) by using the benchmark dataset that compares SSD, Fast R-CNN, and YOLO. This parameter [3] depends on other metrics such as Recall, Precision, etc. The results of the high efficiency were relative to the various data compared to the different models. It showed the floating model on the Fast R-CNN is the most accurate. However, it has the disadvantage of being slow, so it is not reliable in real-time applications. As for YOLO, it has a high processing speed at the expense of detection. We also note that SSD performs worse for small or deformed objects. A Split model speed breakout that divides the process into GPU and CPU sessions [22] operates in a single thread that visualizes the end with frames that come from the CPU factor or GPU. Software additions work only for this model but lead to a significant increase in performance. Multiple indicators were added to support the divided sessions. Model options were added to increase GPU memory allocation in the tension of the tensioner to support the model. Current Rate: 30 to 32 fps (no speed improvement visualization).

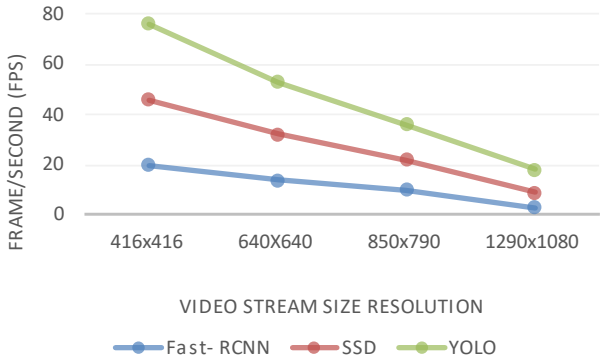

Fig. 12. Time processing for proposed models.

Table 2. Accuracy performance for models.

	Accuracy %		
Class Type	**Fast R-CNN**	**SSD**	**YOLO**
Single Human	70.1	74.2	65.67
Pedestrian	65.2	70.27	59.4
Single Vehicle	90	88	80
Multiple Vehicle	63	82.45	66.3
Small Vehicle	50.35	69.7	53.6
Large Vehicle	65.5	88.6	64.2

Table 3. Average precision evaluation performance.

Framework	Backbone	MS-Coco Datasets AP	Pascal-VOC07 Datasets AP	Pascal-VOC12 Datasets AP
Fast R-CNN	ResNetV2	45.32	70.34	67.2
YOLOV3	DarkNet35	56.76	78.10	73.18
SSD	VGG16	48.2	75.21	73.8

6. Conclusion

Object detection and tracking are severe challenges in real-time scenarios. The main objective of the proposed algorithm is to discover and track different objects in videos from various sources and then classify them. Where processing for all video sections and at the same time, which in turn leads to building an integrated security system that works in real-time environments. The results show that the combined algorithm gives useful detection and tracking for different things and applications such as video surveillance systems. The use of the DS-SDK increased capacity and improved the performance of the proposed algorithm. Data can be received and analysed from more than one source, with smooth flow and high smoothness. The use of a trained SSD network yield results with high confidence as it outperformed the rest of the models. However, the trained YOLO network has speed processing at the expense of accuracy. Finally, the proposed algorithm can be implemented in many fields, such as civil or military surveillance systems, the protection of public places, and government institutions, the medical and agricultural applications.

Nomenclatures

b_g	Ground Truth position of a bounding box
b_p	Predicated position of a bounding box
E	Exponential function
g	Intensity
h_k	Height of bounding box
I, J	Curves
L	Loss function
L_{con}	Confidence loss function
L_{loc}	Location loss function
Log	Logarithm function
N	The number of matching boxes
w_k	Width of a bounding box
w	domain
x	Centre coordinates
x_k	Centre horizontal position value
y_k	Centre vertical position value

Greek Symbols

α	Balance parameter between the losses.
σ	Variance function

Abbreviations

API	Application Programming Interface
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DS-SDK	DeepStream-Software Development Kit
Fast-RCNN	Fast Region Convolution Neural Network
FPS	Frames Per Second
KCF	Kernelized Correlation Filters
KLT	Kanade – Lucas – Tomasi
VGG	Visual Geometry Group
GPU	Graphical Processing Unit
ResNet	Residual neural Network
RPN	Region Proposal Networks
SSD	Single Shot Detector
YOLO	You Only Look Once

References

1. Hemanth, D.J.; and Estrela, V.V. (Eds.) (2017). *Deep learning for image processing applications*. 31, IOS Press.
2. Abdullah, H.N.; and Abdulghafoor, N.H. (2020). Objects detection and tracking using fast principal component purist and Kalman filter. *International Journal of Electrical & Computer Engineering*, 10(1), 1317-1326.
3. Gad, A.F.; Gad, A.F.; and John, S. (2018). *Practical computer vision applications using deep learning with CNNs*. A press.

4. Ren, S.; He, K.; Girshick, R.; and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 91-99.
5. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C. Y.; and Berg, A. C. (2016). SSD: Single shot multibox detector. *European conference on computer vision*, Springer, Cham, 9905, 21-37.
6. Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*. Las Vegas, NV, 779-788.
7. Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fisher, L.; Wojna, Z.; Song, Y.; Guadarrama, S. and Murphy, K. (2017). Speed/accuracy tradeoffs for modern convolutional object detectors. *Proceedings of the IEEE conference on computer vision and pattern recognition*. Honolulu, HI, 7310-7311.
8. Chandan, G.; Jain, A.; and Jain, H. (2018). Real-time object detection and tracking using deep learning and OpenCV. *International Conference on Inventive Research in Computing Applications (ICIRCA)*. Coimbatore, India, 1305-1308.
9. Wang, S.; Ozcan, K.; and Sharma, A. (2017). Region-based deformable fully convolutional networks for multi-class object detection at signalized traffic intersections: NVIDIA AICity challenge 2017 Track 1. *IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People, and Smart City Innovation (SmartWorld/SCALCOM/ UIC/ATC/CBDCom/IOP/SCI)*. Francisco, CA, 1-4.
10. Redmon, J.; and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*. Retrieved December 20, 2019, from http://arxiv.org/abs/1804.02767v1.
11. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
12. Tijtgat, N.; Van Ranst, W.; Goedeme, T.; Volckaert, B.; and De Turck, F. (2017). Embedded real-time object detection for a UAV warning system. *Proceedings of the IEEE International Conference on Computer Vision Workshops*. Venice, Italy, 2110-2118.
13. Raj, M.; and Chandan, S. (2018). Real-time vehicle and pedestrian detection through SSD in Indian traffic conditions. *IEEE International Conference on Computing, Power, and Communication Technologies (GUCON)*. Uttar Pradesh, India, 439-444.
14. Seidaliyeva, U.; Akhmetov, D.; Ilipbayeva, L; and Matson, E.T. (2020). Real-Time and accurate drone detection in a video with a static background. *Sensors*, 20(14), 3856.
15. Artamonov, N.S.; and Yakimov, P.Y. (2018). towards real-time traffic sign recognition via YOLO on a mobile GPU. *Journal of Physics: Conference Series*, 1096(1), 012086.
16. Blanco-Filgueira, B.; García-Lesta, D.; Fernández-Sanjurjo, M.; Brea, V. M.; and López, P. (2019). Deep learning-based multiple object visual tracking on

embedded system for IoT and mobile edge computing applications. *IEEE Internet of Things Journal*, 6(3), 5423-5431.

17. Hossain, S.; and Lee, D.J. (2019). Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices. *Sensors*, 19(15), 3371.
18. Vandersteen, M.; Van Beeck, K.; and Goedemé, T. (2019). Super accurate low latency object detection on a surveillance UAV. *16th International Conference on Machine Vision Applications (MVA)*. Tokyo, Japan, 1-6.
19. Kaustubh, P. (2018). NVIDIA deepstream SDK 4.0.2 Release. Retrieved December 20, 2019, from https://docs.nvidia.com/metropolis/deepstream/dev-guide.
20. Han, S.; Shen, H.; Philipose, M.; Agarwal, S.; Wolman, A.; and Krishnamurthy, A. (2016). MCDNN: An approximation-based execution framework for deep stream processing under resource constraints. *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*. New York, US, 123-136.
21. Hamieh, I.; Myers, R.; Nimri, H.; Rahman, T.; Younan, A.; Sato, B.; El-Kadri, A.; Nissan, S.; and Tepe, K. (2020). *LiDAR and camera-based convolutional neural network detection for autonomous driving* (No. 2020-01-0136). SAE Technical Paper.
22. Colbert, M.; and Krivanek, J. (2007). GPU-based importance sampling. *GPU Gems, 3*, 459-476.
23. Hedborg, J.; Skoglund, J.; and Felsberg, M. (2007). KLT tracking implementation on the GPU. *SSBA, Swedish Symposium in Image Analysis*, Linköping, Sweden, 14-15.