Object Tracking Using Improved Deep_Sort_YOLOv3 Architecture

Article *in* ICIC Express Letters · October 2020

DOI: 10.24507/icicel.14.10.961

Linh Dang
Hanoi University of Science and Technology
18 PUBLICATIONS **40** CITATIONS
SEE PROFILE

Thang Cao
The University of Tokyo
46 PUBLICATIONS **141** CITATIONS
SEE PROFILE

ICIC Express Letters ICIC International ©2020 ISSN 1881-803X
Volume **14**, Number **10**, October **2020** pp. **961–969**

OBJECT TRACKING USING IMPROVED DEEP_SORT_YOLOV3 ARCHITECTURE

Tuan Linh Dang[1], Gia Tuyen Nguyen[2] and Thang Cao[2]

[1]School of Information and Communications Technology
Hanoi University of Science and Technology
No. 1, Dai Co Viet Road, Hanoi 100000, Vietnam
linhdt@soict.hust.edu.vn

[2]Machine Imagination Technology Corporation (MITECH)
3-7-87 Koyanagi, Fuchu, Tokyo 183-0013, Japan
{ tuyen; cao }@mitech.jp

Received March 2020; accepted June 2020

Abstract. *This paper proposes a new architecture for object tracking. This design is the improved version of the deep_sort_yolov3 architecture. The correlation tracker of the Dlib is also inserted into the proposed architecture to reduce the identity switches. In addition, the novel architecture is designed with a parallel approach to boost the operating speed. Experimental results with different videos showed that the proposed architecture obtained lower identity switches and higher operating speed compared to the conventional deep_sort_yolov3 approach.*

Keywords: Object detection, Object tracking, YOLO, DeepSORT, Dlib, Parallel processing

1. Introduction.

Nowadays, object tracking has been used widely in camera, car, shop, security, inspection, restaurant, and traffic. The goal of the object tracking is to locate the position of a moving object in one video frame.

Object detection is the first phase in any object tracking program. Computer vision applications need to detect objects before further processing these objects. The purpose of object detection is to determine whether any instance of objects is in an image frame, and the location where the instance is found. A famous method for object detection is the Viola-Jones framework [1]. A study has investigated the template matching technique to detect objects in different scales and variations [2]. Another approach uses characteristics of the objects such as contextual and geometrical knowledge for object detection [3]. Recently, the appearance of deep learning has created a new avenue for object detection [4]. One well-known approach is you only look once (YOLO) algorithm that detects the objects using a convolutional neural network (CNN). The advantage of the YOLO algorithm is the high processing speed. Also, with the appearance of YOLOv3, the YOLO detector obtained higher accuracy and speed compared with state-of-the-art approaches for the object detection [5].

After the object detection, the tracking operation is used to find the positions of the detected objects in each frame. The inputs of the tracking are the detected objects which are put inside bounding boxes. The tracking phase will track the detected objects until the end of the video stream. For example, if three people are detected, three separate bounding boxes are created, and these bounding boxes will be tracked in all subsequent video frames. Researchers have employed the mean-shift algorithm to track objects based on the distribution of the object features which can be represented as scale-invariant feature transform [6], speeded up robust features [7], binary robust independent elementary

DOI: 10.24507/icicel.14.10.961

features [8], histogram of oriented gradients [9]. If the distribution is changed because of the object movement, the tracking algorithm will look for a bounding box that has the closest distribution with the tracking object. Multiple object tracking is a challenging task compared to single object tracking. Previous studies have employed Kalmal filter for multiple object tracking [10, 11]. The particle filter algorithm was also investigated for efficient multiple object tracking with initialization and learning phase [13]. Nowadays, a well-known object tracking approach is the DeepSORT algorithm which combines the Kalman filter, the Mahalanobis distance, the Hungarian algorithm, and the appearance feature vector to track the objects [14].

The combination of YOLO object detection and DeepSORT object tracking has opened a new avenue for the research such as the deep_sort_yolov3 architecture [15, 16, 17]. However, this architecture is operated in a sequential way that may not obtain a high processing speed for multiple object tracking. In addition, the DeepSORT tracking cannot track the object if the YOLO cannot detect any bounding box of this object which leads to the degradation of the object tracking concerning the identity switches. In this situation, the missed object from YOLO will get a new identity when this object is detected again in subsequent frames. Therefore, it is necessary to have a new architecture to increase the operating speed and reduce the identity switches of the original deep_sort_yolov3.

The main contribution of this manuscript is to propose an improved version of the deep_sort_yolov3 architecture which could obtain a higher frame per second and lower identity switches than the traditional deep_sort_yolov3 approach.

This paper is presented as follows. Section 2 introduces the YOLO algorithm and the DeepSORT algorithm. Section 3 details our proposed architecture. Section 4 shows the experimental results. Section 5 concludes our manuscript.

2. **Literature Review.**

2.1. **YOLO.**

The YOLO algorithm, which uses a deep learning network for real-time object detection, has two tasks. The first task is to determine the location of an object, and the second task is to classify the detected object from the first task. The YOLO algorithm conducts both tasks with a single neural network only so that the YOLO can run at high speed. The input of the YOLO algorithm is image pixels, and the output of the YOLO algorithm is bounding boxes and class probabilities of these bounding boxes. The operation of the YOLO algorithm is described as below [5, 18].

- The input image is divided into $S \times S$ grid of cells. Each cell detects only one object by predicting B bounding boxes for this object. The cell also predicts C conditionals class probabilities where C is the number of classes. For example, if the YOLO algorithm is used to detect three classes of car, bike, bicycle, there are three conditionals class probabilities $P(car|object)$, $P(bike|object)$, and $P(bicycle|object)$.
- Each bounding box has five elements $(x, y, w, h, confidence)$ where *confidence* is the probability that the box contains an object, x and y are the coordinates of the bounding box, and w and h are the width and height of the bounding box.
- The *confidence* is multiplied with the conditional class probabilities to obtain the class-specific confidence score for each bounding box.

2.2. **DeepSORT.**

The Simple Online and Real-time Tracking (SORT) is the simple and effective multiple object tracking approach that uses the Kalman filter and the Hungarian algorithm to track an object [19]. However, the efficiency of the SORT algorithm will decrease with occlusions, different viewpoints of the camera. The authors of the DeepSORT algorithm introduced another distance metric based on the "appearance" of the object to improve the SORT algorithm. The DeepSORT algorithm has main components as follows [14].

- Track estimator, the original component of the SORT algorithm, uses the Kalman filter to predict the locations of the object bounding boxes. The prediction is based on previous object velocities. The track estimator is the location metric that uses intersection over union (IoU) distance between the detected bounding box and the predicted bounding box.
- Appearance descriptor is the new component of the DeepSORT algorithm. The appearance information is extracted using CNN so that the features from one object class are similar, and the features from different object classes are different in the feature space. The appearance descriptor is the appearance metric.
- Data association assigns the detected bounding boxes to the existing track of an object using location metric and appearance metric. Each existing track is an object identity (ID).
- Track handing controls the object track. If a new detected bounding box cannot be associated with any track at frame f, it will be put into a tentative track. The DeepSORT will try to associate the tentative track with other tracks in subsequent frames. If the association is successful, the track is updated. Otherwise, the tentative track will be removed.

3. Proposed Architecture.

3.1. Overview of the proposed architecture. The original deep_sort_yolov3 approach is presented in Figure 1. An input video is divided into different frames that are sent to a detector to detect objects. Each detected object will be serially tracked. For example, object 2 is only tracked after the tracing of object 1 is already finished. With n detected

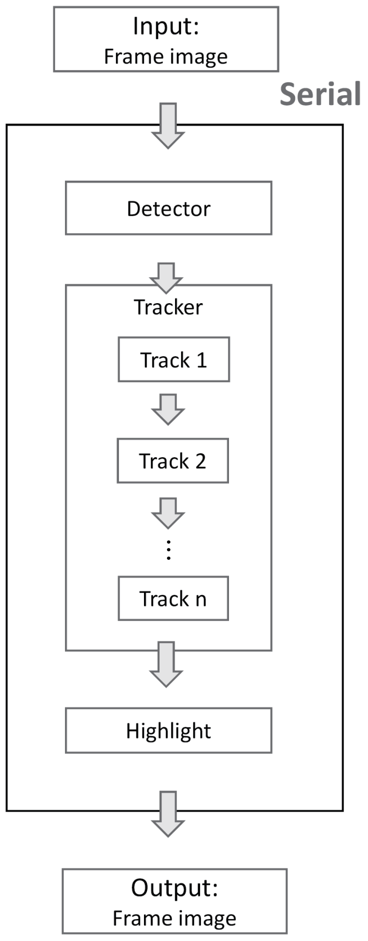

FIGURE 1. Original architecture

objects, n trackers are employed serially. After all trackers are finished, the output frame is generated. This approach requires much time for the tracking process. Also, the efficiency of the DeepSORT algorithm is based on the results of the YOLO process. If the YOLO process cannot detect any bounding box of an object, the DeepSORT algorithm cannot track this object. To overcome the issues of the conventional deep_sort_yolov3, this paper proposes a new architecture as can be seen in Figure 2.

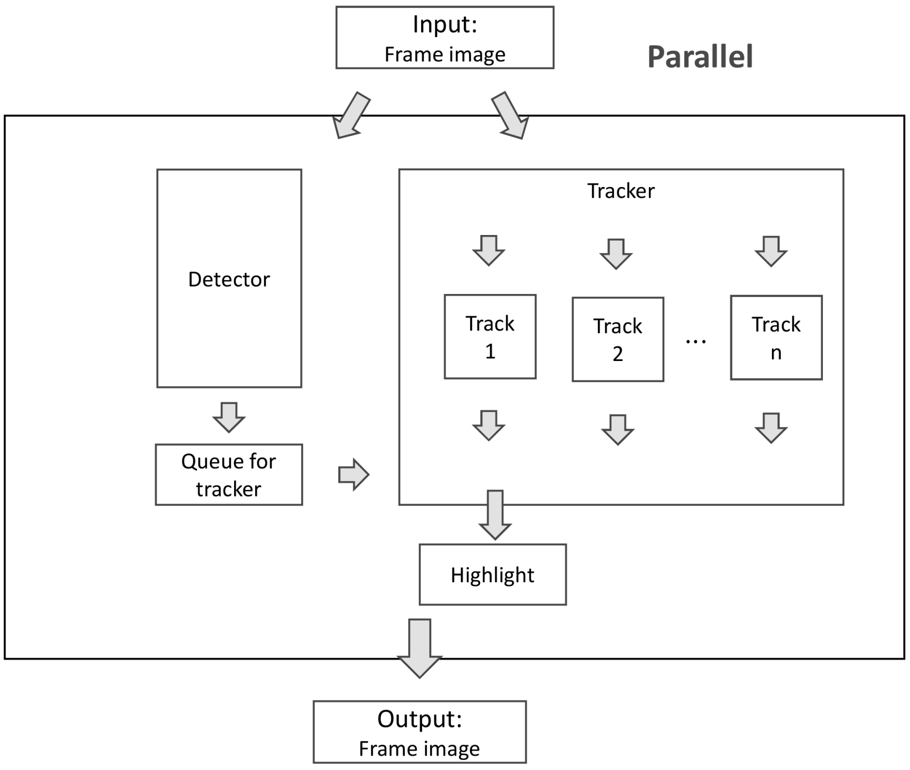

FIGURE 2. Overview of the proposed architecture

The video frames are sent to the detector that uses the YOLO algorithm. The outputs of the detection phase are fed to the tracker module. Different from the conventional serial approach, the proposed architecture uses both the tracker and the detector in parallel. In the serial version, the tracker needs to wait for the end of the detector. On the other hand, any object detected by YOLO will be tracked immediately in the tracker module in the proposed parallel approach. A queue is inserted between the detector and the tracker for the synchronization. The video frames are also sent to the tracker module. To increase the speed, the tracker module is also implemented using the parallel approach.

When the YOLO cannot detect any object in the frame, no bounding box is sent to the DeepSORT process. Hence, the DeepSORT cannot track this missed object. To overcome this issue, our architecture uses another tracker called Dlib correlation tracker (Dlib tracker) that has a discriminative correlation filter to estimate the transitions of the objects [20]. To increase the operating speed, the Dlib tracker is implemented using the multiple process approach.

3.2. **Details of the proposed architecture.** The details of our architecture are shown in Figure 3. The input video is divided and processed frame by frame. An object normally appears on several continuous frames in one video. In addition, YOLO object detection has deep learning processing that requires time and resources. Thus, the object detection is conducted in some frames while in other next frames the object detection is not used. The DeepSORT process cannot work without the results from the object detection phase. Hence, several frames are processed by YOLO, DeepSORT, and Dlib components. On

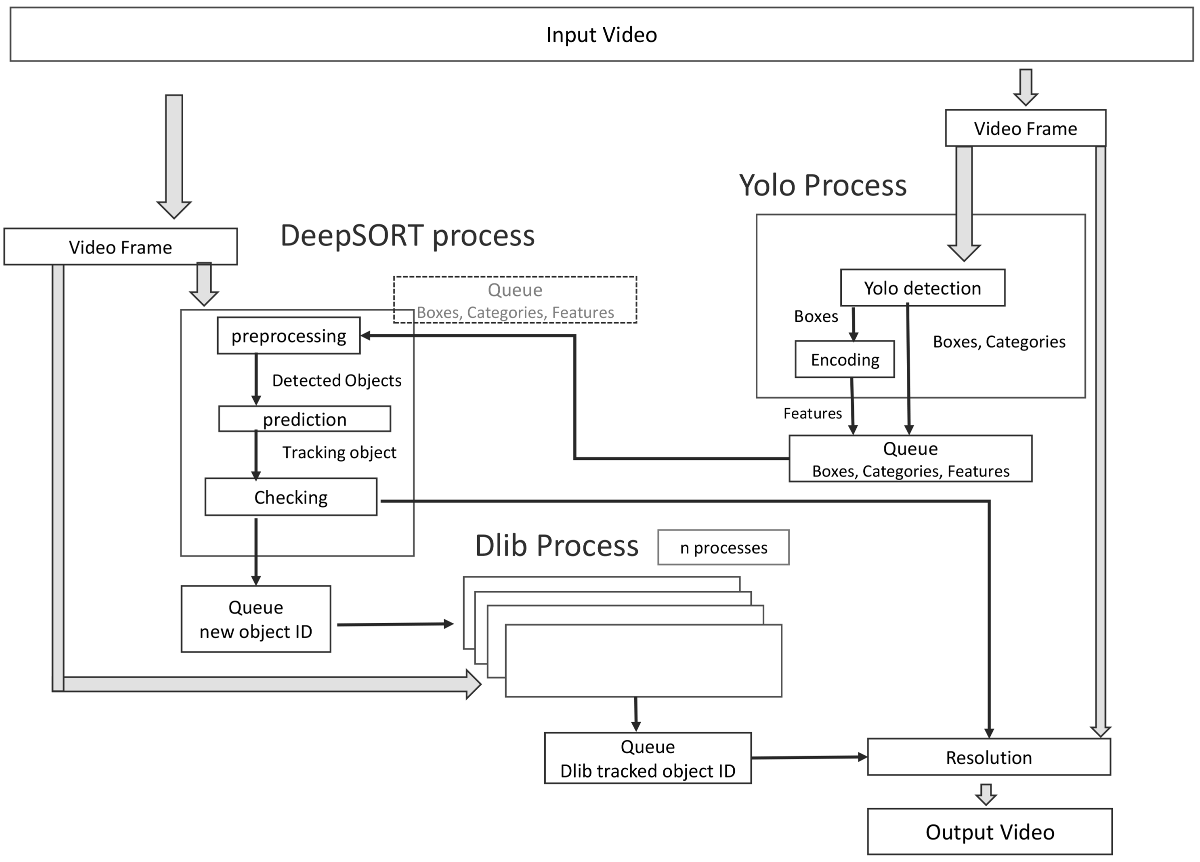

Figure 3. Details of the proposed architecture

the other hand, other frames are processed only by the Dlib component. The processing of the frames that have all three components can be explained as below.

1) The YOLO detection is conducted to get information concerning the bounding box, the class, and the features of detected objects.
2) The information about each object is fed to the DeepSORT module through a queue.
3) In DeepSORT process, there are two inputs that are the video frame and the information of the detected objects, respectively. The preprocessing in the DeepSORT module removes the overlapping bounding boxes. The goal of the tracking is to assign the detected bounding boxes with the object IDs already appearing in the previous frame. If a bounding box cannot be assigned to any previous object ID, a new object ID will be assigned to this bounding box. As presented, DeepSORT tracks objects using two metrics that are the location metric (IoU) and the appearance metric (feature). The detected object is tracked using the IoU metric first. If the bounding box cannot be assigned to any object ID using the IoU metric, the bounding boxes will be processed in the features component. In checking component, if the bounding box is already assigned to an object ID, the information concerning the object ID and corresponding bounding box will be sent to the resolution module. On the other hand, the unassigned bounding boxes are sent to the Dlib process through a queue called "new object ID". The operation is shown in Figure 4.
4) The Dlib process receives the video frame and unassigned boxes from the "new object ID" queue to create a new object ID for each unassigned box. The new object ID is logged and will be used in both DeepSORT and Dlib processes in subsequent frames. The Dlib starts tracking the new object IDs from this current frame. The Dlib algorithm is implemented using a multiple process approach. Each bounding box that

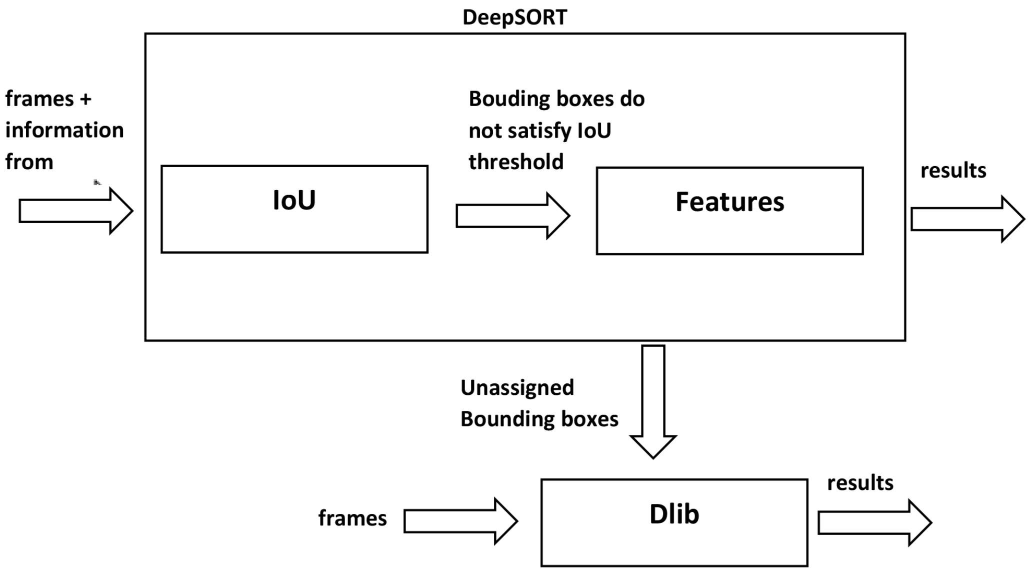
DeepSORT
frames + information from
Bouding boxes do not satisfy IoU threshold
IoU
Features
results
Unassigned Bounding boxes
frames
Dlib
results

Figure 4. DeepSORT-Dlib tracker

comes from the "new object ID" queue is tracked using one CPU process to increase the operating speed of the Dlib tracking operation.

5) The Dlib tracking is independent with the DeepSORT tracking so that each object is tracked by both the Dlib tracker and the DeepSORT tracker.
   - If the matching between the bounding box and the object ID of DeepSORT is determined by features, the bounding box from DeepSORT is used for the object ID. Normally, this situation occurred when an object is hidden in a crowd.
   - If the matching between the bounding box and the object ID of DeepSORT is determined by IoU, a new IoU of bounding box tracked by DeepSORT and the bounding box tracked by Dlib is calculated. The new IoU is the threshold to decide whether the object ID is the bounding box from Dlib or DeepSORT.

In the Dlib-only situation, a frame is sent directly to the Dlib process without the need for YOLO or DeepSORT. Hence, even the YOLO cannot detect the object, the object tracking can continue. To control the number of processes, if an object is out of the tracking region of interest (ROI), the tracking process of this object will be destroyed.

4. **Experimental Results.**

All experiments were conducted with HP Z800 Workstation equipped with 24 Intel Xenon X5675 3.07 GHz. For deep learning processing, this computer also had GeForce RTX 2080 Ti. The proposed deep_sort_yolov3 architecture was tested with the video filmed at Akihabara in Tokyo, Japan. To investigate different situations, two different videos were investigated. The first video had many people while there were fewer people in the second video. Each video was a 26-second video. An example of the results of uncrowded video can be observed in Figure 5, and an experimental result for crowded video is shown in Figure 6.

In the experiments, The frame number $f \bmod 3 = 1$ such as frames 1, 4, 7, 10 had all YOLO, DeepSORT, and Dlib processes. Other frames such as frames 2, 3, 5, 6 employed only the Dlib process. Based on our experiments and previous research [16], a suitable set of parameters was as follows.

- The ROI size of the input frame was 640×480. This ROI size was sufficient for object tracking without compromising the privacy of the people captured in the video.
- IoU_thres = 0.64. This threshold was used to track the objects by the positions of the bounding box intersections. If this value was too small or too big, the objects could not be tracked.

- distance_thres = 0.30. This threshold was used to compare the features of the objects. The distance_thres value was lower than IoU_thres value so that the features component can track the object that the IOU component cannot track.

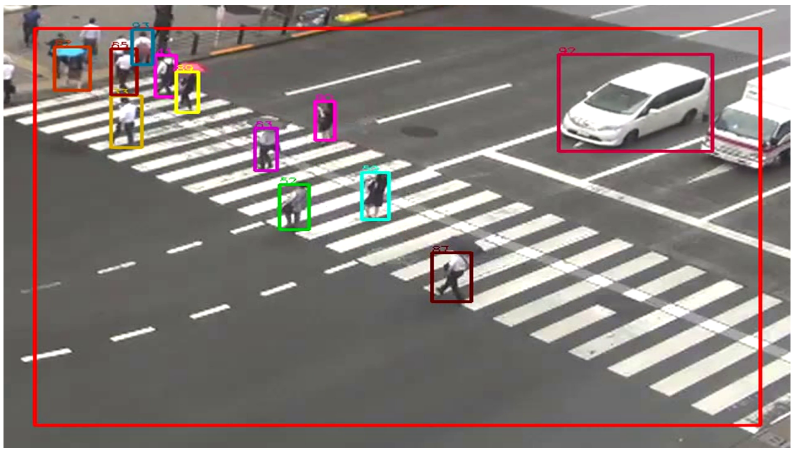

Figure 5. Uncrowded video

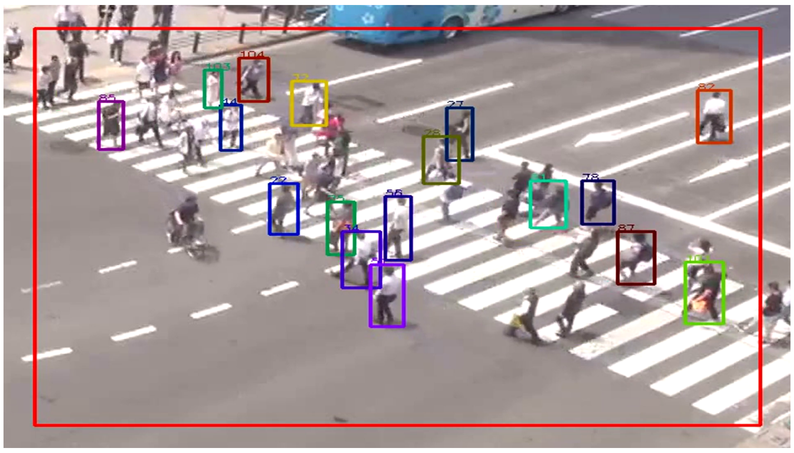

Figure 6. Crowded video

Our experiments were conducted to investigate two aspects. The first one is the advantage of the DeepSORT-Dlib architecture compared with the original DeepSORT-only architecture concerning the identity switches. The second one is the speed advantage of the proposed parallel processing compared with the original serial processing.

4.1. **Identity switches experiment.** If the object changes the identity during the tracking phase, the tracking process will obtain the wrong results. The experiments were conducted to investigate the efficiency of the DeppSORT- Dlib with both crowded video and uncrowded video concerning the identity switches.

The experimental results are presented in Table 1. The number in each cell is the number of identity switches divided by the total number of objects. For example, the DeepSORT-Dlib architecture with the uncrowded video had 81 objects that had identity switches in 6709 detected objects. Compared with the uncrowded video, the number of detected objects and the number of switched increased in the crowded video. However, in both situations, the DeepSORT-Dlib approach obtained lower identity switches.

Table 1. Experiment to investigate the identity switches

	Uncrowded video	Crowded video
DeepSORT-only	271/6804 (3.96%)	519/10522 (4.93%)
DeepSORT-Dlib	81/6709 (1.21%)	324/10298 (3.15%)

4.2. **Operating speed experiment.** The speed advantage concerning the frame per second (FPS) of the proposed parallel architecture was investigated in both crowded video and uncrowded video.

The experimental results are illustrated in Table 2. The number in each cell is the number of frames divided by the number of seconds to run the video. For example, it required 121.76 seconds to run the uncrowded video in the proposed parallel approach that had 1500 frames. Hence, the FPS for this video was 12.32. Results demonstrate that the proposed parallel architecture obtained higher FPS than the original serial architecture in both uncrowded video and crowded video. The results are explained by several reasons.

- The DeepSORT tracker needs to wait for the end of the YOLO detector in the serial version. On the other hand, any object detected by YOLO will be tracked immediately in the tracker modules in the proposed parallel approach.
- The Dlib tracker is implemented using the parallel approach.
- The object detection in the proposed architecture is conducted in some frames only so that YOLO detection time was reduced.

Table 2. Experiment to investigate the operating speed

	Uncrowded video	Crowded video
Serial approach	1500 frames/195.84 seconds (7.66 FPS)	1730 frames/306.54 seconds (5.64 FPS)
Proposed parallel approach	1500 frames/121.76 seconds (12.32 FPS)	1730 frames/204.42 seconds (8.46 FPS)

5. **Conclusions.** This paper proposed a new deep_sort_yolov3 architecture which overcomes the issues of original deep_sort_yolov3 architecture by adding two contributions.

The first contribution relates to identity switches. YOLO object detection could miss the object so that no detected bounding boxes will be forwarded to the DeepSORT components which leads to no tracking results from the DeepSORT. When the YOLO detects again these objects in subsequent frames, the objects will be assigned a new object ID. The Dlib tracker is used to solve this problem because it needs only a bounding box of an object in the first frame. After that, the Dlib can track the object frame by frame without the YOLO component. Even YOLO cannot detect any object, the proposed architecture may continue to track the objects.

The second contribution of this paper concerns the operating speed. In the original architecture, the DeepSORT only tracks the object when the YOLO detection is already finished. In our proposed architecture, any detected object from YOLO is sent immediately to the DeepSORT detection. Hence, object detection and object tracking are conducted in parallel. The Dlib is also implemented in the multiple process approach where each object is tracked by one CPU process. In addition, the object appears in several frames. Therefore, the object detection in the proposed architecture is conducted in some frames only so that we can reduce the YOLO detection time.

Experimental results have demonstrated that the proposed architecture reduced the number of identity switches compared with the original approach. The parallel approach also obtained a higher FPS than the conventional serial approach.

Future research will test the proposed architecture with different videos to investigate different situations. The accuracy of the tracking still depends on the YOLO detection so that another avenue for future study is to improve the YOLO algorithm.

Acknowledgment. This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2018-PC-209.

REFERENCES

[1] P. Viola and M. Jones, Rapid object detection using a boosted cascade of simple features, *Proc. of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[2] G. Feng et al., Template-based object detection through partial shape matching and boundary verification, *International Journal of Signal Processing*, vol.4, pp.148-157, 2008.

[3] G. Cheng and J. Han, A survey on object detection in optical remote sensing images, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol.117, pp.11-28, 2016.

[4] J. Choi, M. Han and N. Kim, Object detection of cochlea in micro-computed tomography using faster region convolutional neural network, *ICIC Express Letters, Part B: Applications*, vol.10, no.7, pp.651-656, 2019.

[5] J. Redmon et al., You only look once: Unified, real-time object detection, *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.779-788, 2016.

[6] K. Mikolajczyk and C. Schmid, A performance evaluation of local descriptors, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.27, no.10, pp.1615-1630, 2005.

[7] H. Bay et al., Speeded-up robust features (SURF), *Computer Vision and Image Understanding*, vol.110, no.3, pp.346-359, 2008.

[8] M. Calonder et al., BRIEF: Binary robust independent elementary features, *European Conference on Computer Vision*, pp.778-792, 2010.

[9] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.886-893, 2005.

[10] X. Li et al., A multiple object tracking method using Kalman filter, *2010 IEEE International Conference on Information and Automation*, pp.1862-1866, 2010.

[11] B. Sahbani and W. Adiprawita, Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system, *2016 the 6th International Conference on System Engineering and Technology*, pp.109-115, 2016.

[12] S. Fukui et al., Particle filter based tracking with image-based localization, *Procedia Computer Science*, vol.96, pp.977-986, 2016.

[13] H. S. G. Supreeth and C. M. Patil, Efficient multiple moving object detection and tracking using combined background subtraction and clustering, *Signal, Image and Video Processing*, vol.12, no.6, pp.1097-1105, 2018.

[14] N. Wojke, A. Bewley and D. Paulus, Simple online and realtime tracking with a deep association metric, *2017 IEEE International Conference on Image Processing*, pp.3645-3649, 2017.

[15] Z. Qiu et al., Old man fall detection based on surveillance video object tracking, *International Symposium on Parallel Architectures, Algorithms and Programming*, pp.159-167, 2019.

[16] Qidian, *Real-Time Multi-Person Tracker Using YOLOv3 and Deep_Sort with Tensorflow*, https://github.com/Qidian213/deep_sort_yolov3, Accessed in February 2020.

[17] Kapania et al., Multi object tracking with UAVs using deep SORT and YOLOv3 RetinaNet detection framework, *Proc. of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems*, pp.1-6, 2020.

[18] J. Redmon and A. Farhadi, *YOLOv3: An Incremental Improvement*, https://arxiv.org/pdf/1804.02767.pdf, Accessed in February 2020.

[19] A. Bewley et al., Simple online and realtime tracking, *2016 IEEE International Conference on Image Processing*, pp.3464-3468, 2016.

[20] M. Danelljan et al., Accurate scale estimation for robust visual tracking, *British Machine Vision Conference*, 2014.