IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 22, 2019, accepted May 18, 2019, date of publication May 27, 2019, date of current version June 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919332

Convolutional Neural Network-Based Real-Time Object Detection and Tracking for Parrot AR Drone 2

ALI ROHAN[1,2], MOHAMMED RABAH[1], AND SUNG-HO KIM[1,2]
[1]School of Electronics and Information Engineering, Kunsan National University, Gunsan 54150, South Korea
[2]Department of Control and Robotics Engineering, Kunsan National University, Gunsan 54150, South Korea

Corresponding author: Ali Rohan (ali_rohan2003@hotmail.com)

ABSTRACT Recent advancements in the field of Artificial Intelligence (AI) have provided an opportunity to create autonomous devices, robots, and machines characterized particularly with the ability to make decisions and perform tasks without human mediation. One of these devices, Unmanned Aerial Vehicles (UAVs) or drones are widely used to perform tasks like surveillance, search and rescue, object detection and target tracking, parcel delivery (recently started by Amazon), and many more. The sensitivity in performing said tasks demands that drones must be efficient and reliable. For this, in this paper, an approach to detect and track the target object, moving or still, for a drone is presented. The Parrot AR Drone 2 is used for this application. Convolutional Neural Network (CNN) is used for object detection and target tracking. The object detection results show that CNN detects and classifies object with a high level of accuracy (98%). For real-time tracking, the tracking algorithm responds faster than conventionally used approaches, efficiently tracking the detected object without losing it from sight. The calculations based on several iterations exhibit that the efficiency achieved for target tracking is 96.5%.

INDEX TERMS Convolutional neural network, deep learning, object detection, target tracking, unmanned aerial vehicles.

I. INTRODUCTION

Deep learning (DL) in Artificial Intelligence (AI) has recently gained a significant interest. It is used in a wide range of applications such as autonomous systems, facial recognition, self-driving cars, image and speech recognition, classification, and object detection. Among the most promising systems that can utilize Deep Learning are Unmanned Aerial Vehicles (UAVs), which are becoming an attractive solution for a wide range of applications. Recently, Convolutional Neural Networks (CNNs) have achieved great results in different fields of recognition, detection, and classification, especially in computer vision. CNN is a class of deep neural networks, and is mostly applied to analyze visual imagery. For the application of object detection and classification, CNN is considered as a very powerful tool. CNNs are biologically inspired hierarchical models that can be trained to perform a variety of detection, recognition, and segmentation tasks [1]. The structure of a CNN typically comprises a feature extractor stage followed by a classifier. In past years, a lot of progress has been made on CNN-based object detection. Several object detectors have been proposed by the deep learning community, including Faster R-CNN [2], R-FCN [3], YOLO [4] and SSD [5]. The main emphasis of these designs is placed on improving (1) detection accuracy and (2) computational complexity of their methods in order to achieve real-time performance for mobile and embedded platforms [6]. CNN-based object detectors are divided into two categories with respect to their high-level structure: (1) region-based detectors and (2) single-shot detectors. Region-based detectors usually consist of a region-proposal stage followed by a classifier. Faster R-CNN is an example of region-based detectors. The problem with region-based detectors is that they are computationally heavy and it is difficult to achieve high performance in embedded platforms. Single-shot detectors, on the other hand, employs a single CNN to perform end-to-end object detection. YOLO and SSD are examples of single-shot detectors. YOLO is designed

The associate editor coordinating the review of this manuscript and approving it for publication was Canbing Li.

for real-time execution and, by design, provides a trade-off favoring high performance over accuracy [7].

Generally, traditional techniques utilize background subtraction [8] or Haar Cascade classifiers to detect object [9]. In [10], authors proposed an automatic disease detection and classification method in radish fields using a camera attached to a UAV. In [11], authors used CNN to detect drones from other flying objects using deep convolutional neural networks. In [12], authors used convolutional neural network for real-time analysis of information and performance in detecting cattle using drone. In [13], authors presented target detection and safe landing algorithm for UAV. In [14], an approach for drone wireless charging was implemented using Hill-climbing algorithm. Many UAV studies have tried to detect and track certain types of objects such as vehicles [15, 16], people including moving pedestrians [17], [18], and landmarks for autonomous navigation and landing [19], [20] in real-time.

In this work, an approach to detect and track the target object, moving or still, using SSD object detector for a UAV is presented. A CNN based on SSD architecture is trained to detect a single class. In case of single class detection, the training of CNN requires a particular approach; it is different from the normal training of the network. For this, we have implemented a training method using positive (images with object) and negative images (images with no object) for training. Also, SSD is selected because it aims to combine the performance of YOLO with the accuracy of region-based detectors [5]. SSD provides higher accuracy for object detection than YOLO (normally used for real time implementation). The problem with SSD is the computational time higher than other object detectors like Faster R-CNN, R-FCN, and YOLO. To gain maximum possible accuracy, SSD is implemented with an optimization method where the computational load is divided onto CPU and GPU, resulting in a very low computational time (10ms) compared to the normal time (27ms). Our system is divided into two parts: (1) object detection and (2) target tracking. Parrot AR Drone 2 is used to implement the object detection and tracking. The object or class used to detect in this work is a "person." For tracking, we used an approach to use the front camera of the AR Drone 2. In order to use front-facing camera, we have developed a tracking algorithm based on the drone's dynamics. In the tracking algorithm, the control parameters include the following: roll φ, pitch θ, yaw Ψ, and altitude Z, which are all controlled using PID controllers. These PID controllers use the position and distance of a target object as an input. Both position and distance are calculated using single front-facing camera of the drone.

This paper is divided into following sections: Section 1 provides an introduction, Section 2 describes the basic architecture of the system; Section 3 explains the object detection and tracking algorithm; Section 4 contains the results, discussion, and future plan of the work; and Section 5 concludes the study.

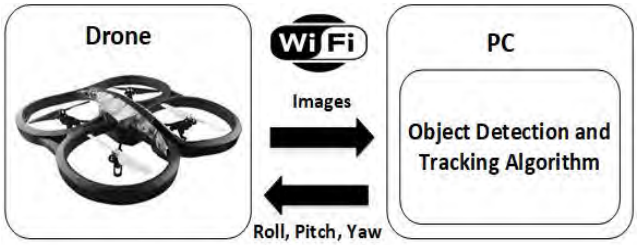


FIGURE 1. Basic architecture of the system.

II. BASIC ARCHITECTURE OF THE SYSTEM

Fig.1 shows the basic architecture of the system. A personal computer (PC) is used to communicate with the AR Drone 2 through a Wi-Fi link. The drone sends the images at a constant frequency of 30Hz (preset by drone driver). These images are received and processed in PC through a CNN for object detection. The detected object's position and distance are calculated to estimate the roll φ, pitch θ, yaw Ψ, and altitude Z values using PID controller-based tracking algorithm. The estimated roll φ, pitch θ, yaw Ψ, and altitude Z values are sent back to the drone to initiate the tracking process.

A. HARDWARE SPECIFICATIONS

The AR Drone 2 is a low-cost hardware platform, costing about US$ 400 [21]. The size of drone is about 50cm $\times$ 50cm, weighing about 500g.The design of the drone makes it feasible to use in indoor and outdoor environments.

The AR Drone 2 is equipped with two cameras, an Inertial Measurement Unit (IMU) including a 3-axis gyroscope, 3-axis accelerometer, 3 axis-magnetometer, a pressure sensor, and an ultrasonic altitude sensor. The cameras, a front and a bottom camera differ in specifications. The front camera has a resolution of 1280 $\times$ 720 at 30fps with a diagonal field of view of 92°, whereas, the bottom camera has a resolution of 320 $\times$ 240 at 60fps with a diagonal field of view of 64°. The camera used in this work is a front camera due to its higher resolution. Accordingly, based on our application of tracking a moving or still object (person), front camera suits well as it always faces the target object.

The AR Drone 2 has a 1 GHz ARM Cortex-A8 CPU and an embedded version of Linux as its operating system. The embedded software on the board measures the horizontal velocity of the drone using its bottom camera and estimates the state of the drone in terms of its roll φ, pitch θ, yaw Ψ, and altitude Z using available sensor information. The horizontal velocity is measured based on two complementary computer vision features: one based on optical flow and the other based on tracking image features (like corners), with the quality of the speed estimates highly dependent on the texture of the input video streams [22]. All sensor measurements are updated at 200Hz. The AR Drone 2 can communicate with other devices like smartphones or laptops

TABLE 1. PC specifications.

Name	Detail
CPU	AMD Ryzen 7 2700 X Eight-core processor @ 3.70 GHz
Memory	16 GB
GPU	NIVIDIA GeForce GTX 1080 Ti

through a standard Wi-Fi network. In this work, a PC is used to communicate with the drone. The specifications of the PC are given in Table 1.

B. OPERATING SYSTEM OVERVIEW

The drone is connected to a PC through Wi-Fi link; in the PC, Robot Operating System (ROS) framework is used to implement object detection and tracking algorithm. The ROS comprises of two nodes: Node 1 is the object detection and tracking node and Node 2 is AR Drone device driver package. Each node communicates with others using ROS transport protocol [23]. In this system, six ROS topics are used. Each ROS topic is responsible to carry the data in the form of an encoded message between the nodes. The node graph of the system is shown in Fig.2.

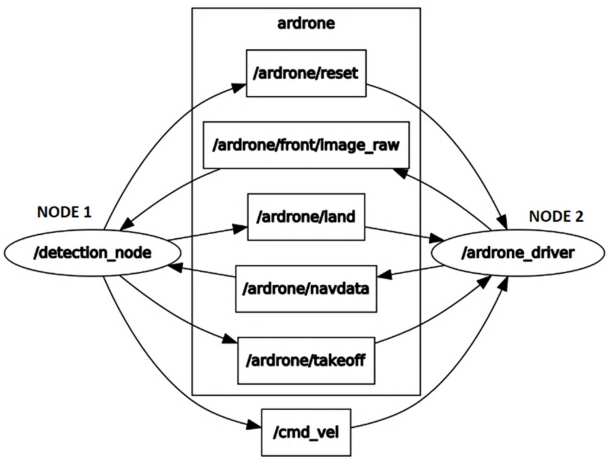


FIGURE 2. ROS node graph.

Node 1 publishes 04 ROS topics related to drone operation. These ROS topics are */ardrone/takeoff*, */ardrone/land*, */ardrone/reset* and */cmd_vel*. The */cmd_vel* ROS topic is responsible to carry the roll φ, pitch θ, yaw Ψ and altitude Z commands for AR Drone 2. Node 1 subscribes to 02 ROS topics such as */ardrone/front/image_raw* and */ardrone/navdata*. These ROS topics are responsible for the transmission of video and navigation data respectively.

On the other hand, Node 2 subscribes to 04 ROS topics published by Node 1 and publishes 02 ROS topics such as */ardrone/front/image_raw* and */ardrone/navdata*.

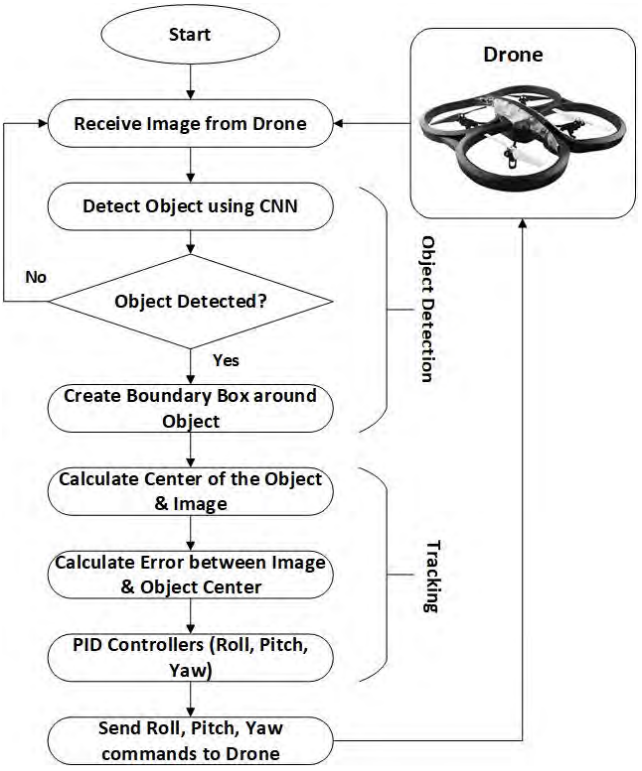


FIGURE 3. Flowchart of the overall system.

III. OBJECT DETECTION AND TRACKING ALGORITHM

Fig.3 shows the overall flow chart of the system. The drone sends image to a PC. The received image is processed through a CNN for object detection. If the object is detected in the image, the CNN returns the image with a boundary box on the detected object. This boundary box provides information on object position. Further, the error between the image center and the object center is calculated. This error is given as an input to PID controllers that generate roll φ, yaw Ψ, and altitude Z commands. For pitch θ, the relative distance of the object is calculated using the position data of the detected object. Based on the relative distance of the object, PID controller generates pitch θ command to operate the drone.

If there is no object detected in the image, the roll φ, pitch θ, yaw Ψ, and altitude Z values remain the same and the drone holds its position until an object is detected.

A. OBJECT DETECTION

AR Drone 2 sends the image taken by the front camera to a PC through Wi-Fi link. The received image is passed through a CNN, trained to detect the desired object (person). The CNN used in this work is based on a single shot detector (SSD) architecture. The architecture of the SSD is shown in Fig.4. The SSD approach is based on a feed-forward convolutional network that produces a fixed size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detection [24]. As shown in Fig.4, SSD

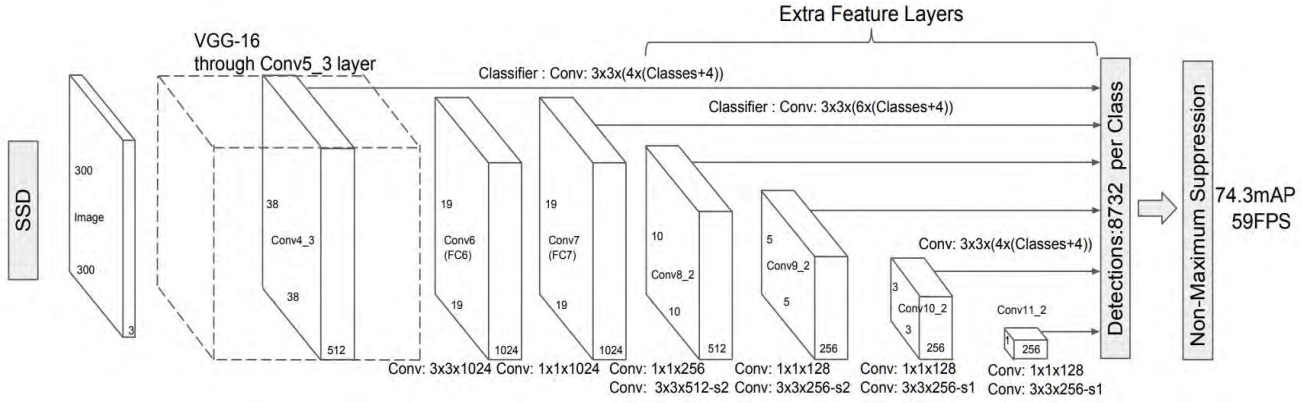


FIGURE 4. Basic architecture of SSD.

architecture builds on the VGG-16 (Visual Geometry Group -16) architecture, but discards the fully connected layers. VGG-16 is used as the base network because of its strong performance in high quality image classification tasks. It is also effective in addressing problems where transfer learning helps in improving results. Instead of the original VGG fully connected layers, a set of auxiliary convolutional layers (from conv6 onwards) was added, thus enabling to extract features at multiple scales and progressively decreasing the size of the input to each subsequent layer. The detailed description of SSD architecture can be found in [24].

1) CNN TRAINING

The CNN was trained to detect a single class, in this particular study, a person. It should be noticed that the training is not done to detect a particular person. Detection of a particular person requires different techniques where some special features are extracted. In this work our focus is on single class detection rather than unique single class. An image database was formed containing positive and negative images (images without object). We took 5,100 images, with the ratio of 3:1 (negative to positive images). Each positive image containing the object (person) was labeled. Images were divided into training and test dataset. 80% of the images were used for training dataset while 20% images were used for testing the network. The network was trained for 50,000 iterations on a PC with specifications mentioned in Table 1. Fig.5 shows some examples of the images used for training and testing the CNN. The accuracy achieved for the object detection is 98.2 %. Fig.6 presents some results of the CNN after training.

FIGURE 5. Examples of images used for training and testing the CNN.

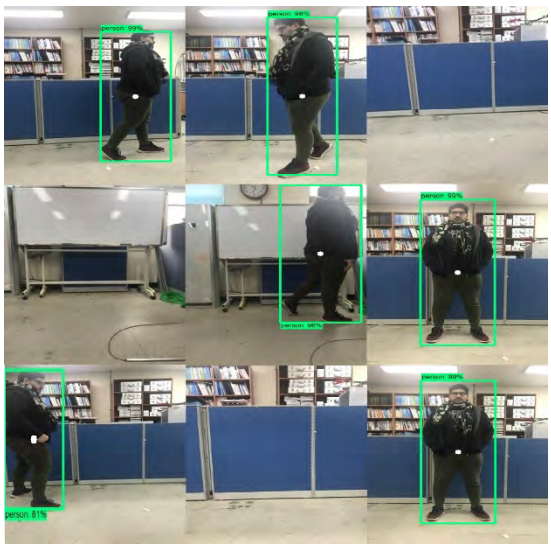


FIGURE 6. Output of the CNN for some images from train and test dataset after training.

B. OBJECT TRACKING

Once the object is detected in an image, the CNN returns the image with a boundary box on the detected object. This boundary box contains the information about the position of the object on an image. Fig.7 shows an example of the boundary box with the position of the object in an image. It can be seen from Fig.7 that the object position is given by pixel values $(x_{min}, y_{\min})$, $(x_{\max}, y_{\min})$, $(x_{\min}, y_{\max})$, (x_{max}, y_{max}).

From these position values the center of the object is calculated as:

$$(x_o, y_o) = \left(\frac{x_{min} + x_{max}}{2}, \frac{y_{min} + y_{max}}{2} \right) \quad (1)$$

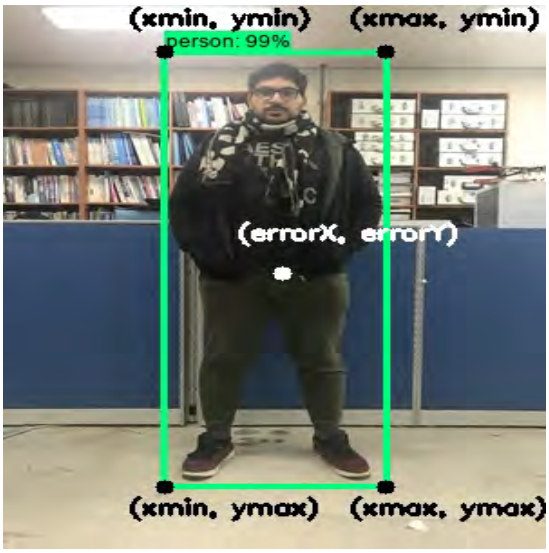


FIGURE 7. Boundary box with detected object.

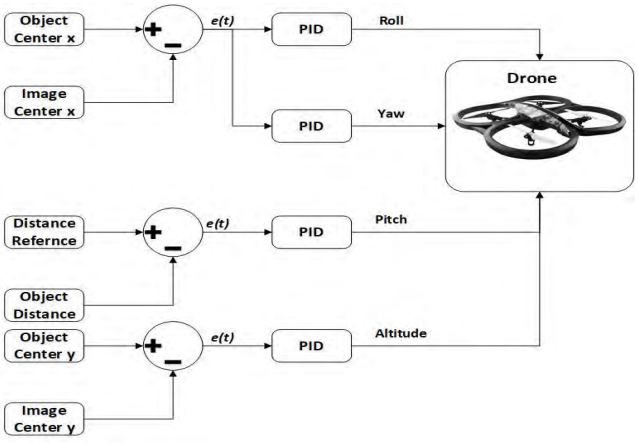


FIGURE 8. Block diagram of the control scheme for tracking the detected object.

Also, the center of the image is required to track the object, and is calculated as:

$$(x_i, y_i) = \left(\frac{img_{width} - img_{width}}{2}, \frac{img_{height} - img_{height}}{2} \right) \quad (2)$$

$$(x_i, y_i) = (0, 0) \quad (3)$$

For simplicity, the center of the image is taken as$(0, 0)$. The error between the center of the image and the object center is given as:

$$e_x(t) = x_o - x_i = x_o \quad (4)$$

$$e_y(t) = y_o - y_i = y_o \quad (5)$$

From the equations above, it can be concluded that in order to properly track an object, $e_x(t)$ and $e_y(t)$ should always be equal or near to zero. In other words, the detected object center should always be equal to the image center for proper tracking.

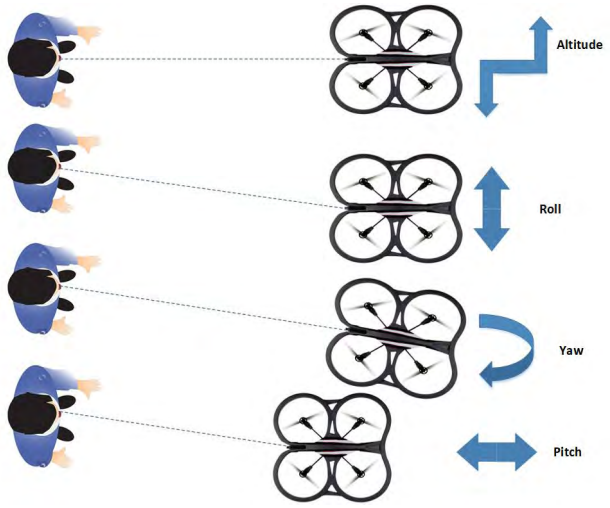


FIGURE 9. Basic drone movements.

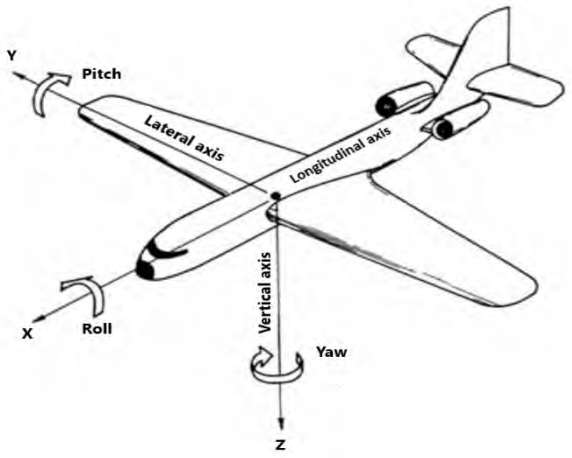


FIGURE 10. Basic axis of movement of drone.

For this, a control scheme based on PID controllers is developed. Fig.8 shows the block diagram of the control scheme for tracking the detected object. Fig.9. shows the basic drone movements of the AR Drone 2. There are four control parameters responsible for drone movement. Throttle (altitude Z) is responsible for upward or downward movement, roll φ is responsible for left or right movement, pitch θ is responsible of forward or backward movement, and yaw Ψ is responsible for clockwise or counter-clockwise rotation of the drone. Fig.10 shows the basic axis of the movement of drone. In this work, based on the AR Drone 2 movement axis, x-axis (position of the object) values are used to control the roll φ and yaw Ψ while y-axis (position of the object) values are used to control the altitude Z. On the other hand, pitch θ is controlled by measuring the relative distance between the object and the drone. To calculate the relative distance, the width of the boundary box of the detected object is measured. If the width of the boundary box is higher than a set reference value, the drone will move backward otherwise it will move forward.

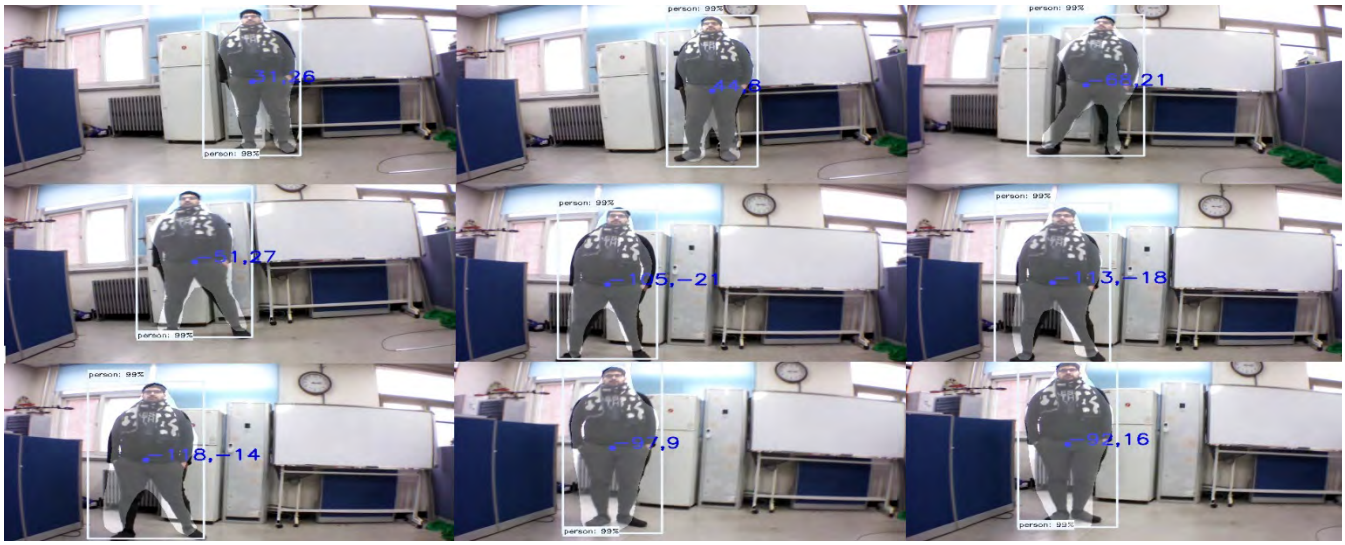


FIGURE 11. Real-time output of the CNN through drone camera.

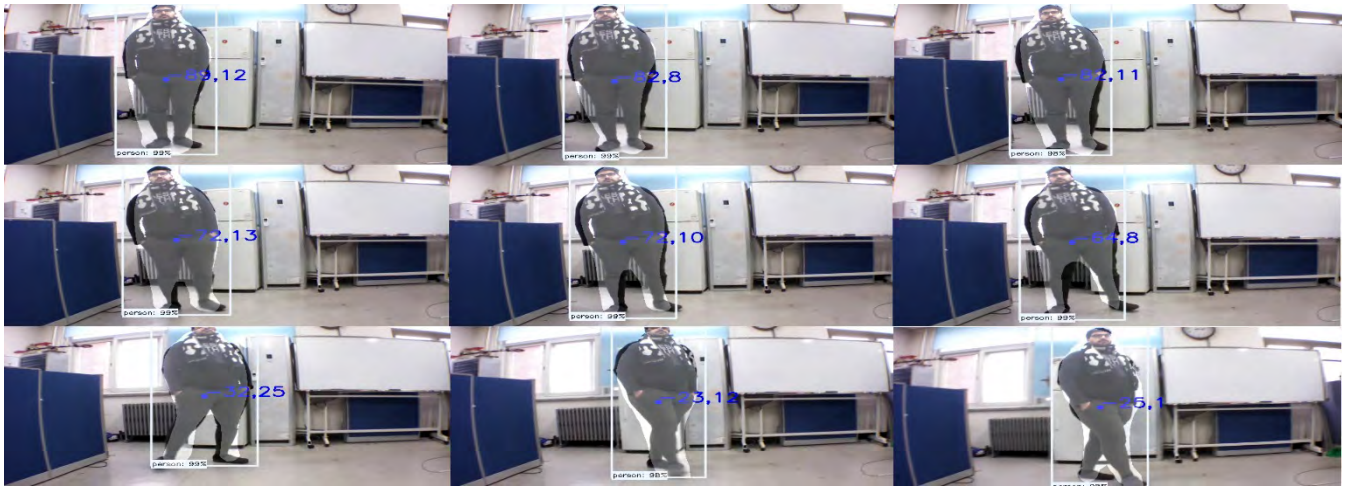


FIGURE 12. Some more results of the output of the CNN through drone camera.

IV. RESULTS AND DISCUSSION

In order to implement the SSD architecture-based CNN in real-time, we have to use Split-Model method to reduce the computational time of the object detector. In Split-Model method, the processes are carried out by CPU and GPU rather than done exclusively by either CPU or GPU. It gave us the advantage of low computational time. The time it takes for the object detector to process a frame and give output on the detected object was 10ms, which is 2.7 times less than the normal computational time of the SSD (27ms). The reduction in the computational time made it feasible to implement SSD on a system like drone where response time is a very critical parameter to consider. Furthermore, it provided the chance to take advantage of SSD's accuracy over other conventionally used real-time object detectors like YOLO. In 10ms, frames received through the Wi-Fi link from the drone's front camera are processed and the CNN gives the output in the form of an image with the position of the detected object. The Frames-Per-Second (FPS) achieved was 58 FPS. Fig.11 and 12 show the results of the object detector output recorded during a real-time test of the drone. Fig.13 shows the results obtained for tracking of the detected target in any direction. From Fig.11, 12, 13, it can be seen that the target object moves in different directions but drone keeps the target in the center of the image and continue to track the object. In the results presented the target moved in forward and backward direction, left and right direction, also with varying height.

To evaluate the performance of the object detector we applied the following four metrics:

(1) Intersection over Union (IoU): In the context of object detection, the IoU metric captures the similarity between the predicted region and the ground-truth region for an object

FIGURE 13. CNN detection and drone tracking for varying altitude of the object.

present in the image. This metric is defined as the size of the intersection of the predicted and ground-truth regions divided by their union.

(2) Sensitivity: This metric is defined as the proportion of true positives that are correctly identified by the detector. This metric is calculated by taking into account the True Positives (*Tpos*) and False Negatives (*Fneg*) of the detected object as given by (6).

$$Sensitivity = \frac{Tpos}{Tpos + Fneg} \tag{6}$$

(3) Precision: This metric is a widely used metric by the object detection community. It is defined as the proportion of True Positives among all the detections of the system and is given by (7).

$$Precision = \frac{Tpos}{Tpos + Fpos} \tag{7}$$

(4) Frames-Per-Second (FPS): FPS is the rate at which an object detector is capable of processing incoming camera frames.

In order to compute the overall performance of the object detector, we define a composite score metric. This metric consists of a linear combination of IoU, Sensitivity, and Precision together with the achieved FPS. We parametrize the score with respect to a vector of weights $w \in [0, 1]^4$ as given by (8). We prioritized FPS with a weight of 0.4 over the other three accuracy-related metrics, which were equally weighted with 0.2.

$$\begin{aligned} Score\,(w) = {} & w_1 \times FPS + w_2 \times IoU + w_3 \\ & \times Sensitivity + w_4 \times Precision \end{aligned} \tag{8}$$

With $Tpos = 4998$, $Fneg = 90$ and $Fpos = 12$ out of 5100 images database. The calculated score with sensitivity of 0.98 and precision of 0.99 for the object detector was 23.594. We compared our SSD object detector with a

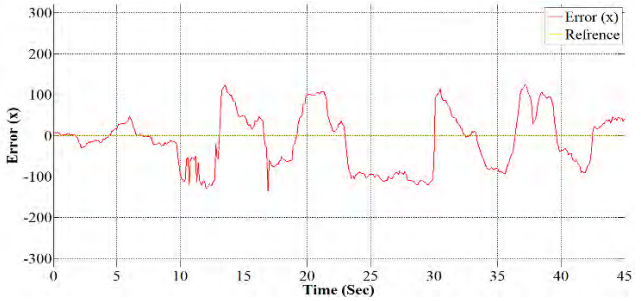


FIGURE 14. Output of the PID controller for Roll.

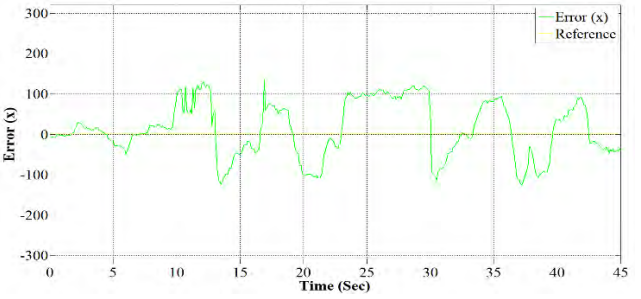


FIGURE 15. Output of the PID controller for Yaw.

YOLO object detector with same image database and the score calculated for the YOLO was 21.18. More importantly, the accuracy achieved for the SSD was 98% higher than YOLO's 96%.

With respect to object tracking, the results achieved are presented in Fig.14–18. In Fig.14, the response of the drone in the form of a calculated error for roll φ is presented. The graph was drawn after a time test for 45 seconds. During this period, the target object (person) kept on moving. It can be noticed that the drone tends to follow the target object and tries to minimize the error between the reference and the

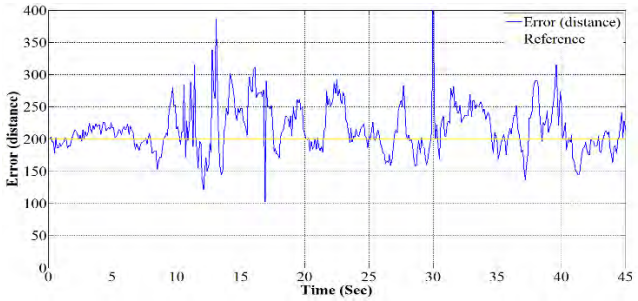


FIGURE 16. Output of the PID controller for Pitch.

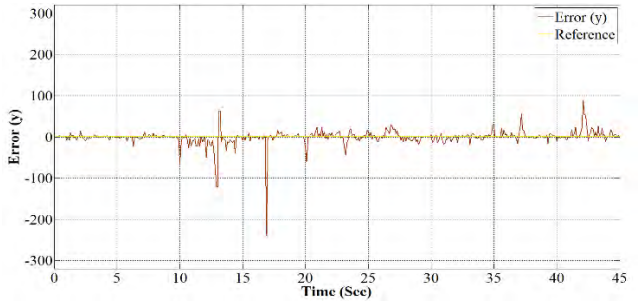


FIGURE 17. Output of the PID controller for Altitude.

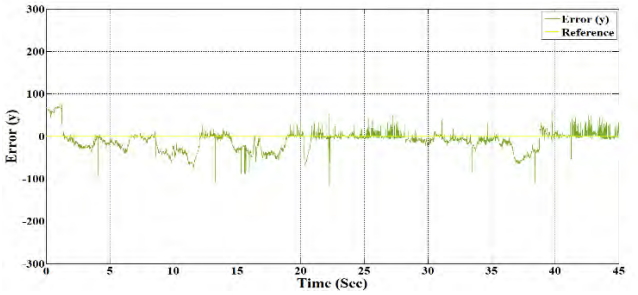


FIGURE 18. Output of the PID controller for varying Altitude.

current value of the x-axis position of the detected object x_o. In addition, from Fig. 15, the recorded response for the yaw Ψ is almost the same as for the roll φ except the direction of the yawing is opposite in this case. This is because of the limitation of 2d axis of the image using the front camera of the drone, as the roll φ and yaw Ψ are along the same axis of movement and the current value of the x-axis position of the detected object $x_o$was used to control these two movements.

For pitch θ, the relative distance was fixed to a reference of 200, which is equal to the distance of 1m on ground between the target object and the drone. It is observed in Fig.16 that the error between the reference and the calculated relative distance changes in relation to the movement and the distance between the target object and the drone. In controlling the altitude of the drone, the y-axis position of the detected object y_o was used. Fig. 16 shows the recorded response between the reference altitude Z and the calculated altitude Z of the drone. It should be noticed that the results presented were recorded after the initial takeoff and tracking algorithm activation. In Fig.17, the altitude Z at start is zero because when the drone takes off the altitude Z is fixed to 1m. After takeoff, the drone starts hovering at the fixed altitude Z and waits for the activation of the object detection and tracking algorithm. Once an object is detected, the drone starts the tracking process. In Fig.17, the reference value of zero is equal to the fixed altitude Z of 1m. Fig.18, shows the result obtained when the detected target changed the height, it can be observed that the drone continues to track the detected object and keeps on following it (as shown in Fig.13). It should also be considered that the minimum height selected is 1m which is necessary in the system like drone due to the fact that when drone is flying near to the ground, there are some forces known as OGE (outer ground force) and IGE (inner ground force) acting on the drone's body. These forces cause disturbance in the drone flight [25]. The tracking system works well to eliminate the error in all of the three-axis (x-axis= roll φ & yaw Ψ, y-axis = altitude Z, z-axis = pitch θ). The efficiency (calculated based on 50 tests iterations) achieved for target tracking is 96.5%. The tracking algorithm can be further improved by using some other type of controllers such as Fuzzy Logic and Fuzzy-PID controllers. We previously implemented those controllers for target tracking in [26] but in this case as mentioned previously that main problem with CNN is the computational time so adding Fuzzy Logic or Fuzzy-PID controller increases the time and also response of the drone gets bad. Hence further research is required to implement such kind of controllers with CNN and AI.

Some problems we had to tackle during the implementation of the object detection and tracking algorithm were the drone response time and limitation of the drone device drivers provided for ROS. The drone response time for AR Drone 2 needed to be synchronized properly with the ROS device drivers. Using ROS drivers for drone, it was noticed that at least 100ms time is required to update the drone control parameters roll φ, pitch θ, yaw Ψ, and altitude Z. That is why, in the presented results, it is documented that the response of drone is a bit slow, although in practical results, no lag or delay was observed.

Also, while implementing SSD object detector it was noticed that object detector missed the target object. It is because of some limitations of the CNN. Practically, it is very hard to observe and there is not much effect on the tracking process of the drone but it can be further improved.

A. FUTURE WORK

Since getting the results from the AR Drone 2, we have recently tried to implement the object detection and tracking algorithm for an application of bridge surveillance to detect structural cracks. Previously Open-CV is used in such kind of application but with recent development of AI and CNN, better results can be achieved. We are trying to implement the same algorithm but using a different drone, this time a quadcopter big enough in size for such kind of application.

The real challenge is the implementation of the same algorithm on an embedded Artificial Intelligence (AI) computing device NVIDIA Jetson TX2 as the computational time will increase in comparison to the one, we have achieved in this work. The problems found during this work such as object detector missed detection will also be solved by introducing some trackers to smooth the detection process and Fuzzy Logic will be used to control the drone in tracking.

V. CONCLUSION

In this work, an approach for real-time implementation of CNN-based object detector and tracking system for AR Drone 2 using SSD architecture is presented. In real-time system where computational time is very big parameter and it effects the efficiency of the system, SSD implementation with one class detection is a challenging task. We have developed an approach which can be implemented in real-time and advantages of high accuracy of SSD architecture can be used by reducing the computational time. The proposed system comprises of two parts: object detection and target object tracking. The efficiency achieved for object detection is 98% which is very reasonable for a complex system like drone. Target object tracking with simple PID controllers is combined with the detection algorithm by designing a specific approach where x, y and z axis (a 3d-plan) is considered. The accuracy achieved for the target object tracking based on several experiments is 96.5%. The achieved results have proved that using an SSD object detector, more accurate results can be achieved within a reasonable computational time. Significantly, it is very feasible to implement such kind of strategy in a complex system like drone.

REFERENCES

[1] N. Audebert, B. Le Saux, and S. Lefèvre, "Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images," *Remote Sens.*, vol. 9, no. 4, p. 368, 2017. [Online]. Available: http://www.mdpi.com/2072-4292/9/4/368

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[3] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. NIPS*, 2016, pp. 379–387.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Proc. ECCV*, 2016, pp. 21–37.

[6] G. D. T. N. Subarna Tripathi and B. Kang, "Low-complexity object detection with deep convolutional neural network for embedded systems," *Proc. SPIE*, vol. 10396, pp. 10396-1–10396-15, Sep. 2017.

[7] A. De Bruin and M. Booysen, "Drone-based traffic flow estimation and tracking using computer vision," in *Proc. Annu. Southern Afr. Transp. Conf.*, Jul. 2015.

[8] J. Redmon. (2016) *Darknet: Open Source Neural Networks in C*. [Online]. Available: http://pjreddie.com/darknet/

[9] C. L. Azevedo, J. L. Cardoso, M. Ben-Akiva, J. P. Costeira, and M. Marques, "Automatic vehicle trajectory extraction by aerial remote sensing," *Proc. Soc. Behavioral Sci.*, vol. 111, pp. 849–858, Feb. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877042814001207

[10] L. M. Dang, S. I. Hassan, I. Suhyeon, A. K. Sangaiah, I. Mehmood, S. Rho, S. Seo, and H. Moon, "UAV based wilt detection system via convolutional neural networks," *Sustain. Comput., Inform. Syst.*, 2018.

[11] M. Saqib, S. D. Khan, N. Sharma, and M. Blumenstein, "A study on detecting drones using deep convolutional neural networks," in *Proc. IEEE 14th Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Lecce, Italy, Aug./Sep. 2017, pp. 1–5. doi: 10.1109/AVSS.2017.8078541.

[12] A. Rivas, P. Chamoso, A. González-Briones, and J. M. Corchado, "Detection of cattle using drones and convolutional neural networks," *Sensors*, vol. 18, no. 7, p. 2048, 2018. doi: 10.3390/s18072048.

[13] M. Rabah, A. Rohan, M. Talha, K. H. Nam, and S. H. Kim, "Autonomous vision-based target detection and safe landing for UAV," *Int. J. Control Automat. Syst.*, vol. 6, no. 6, pp. 3013–3025, 2018.

[14] A. Rohan, M. Rabah, F. Asghar, M. Talha, and S.-H. Kim, "Advanced drone battery charging system," *J. Elect. Eng. Technol.*, vol. 14, no. 3, pp. 1395–1405, 2019.

[15] T. P. Breckon, S. E. Barnes, M. L. Eichner, and K. Wahren, "Autonomous real-time vehicle detection from a medium-level UAV," in *Proc. 24th Int. Conf. Unmanned Air Vehicle Syst.*, 2009, pp. 29.1–29.9.

[16] J. Gleason, A. V. Nefian, X. Bouyssounousse, T. Fong, and G. Bebis, "Vehicle detection from aerial imagery," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 2065–2070.

[17] A. Gaszczak, T. P. Breckon, and J. Han, "Real-time people and vehicle detection from UAV imagery," *Proc. SPIE*, vol. 7878, Jan. 2011, Art. no. 78780B.

[18] H. Lim and S. N. Sinha, "Monocular Localization of a moving person onboard a Quadrotor MAV," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2015, pp. 2182–2189.

[19] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robot. Auton. Syst.*, vol. 62, no. 11, pp. 1646–1656, 2014.

[20] C. Forster, M. Faessler, F. Fontana, M. Werlberger, and D. Scaramuzza, "Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 111–118.

[21] *AR.Drone 2.0*. Accessed: Jan. 21, 2019. [Online]. Available: http://ardrone2.parrot.com/

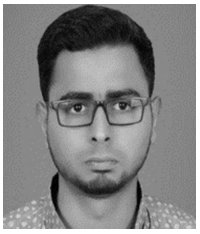
[22] P.-J. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the AR.Drone micro UAV," in *Proc. 18th IFAC World Congr.*, vol. 18, no. 1, pp. 1477–1484, 2011.

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. IEEE Int. Conf. Robot. Automat., Open-Source Softw. Workshop (ICRA)*, 2009, p. 5.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," 2015, *arXiv:1512.02325*. [Online]. Available: https://arxiv.org/abs/1512.02325

[25] M. Talha, F. Asghar, A. Rohan, M. Rabah, and S. H. Kim, "Fuzzy logic-based robust and autonomous safe landing for UAV quadcopter," *Arabian J. Sci. Eng.*, vol. 44, no. 3, pp. 2627–2639, 2019. doi: 10.1007/s13369-018-3330-z.

[26] M. Rabah, A. Rohan, S. A. S. Mohamed, and S.-H. Kim, "Autonomous moving target-tracking for a UAV quadcopter based on fuzzy-PI," *IEEE Access*, vol. 7, pp. 38407–38419, 2019. doi: 10.1109/access.2019.2906345.



ALI ROHAN received the B.S. degree in electrical engineering from The University of Faisalabad, Pakistan, in 2012, and the M.S. degree in electrical, electronics and control engineering from Kunsan National University, South Korea, in 2018. He is currently pursuing the Ph.D. degree in electrical, electronics and control engineering from Kunsan National University, South Korea, where he is also a Research Associate. From 2012 to 2013, he was a Development Engineer with the Niagara Group of Industries, Pakistan. From 2013 to 2015, he was a Project Engineer with Circle Club, Pakistan. From 2015 to 2016, he was a Project Manager of Steam Masters, Pakistan, and also as a Lecturer with the Department of Electrical and Telecommunication Engineering, Government College University Faisalabad, Pakistan. His Research interests include machine learning, AI, UAV's, power electronics, fuzzy logic, EV systems, and flywheel energy storage systems.

MOHAMMED RABAH received the B.S. degree in electronics and telecommunication engineering from AL-SAFWA High Institute of Engineering, Cairo, Egypt, in 2015, and the M.S. degree in electrical, electronics and control engineering from Kunsan National University, Kunsan, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical, electronics and control engineering. His research interests include UAV's, fuzzy logic systems, and machine learning.

SUNG-HO KIM received the B.S. degree in electrical engineering from Korea University, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from Korea University, in 1986 and 1991, respectively. In 1996, he held a postdoctoral position with Japan Hiroshima University. He is currently a Professor with Kunsan National University. His research interests include fuzzy logic, sensor networks, neural networks, intelligent control systems, renewable energy systems, and fault diagnosis systems.