IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 7, 2020, accepted February 19, 2020, date of publication February 28, 2020, date of current version March 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977120

Heterogeneous Parallelization for Object Detection and Tracking in UAVs

MOHAMMED RABAH[1,2], ALI ROHAN[1], MOHAMMAD-HASHEM HAGHBAYAN[3], JUHA PLOSILA[3], AND SUNG-HO KIM[4]

[1]School of Electronics and Information Engineering, Kunsan National University, Gunsan 54150, South Korea
[2]Department of Electronics and Telecommunication Engineering, Al-Safwa High Institute of Engineering, Cairo 11837, Egypt
[3]Department of Future Technologies, University of Turku (UTU), 20500 Turku, Finland
[4]Department of Control and Robotics Engineering, Kunsan National University, Gunsan 54150, South Korea

Corresponding author: Mohammed Rabah (mohamedmostafamousa1991@gmail.com)

ABSTRACT Recent technical advancements in both fields of unmanned aerial vehicles (UAV) control and artificial intelligence (AI) have made a certain realm of applications possible. However, one of the main problems in integration of these two areas is the bottle-neck of computing AI applications on UAV's resource limited platform. One of the main solution for this problem is that AI and control software from one side and computing hardware mounted on UAV from the other side be adopted together based on the main constraints of the resource limited computing platform on UAV. Basically, the target constraints of such adaptation are performance, energy efficiency, and accuracy. In this paper, we propose a strategy to integrate and adopt the commonly used object detection and tracking algorithm and UAV control software to be executed on a heterogeneous resource limited computing units on a UAV. For object detection, a convolutional neural network (CNN) algorithm is used. For object tracking, a novel algorithm is proposed that can execute along with object tracking via sequential stream data. For UAV control, a Gain-Scheduled PID controller is designed that steers the UAV by continuously manipulation of the actuators based on the stream data from the tracking unit and dynamics of the UAV. All the algorithms are adopted to be executed on a heterogeneous platform including NVIDIA Jetson TX2 embedded computer and an ARM Cortex M4. The observation from real-time operation of the platform shows that using the proposed platform reduces the power consumption by 53.69% in contrast with other existing methods while having marginal penalty for object detection and tracking parts.

INDEX TERMS CNN, object detection, object tracking, Gain-Scheduled PID, quadcopter.

I. INTRODUCTION

Object recognition and tracking is one of the most challenging tasks in autonomous aerial vehicles since the detection and tracking the objects should be accurate and agile in run-time and with rational energy consumption. There has been several methods focusing on object tracking and based on making use of different sensors [1], thereof using vision-based object detection is one of the most cheapest and convenient one beside which the obtained information from vision sensors, e.g., RGB camera, can be used in other tasks simultaneously, e.g., odometry and navigation [2], [3].

The main drawback of vision-based object tracking methods is the high computation cost of executing their algorithms w.r.t. the performance, energy, and accuracy [4]. Using cloud servers for object detection is not possible solution since the communication cost between the drone and cloud enormously prolongs response time in real-time stream data processing of object tracking. Furthermore, detecting objects in run-time basically faces noisy and low resolution images accompanied by the background motion that negatively affects the accuracy of the detection outcome [5]. Beside all of these, other real-time or non-real time tasks, e.g., navigation and control tasks, should be executed on-board consuming resources and energy. Therefore, application execution on this platform demands appropriate system architecture and adopted algorithms to improve the system constraints as much as possible while meeting the strict requirements.

In this paper, a heterogeneous platform for stream data processing in real-time object detection and tracking is proposed.

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

The main ideas of the proposed system; 1) achieve high framerate for real-time detection on an embedded platform, 2) overcome the resources limitation of the embedded platform by consuming less power, which will extend the flight time of the quadcopter, and finally, 3) achieve higher accuracy in detecting the target, thus the quadcopter can keep tracking the object without losing it. For this, two heterogeneous onboard processing units are used, i.e., *big* and *little* for object detection and following, and the tasks are managed on them based on the requirements and constraints of the system. The heavy parts of the object tracking application are *mapped* on big processing unit while the light weight part of the application are mapped on the little processing units. Since the object detection is a *stream application* [6], the big processing unit activity is adjusted by the requirement of the processing for each iteration that determined by [7]. The required heart-beat is adjusted via the ability of the little part and accuracy of the tracking. The task scheduling part is designed in a way that the workload is evenly distributed in this heterogeneous platform. Adaptive multi-layered CNN is used for object detection that is running on big processing unit. While the little processing unit is responsible for tracking process. Using this heterogeneous platform and appropriate scheduling of tasks, we adjust the performance and accuracy of the tracking part according to the requirements of the feed-back based system.

In this work, an SSD architecture is implemented on an embedded Artificial Intelligence (AI) computing device NVIDIA Jetson TX2 (big processing unit). The CNN is trained to detect two classes. The first class contains images with an object, which can be thought as positive images, while the other class contains images with no object which can be thought as negative class. Furthermore, the object detection algorithm is combined with an object tracking algorithm based on Gain-Scheduled PID controller to follow the detected object under variable speed. The reason for choosing the Gain-Scheduled PID controller is to overcome the instability and non-linearity of the quadcopter system, thus enables it to follow a target under various speeds. The output of the object tracking algorithm is sent to the flight controller (little processing unit) to start the tracking process by sending the required pulse width modulation (PWM) values to the motors.

The remaining part of this paper is divided as follows: Section II review the related work about workload balancing, task scheduling, and mapping. Moreover the state-of-the-art for object detection and designing controllers for quadcopters will be discussed in this chapter. System architecture is illustrated in Section III. Section IV explains the real-time object detection and tracking system. Section V demonstrates the simulation studies and the experimental results, followed by the conclusion in section VI.

II. RELATED WORK

Recent studies includes vast investigation on improvement of system performance using heterogeneous distributed parallelization in fog-edge layers. In [8] the authors propose a neural network based approach to the acceleration of approximate programs for on-chip system. In [9] a hierarchical management framework for on-chip asymmetric multi-core architectures is demonstrated. The target multi-core system is ARM big.LITTLE mobile platforms. In this architecture, cores have different size that can be used to execute different types of applications. In [10] the authors have done a similar attempt to exploit energy efficiency in symmetric multi-core processors, which is demonstrated on the AMD Opteron 6168 processor.

In this paper, an adopted version of CNN is used to be executed on big core according to the requirement of the object tracking and the energy limitation of the mobile platform. There has been several object detection algorithms based on CNN e.g., Single-Shot Detector (SSD) [11], YOLO [12], and Faster R-CNN [13]. A CNN algorithm for detecting and labelling disease in a radish field using a UAV is proposed in [14]. In [15], authors proposed an object detection algorithm based on CNN to detect drones. In [16], the authors present a convolutional neural network algorithm to analyze images captured from a drone to identify objects captured in the images. Authors in [17], [18] applied convolutional neural network based object detection schemes on fault diagnosis and fault tolerant control. In this work, authors used CNN to detect the working condition of an induction motor and classify it as a faulty or healthy.

Many control systems for tracking and following object have been developed. A controller system for quadcopter to follow different trajectories is presented in [19]. Authors in [20] developed a non-linear control algorithm for object tracking. An approach for target detection and wireless charging using a Hill-climbing method in presented in [21]. In [1], the authors propose a method for tracking a moving target under different paths using an IR camera. A vision based object detection and safe landing using a fuzzy logic controller is presented in [5], [22].

The previous studies show respectable outputs and performance in detecting objects. However, some of the earlier studies are based on cloud computing which is not suitable for real-time applications, because the cloud computing systems are internet-biased, and service outages are always possible and can occur for various reason. Furthermore, other studies are based on wireless communication which is also not suitable for real-time applications due to its limited coverage area, and high latency which leads to a significant degrade in performance.

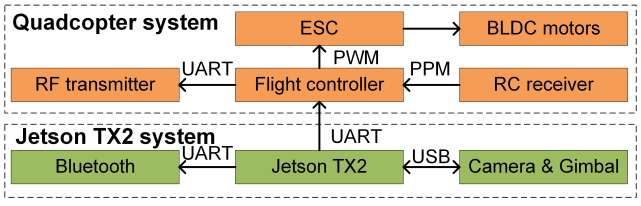


FIGURE 1. Block diagram of the overall system.

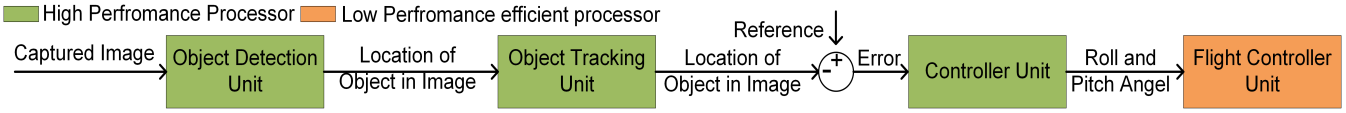


FIGURE 2. Block diagram of the proposed system.

III. SYSTEM ARCHITECTURE

Figure 1 shows the block diagram of the proposed system architecture where the little processing unit is the flight controller, and the big processing unit is Jetson TX2. The little processing unit is an ARM Cortex M4 and is responsible to adjust the attitude and stability of the quadcopter by receiving data coming from the sensors and sending commands to the motors via electronic speed controller modules (ESC). The processor receives information of roll, pitch, yaw and altitude from the 10-DoF IMU device mounted on the board. The peripheral devices around the little core facilitate the IO interface for the little core and are the ESC modules, brushless DC motor (BLDC), RF transmitter, and RC receiver.

The big processing unit in this paper is a NVIDIA Jetson TX2 processing unit that is a powerful processing unit for AI applications in terms of speed and power efficiency. Jetson TX2 consists of 256 GPU units that makes it suitable to compute the parallel tasks in neural network based applications. It also supports NVIDIA Jetpack—a complete SDK that includes the BSP, libraries for deep learning, computer vision, GPU computing, and multimedia processing. The peripherals around the big core are HD Webcam, Bluetooth, and a camera gimbal that provides the IO interface of big processing unit to the environment. The interface between the big and little processing units is a serial UART port that facilitate sending the data from big part to the little part.

IV. OBJECT DETECTION AND TRACKING ALGORITHM

In this work, a real-time feed-back-based object detection and tracking algorithm is proposed that can be adapted to be executed on the proposed heterogeneous system architecture. Figure 2 shows the feed-back-based algorithm for object detection and tracking. The visionary data from the camera is fed into the object detection algorithm to detect the target object for tracking. If the object detection part finds the target object in the scene, the location of the object will be passed to the object tracking algorithm that's responsibility is to extract the final location of the object based on the history and the probability of possible error. After finalizing the location of the target object, this location is passed to the controller that adjusts the location of the quadcopter. This process happens by comparing the observed location, i.e., the location from the Object Tracking Unit, with a reference that shows the expectation. In our case the reference is the center of the image. The Controller Unit operates based on the calculated difference of the observed location from the reference, and thereby, the actuation parameters to each motor will be adjusted.

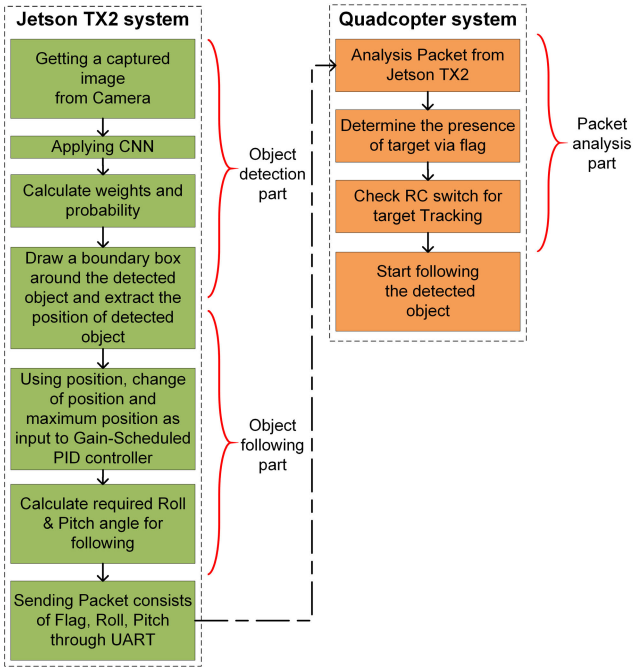


FIGURE 3. Proposed object detection and tracking parts.

In Figure 2, the camera keeps on capturing continues images and send it to the Jetson TX2. The image is given as an input to the pre-trained CNN, features are extracted from the input image and weights are calculated. Based on the estimated weights, the CNN gives an output image by drawing a boundary box around the object with the expected probability in percentage. Then the X and Y position of the detected object is given out. The position (pos) of the detected object is represented in pixel coordinates (X, Y). These coordinates is changed so that the center co-ordinates are converted from $(\frac{width}{2}, \frac{height}{2})$ to $(0, 0)$ by using Equation 1 and Equation 2. These values are used to calculate the required Roll and Pitch angle responsible for following the object.

$$X = x - \frac{width}{2} \tag{1}$$

$$Y = y - \frac{height}{2} \tag{2}$$

where X and Y are the new position in pixels, $width$ and $height$ are the resolution of the input image.

A. OBJECT DETECTION ALGORITHM

As it is shown in Figure 3, after receiving an image of an object, a CNN algorithm is applied. In this paper, a pre-trained single-shot multibox detector (SSD) model is used

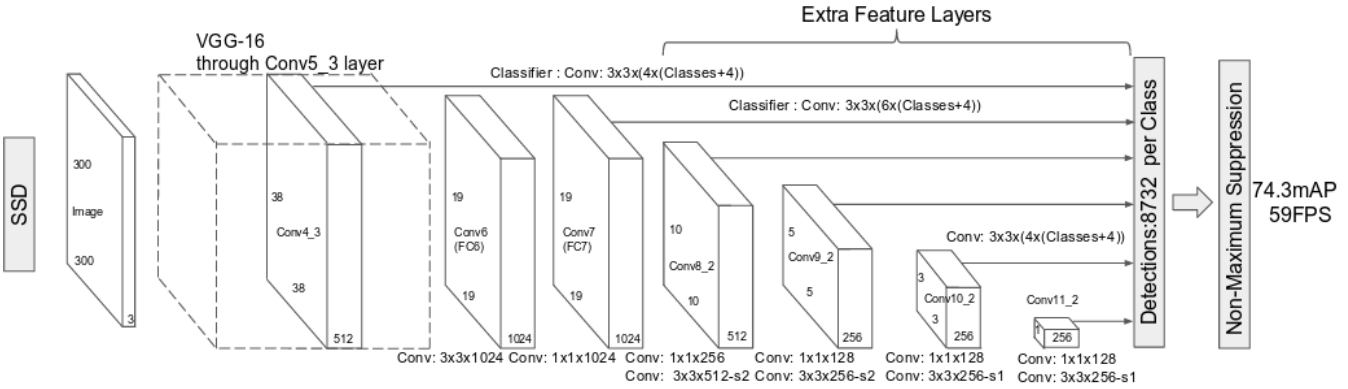


FIGURE 4. General architecture of the CNN method for object detection.

to detect the objects. Figure 4 illustrates the architecture of the SSD. 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 [11]. It can be seen in Figure 4, SSD's architecture builds on the venerable VGG-16 architecture, but discards the fully connected layers. The reason VGG-16 was used as the base network is because of its strong performance in high quality image classification tasks and its popularity for 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) were added, thus enabling to extract features at multiple scales and progressively decrease the size of the input to each subsequent layer.

After applying the SSD model, a boundary box will be drawn around the detected object. The location data of the detected target is extracted and used as an input to the object tracking algorithm to start the tracking process.

B. OBJECT TRACKING ALGORITHM

The location of the detected object is passed to the object tracking unit. The tracking unit works based on the history of the target's objects location and current location of the detected object. After pre-processing the data and filtering, the location of the object in the image will be subtracted from the expected reference and based on the results, i.e., the error, the object tracking unit tries to specify and stabilize the appropriate location of the quadcopter. To do that, two possible situations might happen upon which we define different working modes. One situation is that the error is high enough to activate the thrust of the quadcopter to change the velocity and follow the object that is called mode-out in this paper. The other situation is that the error is not that much high and the quadcopter should keep its current velocity to stabilize over the object. In mode-in the stabilization of the quadcopter is essential, while in mode-out acceleration is. Therefore, two different types of PID controller with different gains are used in each mode. The details of the implementation in each mode are explained in the following.

Mode-Out: In this mode, the current position of the quadcopter and its change of position are continuously checked to determine the required roll/pitch angles responsible for following the detected object. In this mode a Gain-Scheduled PID controller is utilized. Gain scheduling is one of the most popular approaches to nonlinear control design, as it has a better performance and stability than robust ones [23]. The Gain-Scheduled PID controller has three inputs, pos, $\triangle pos$, and $Max\triangle pos$, and one output, roll/pitch angle. The extracted position (pos) is assumed to be the error $e(t)$ between the position of the detected object and the centroid of the image since the centroid is always zero. Then, by calculating the difference between the current position and the previous position, change of position $\triangle pos$ is obtained. The maximum change of position $Max\triangle pos$ is a fixed value that is obtained by storing the maximum change of position of the moving object. Equation 3 indicates how to calculate the required roll and pitch angle for following, while Equation 4 shows how to obtain the $\triangle pos$.

$$y(t) = (1 + |\frac{\triangle pos}{Max\triangle pos}|)K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de(t)}{dt} \tag{3}$$

$$\triangle pos = pos_{cur} - pos_{prev} \tag{4}$$

where the error $e(t) = pos$, $y(t)$ is the output of the controller where in this work it represents roll/pitch angles, K_p,K_i and K_d are the proportional, Integral, and derivative gains, $\triangle pos$ is the change of position, pos_{cur} is the current position of the quadcopter, and pos_{prev} is the previous position.

As it is seen in the previous Equations, a higher $\triangle pos$ value indicates that the quadcopter moved in a higher speed since it covers a bigger distance in a fixed time (36 msec), and vice versa. This will results in increasing/decreasing the proportional gain of the PID controller which is changed according to the ratio between $\triangle pos$ and $Max\triangle pos$.

Following are examples of the operation of the quadcopter in Mode Out:

- If $\triangle pos$ is a small value, K_p will be increased by a small value too resulting in a slight increase in the quadcopter angle and speed, causing the quadcopter to move towards the center of the detected object where it will trigger Mode In.
- If $\triangle pos$ is a big value, K_p will quickly increase up to double, this will increase the angle and speed of the quadcopter so it can quickly react and keep up with the moving object.

Mode-In: In this mode, a typical PID controller is used for stabilizing the quadcopter over the detected object. Also, this mode works as a brake for the quadcopter when it comes from the Mode Out region by using a big proportional gain. Following are examples of the operation of the quadcopter in Mode Out:

- If $\triangle pos$ of the quadcopter approaching the Mode In region is big, the quadcopter angle and speed will increase. This will cause the quadcopter to quickly decrease its angle and speed to be able to stabilize above the center of the detected object.
- If the quadcopter is in the Mode In, the PID controller will be able to stabilize the quadcopter above the center of the detected object which is either fixed or moving at low speed.

C. DATA PACKET PROCESSING PART

After calculating the suitable angle required for following the object, a packet consists of pitch, roll, and a flag is sent to the flight controller every 36 msec. Once the flight controller receives the data, it will look up for the value of flag. If the value is equal to '1', it will start the tracking process. Additionally, a safety switch is provided to support the safe operation of the quadcopter. Figure 5 shows the flowchart of the object following algorithm inside the Jetson TX2, and the data packet processing inside the flight controller.

V. SIMULATION AND EXPERIMENTAL STUDIES

To run the CNN on the big part, an SSD object detector is used in this work. The SSD is able to achieve real-time and high accuracy detection by using low resolution images as an input. The SSD algorithm is implemented on an embedded Artificial Intelligence (AI) computing device of NVIDIA Jetson TX2. The CNN is trained to detect two classes, the first class contains images with an object, which can be thought as positive images, while the other class contains images with no object which can be thought as negative class. The SSD architecture is selected because it combines the performance of YOLO with the accuracy of region-based detectors so it can detect objects in real-time. The problem with SSD is that the computational time is higher than other object detection algorithms such as YOLO. Therefore, SSD is implemented with an optimization method where the computational load is divided onto CPU and GPU, resulting in a low computational

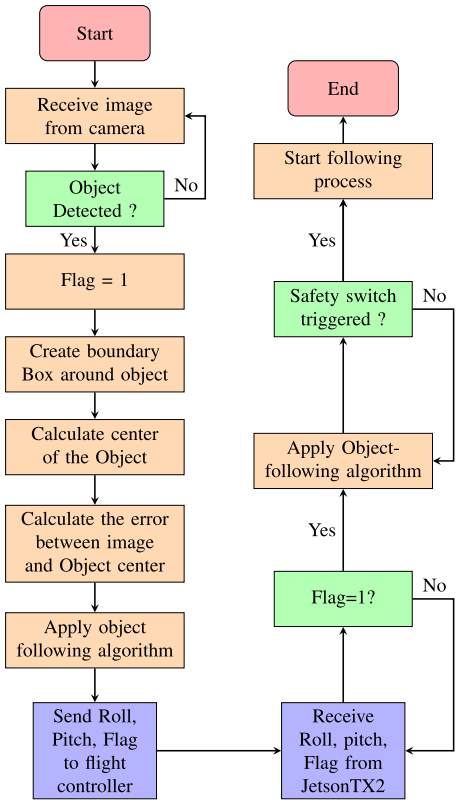


FIGURE 5. Flow chart of Object-tracking and data packet processing.

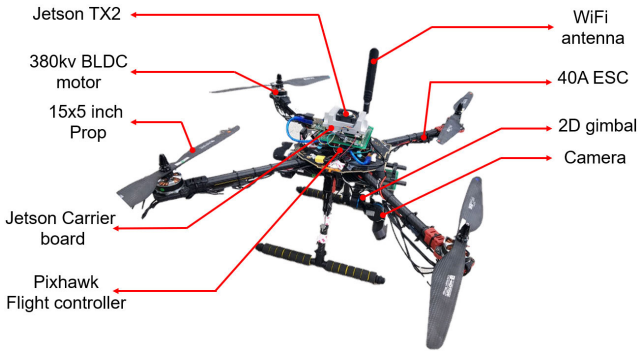
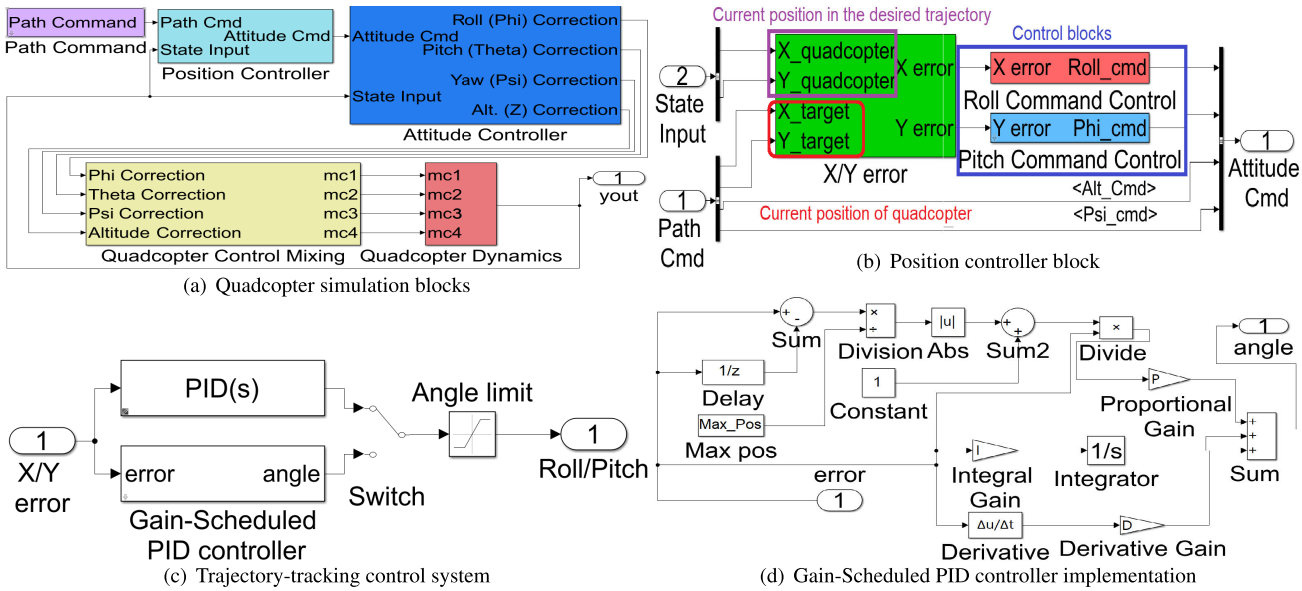


FIGURE 6. Quadcopter experimental setup.

time on the Jetson TX2, 36 ms, compared to the normal time, 111 ms. Furthermore, the object detection algorithm is combined with an object tracking algorithm based on Gain-Scheduled PID controller to follow the detected object under variable speed. The reason for choosing the Gain-Scheduled PID controller is to overcome the instability and non-linearity of the quadcopter system and thus enables it to follow a target under various speeds.

A. QUADCOPTER SETUP

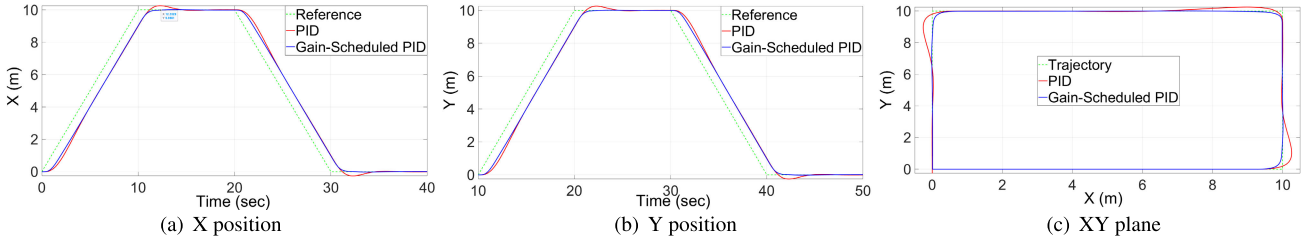
The experimental system is shown in Figure 6. It consists of a quadcopter frame with flight controller, which is based on an ARM Cortex M4 processor. It's equipped with eight PWM outputs which can support up to eight BLDC motors. The flight controller is build on ArduPilot, which is an open



(a) Quadcopter simulation blocks (b) Position controller block

(c) Trajectory-tracking control system (d) Gain-Scheduled PID controller implementation

FIGURE 7. Quadcopter simulation system.



(a) X position (b) Y position (c) XY plane

FIGURE 8. Simulation results of the two controllers in a rectangular trajectory.

source code program written in C++. The big processing unit (Jetxon TX2) is installed on a 3rd party carrier board and is connected to the little processing unit (Flight controller) through UART. A Logitech BRIO camera attached to a 2D-gimbal is connected to the TX2 through USB. This camera is able to automatically adjust the image quality to compensate for too much or too little light with High Dynamic Range (HDR) capabilities. Finally, A Bluetooth linked to the Jetson TX2 is used to send the current position of the detected object to a ground station in meters.

B. SIMULATION STUDIES

MATLAB/SIMULINK based quadcopter simulation system is used to perform a comparative study between the typical PID controller and the Gain-scheduled PID controller. This simulation system is available on MATLAB file exchange and it's free to use. The basic purpose of this system is to study the behavior of a quadcopter system and how different parameters affect the quadcopter flight.

Figure 7(a) shows the full quadcopter simulation system. In this system, the Position Controller block takes three inputs from the user (x,y,t), where 'x' and 'y' represent the coordinate's points in a Cartesian coordinate system and 't' is the time. The three inputs are stored as an array in the Path Command block. In this work a rectangular path is defined and stored in the Path Command block. The Position Controller block outputs the required angle for controlling the quadcopter over the defined path. Figure 7(b) shows the position control block, where **X/Y error** block is responsible for calculating the position error between the current position in the desired path and the current position of the quadcopter, and gives out X error and Y error. The Control blocks in the Position Controller block takes the X error and Y error and use it to calculate the desired Roll/Pitch angles required for tracking based on the outputs of the two controllers. The Control block has been modified so that the Gain-Scheduled PID controller is added beside the typical PID controller to compare between them as it is shown in Figure 7(c). In the Control blocks, the manual switch block is used to switch between the two controllers, and the saturation block is used to limit the output angle between $-15°$ and $15°$. Figure 7(d) shows the implementation of the Gain-Scheduled PID controller based on Equation 3. Figure 8(a) shows the simulation results of the two controllers in a rectangular path. In Figure 8(a) and Figure 8(b), Gain-Scheduled PID controller shows better response and faster settling time in x

TABLE 1. Initialization parameters of the object detectors.

Object detector	Size of input images	Optimizer	Batch	Momentum	Initial learning rate	Decay	Training steps
YOLO V3	416 x 416	Stochastic gradient descent	16	0.9	0.001	0.0005	80,000
Tiny-YOLO V3	416 x 416	Adam	8	0.9	0.0001	0.1	70,000
SSD MobileNet V1	300 x 300	RMSprop	24	0.9	0.004	0.95	100,000
Faster R-CNN	1024 x 600	Momentum	1	0.89	0.00003	-	90,000

and y-direction while PID controller has overshot. Figure 8(c) shows the output of the two controllers in the XY plane. The Gain-Scheduled PID controller has better performance compared to the PID controller which has overshot and longer settling time.

C. EXPERIMENTAL STUDIES

In this section, the performance of the real-time object detection based on CNN and object following based on Gain-Scheduled PID controller is described.

1) OBJECT DETECTION RESULTS

The implementation of SSD is done via TensorRT and Python, due to the availability of the most common machine learning libraries. The SSD model used in this work is an SSD MobileNet V1 which is optimized to run on embedded platforms in real-time [24]. The proposed algorithm is implemented on a Jetson TX2 and programmed using python 3.5. To test the performance of the the object detector, a custom image dataset that contains 2000 images of RC car, rectangle and a circle (positive images) on different backgrounds and 3000 backgrounds (negative images) without objects has been used for training. All images have been captured by the camera installed on the quadcopter on 1280 × 720 pixels and has been manually labeled. The images is divided into two parts, training and testing dataset. 20% of the images is used to test the network, while the 80% is used to train the dataset. A dropout ratio of 0.8, kernel size 3×3 and a box-code size set to 4 is used in this network. The root mean square propagation (RMSprop) optimization algorithm is used for optimizing the loss functions trained for 100,000 steps using the following parameters; a learning rate of 0.004, decay factor 0.95, and decays at an interval of 80,0720 steps. The training was carried out on a desktop with an AMD Ryzen 7 2700X 3.70 GHz, 16 GB RAM, and NVIDIA GTX 1080Ti. Figure 9 shows the detection results of the object detection algorithm, the position of each object and the calculated fps. Once the object is detected, a boundary box will be drawn around the object, and the center of the detected objects will be extracted to be used for the object following algorithm.

To evaluate the performance of the proposed object detector on the Jetson TX2, another three models have been trained using the same dataset and same desktop. Table 1 shows the default initialization parameters of the four models used for training the dataset, while Table 2 shows their comparison results.

TABLE 2. Object detection models comparison.

Object Detection Models	Accuracy	Power consumption	FPS
YOLO V3	90%	18.11 Watt	4.1
Tiny-YOLO V3	68%	16.01 Watt	17
SSD MobileNet V1	81%	11.1 Watt	25.2
Faster R-CNN	75%	18.64 Watt	1.9

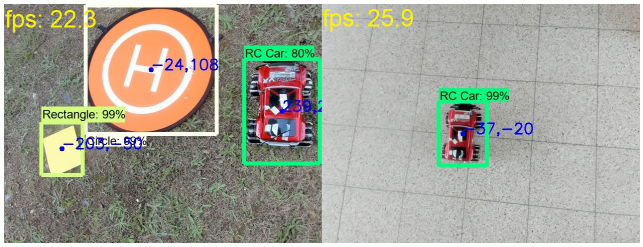


FIGURE 9. Object detection algorithm output.

As shown in Table 2, the YOLO V3 has the highest accuracy, it consumes a lot of power and the processing speed (4.1 fps) was not up for practical use. Therefore, a Tiny-YOLO V3 has been used instead of the YOLO V3 as it can achieve higher fps (17 fps). However, the Tiny-YOLO V3 is not good for practical use because of its high power consumption and low accuracy. Moreover, Faster R-CNN was able to get higher accuracy compared to the Tiny-YOLO v3. However, It has a higher power consumption compared to the previous two object detectors, and a much lower fps (1.9), which make it unsuitable for real-time detection applications. Therefore, an SSD MobileNet V1 is used instead of the other two models, as it consumes less power and achieves higher fps.

The time it takes for the object detector to process a frame and give output on the detected object is 36 msec which is 7 times faster than the YOLO V3, 14 times higher than Faster R-CNN, and 1.7 time faster than the Tiny-YOLO V3. The reduction in the computational time besides the low power consumption of this model, which is 63.15% less than YOLO V3, 67.92% lower than Faster R-CNN, and 44.23% less than Tiny-YOLO V3, made it feasible to implement it on a system like a quadcopter where response time is very critical parameter to consider. Furthermore, the SSD MobileNet V1 accuracy which is slightly less than the YOLO V3 made it the best option besides the computational time to to be used for real-time object detection. In this 36 msec, the frames received though camera are processed and the CNN gives the output in form of an image with the position of the detected object in pixels as shown in Figure 9.

2) OBJECT TRACKING RESULTS

To verify the feasibility of the proposed algorithm, several experiments were performed under variable speed of the detected target to obtain the optimal values for the parameters gains that is used for object-tracking algorithm. The experimental setup shown in Figure 6 is composed of quadcopter frame with a flight controller connected to a Jetson TX2 through UART communication. The proposed object detection and tracking algorithm is implemented on the Jetson TX2 and executed every 36 msec. Initially the quadcopter is flying at an altitude of 2 meters, once the RC car is detected, the real-time object detection is triggered and a boundary box is drawn around the moving RC car based on our proposed object detector. Afterwards, the location of the RC car and its change of position are extracted and are passed to the object-tracking algorithm based on our developed Gain-scheduled PID control. The object-tracking algorithm works differently based on the location of the detected object as been discussed in section IV.B. Finally, the object-tracking algorithm outputs the required values of roll and pitch angles that are responsible for tracking and stabilizing the quadcopter over the moving RC car. These values is sent to the flight controller every 36 msec, where the flight controller uses these values for the tracking process.

This section covers the following; 1) a vision-based algorithm is used to calculate the absolute position of the quadcopter in real-time, 2) the proposed algorithm is compared to a developed PID controller that is used for Human-follow [25], and how much each algorithm takes to be executed, 3) the proposed algorithm is compared to the previous mentioned one to track the RC car in a rectangular path under variable speed (2 m/s and 4 m/s), and the results of their performance are carried out.

For calculating the absolute position of the quadcopter in meters, a vision-based algorithm is used to obtain the absolute position between the detected target and the quadcopter. The algorithm uses the pinhole camera model and the captured images to build a metric map. The coordinate of a point in 3D (X, Y, Z) can be computed from its projection pixel in the 2D (x,y) image and the projection matrix. Firstly, a reference object is used to measure the pixels per metric ratio. In the algorithm, an RC car with a 39cm width and 22cm height is used as a reference object. An image of the reference object is captured from a defined height of 2 meters using the camera attached to the quadcopter. Then, the reference object is detected in the image, based on a color detection algorithm, where a contour technique and a threshold is used to define the boundaries of the detected object. Thus, the width and height of the reference object in pixels are obtained. The pixels per metric can be calculated as follow:

$$p = \frac{h}{w} \tag{5}$$

where p denotes the pixels per metric ratio, h and w are the reference object width in pixels and metric, respectively. The distance between the quadcopter and the detected target is computed as shown in Equation 6 and Equation 7:

$$\hat{x} = \frac{A_1 \times x + B_1}{(C(x-100)^2 + D(y-160)^2 - 1)} \tag{6}$$

$$\hat{y} = \frac{A_2 \times y + B_2}{(C(x-100)^2 + D(y-160)^2 - 1)} \tag{7}$$

where $\hat{x}$ and $\hat{y}$ are the distance in meter, x and y are the distance in pixels. As shown in the previous Equations, each Equation has four unknowns, therefore four known points are used to generate four Equations for each axis to identify these unknowns. The identified parameters are obtained by using regular simultaneous equation solving methods, such as substitution and elimination.

The experimental target is moving in a rectangular path under two different speed, 2 m/s and 4 m/s. Figure 10 and Figure 11 show the results of the performed trials using our object following algorithm and the PID controller that was developed in [25] to follow a human based on CNN detection. Table 3 shows the average time for capturing the image, perform object detection, process the info using our object following algorithm and the PID controller developed in [25]. As can been seen in Table 3, the PID controller is slightly less than Gain-Scheduled PID controller (1.5 ms difference).

TABLE 3. Timing comparison (Average time).

	PID	PID Gain-Scheduled
Capturing image (ms)	2e-5	1.8e-5
Object detection (ms)	31.3	31.8
Object tracking (ms)	2.9	3.9
Total time (ms)	34.2	35.7

Figure 10 compares between the Gain-Scheduled PID controller and the PID controller under 2m/s of the target speed. Figure 10(a) shows that Gain-Scheduled PID controller has a slightly better response in X-position than PID controller, while in Figure 10(b), PID controller shows feeble response and stability in the Y-position. Figure 10(c) depicts the performance of the quadcopter in the XY plane under 2m/s of the target speed. As seen in the Figure, the PID controller has poor performance, poor stability and it fails to reach the center of the detected target.

Figure 11 evaluates the performance of the Gain-Scheduled PID controller and the PID controller under 4m/s of the target speed in a rectangular trajectory. Figure 11(a) shows that PID controller has very bad response and stability in X-position compared to the Gain-Scheduled PID controller, while in Figure 11(b), PID controller has bad response and overshoot. Figure 11(c) demonstrates that under 4m/s of the target speed, Gain-Scheduled PID controller shows good performance and stability compared to the PID controller.

According to the previous evaluation of the experimental results, the Gain-Scheduled PID controller has proven that it has better response, and shorter settling time compared to the typical PID controller. Although PID controller can track the detected target under 2m/s, it shows very poor performance

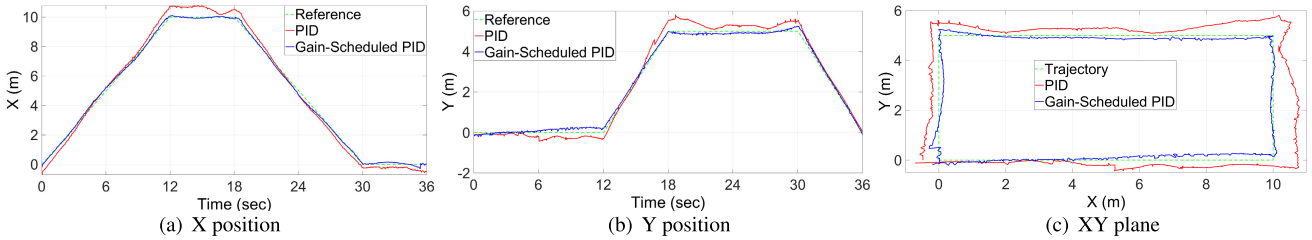


FIGURE 10. Controller result in a rectangular trajectory under 2m/s.

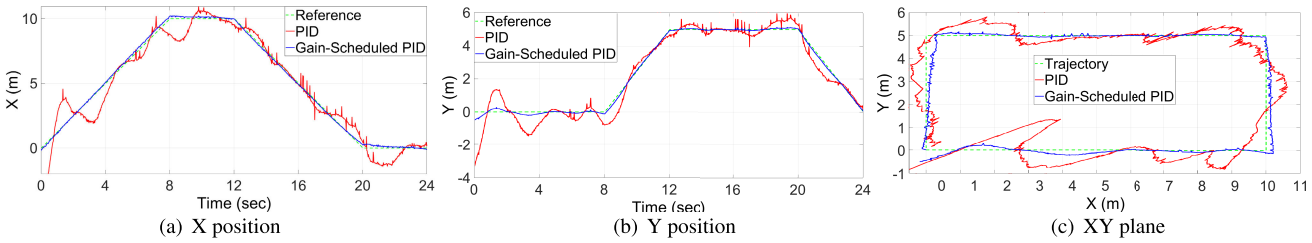


FIGURE 11. Controller result in a rectangular trajectory under 4m/s.

under 4m/s because of its fixed gains. In order to get the best performance from a PID controller, gains need to be changed continuously according to the speed of the target. Therefore, a Gain-Scheduled PID controller is used to overcome the problems of the typical PID controller. Moreover, for further improving the performance of the Gain-Scheduled PID controller, a typical PID controller has been used to brake the quadcopter while it's approaching the middle region as discussed in section IV.B, this will lead to increase in the response and performance of the quadcopter especially in the sharp turns or sudden change of target speed as shown in Figure 10 and Figure 11.

VI. CONCLUSION

In this work, an approach for implementation of a real-time object detection and tracking system for a quadcopter based on CNN have been proposed. The proposed system is implemented on an embedded computer. The object detection algorithm is based on CNN. An SSD model is used to detect the moving object, draw a boundary box around it then extract the center positions of the detected object. Object tracking algorithm based on Gain-Scheduled PID controller is established to follow the detected object under variable speed. The presented algorithm can be used in many applications such as search-and-rescue, tracking a specific object, and providing aerial footage of sports events. Several trials are performed. The experimental results shows that the object detection algorithm is able to detect and classify objects with high accuracy, less power consumption, and high fps, and the object tracking algorithm is able to follow and track the detected object under variable speed. In addition, using this technique to track multiple/specific object might prove an important area for future research. Moreover, developing the proposed work to estimate the distance between the quadcopter and the detected target based on vision-based techniques, and landing safely on a moving/fixed object without the use of an optical ranging sensor is also an important application for future work.

REFERENCES

[1] 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.

[2] S. A. S. Mohamed, M.-H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, "A survey on odometry for autonomous navigation systems," *IEEE Access*, vol. 7, pp. 97466–97486, 2019.

[3] S. A. Mohamed, M.-H Haghbayan, M. Rabah, J. Heikkonen, H. Tenhunen, and J. Plosila, "Towards dynamic monocular visual odometry based on an event camera and IMU sensor," in *Intelligent Transport Systems. From Research and Development to the Market Uptake. INTSYS* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering) vol. 310, A. Martins, J. Ferreira, and A. Kocian, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-38822-5_17.

[4] S. Mittal and J. S. Vetter, "A survey of CPU-GPU heterogeneous computing techniques," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–35, Jul. 2015.

[5] 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, Autom. Syst.*, vol. 16, no. 6, pp. 3013–3025, Dec. 2018.

[6] F. Patrona, I. Mademlis, A. Tefas, and I. Pitas, "Computational UAV cinematography for intelligent shooting based on semantic visual analysis," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 4155–4159.

[7] H. Hoffmann, J. Eastep, M. D. Santambrogio, J. E. Miller, and A. Agarwal, "Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments," in *Proc. 7th Int. Conf. Autonomic Comput. (ICAC)*. New York, NY, USA: ACM, 2010, pp. 79–88, doi: 10.1145/1809049.1809065.

[8] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp. 449–460.

[9] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Proc. 50th Annu. Des. Autom. Conf. (DAC)*, 2013, p. 174.

[10] K. Ma and X. Wang, "Pgcapping: Exploiting power gating for power capping and core lifetime balancing in cmps," in *Proc. 21st Int. Conf. Parallel Archit. Compilation Techn.*, 2012, pp. 13–22.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.
[12] 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.
[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[14] 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., Informat. Syst.*, May 2018.
[15] M. Saqib, S. D. Khan, N. Sharma, and M. Blumenstein, "A study on detecting drones using deep convolutional neural networks," in *Proc. 14th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug. 2017, pp. 1–5.
[16] A. Rivas, P. Chamoso, A. González-Briones, and J. Corchado, "Detection of cattle using drones and convolutional neural networks," *Sensors*, vol. 18, no. 7, p. 2048, Jun. 2018.
[17] Y. Wu, B. Jiang, and N. Lu, "A descriptor system approach for estimation of incipient faults with application to high-speed railway traction devices," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 10, pp. 2108–2118, Oct. 2019.
[18] Y. Wu, B. Jiang, and Y. Wang, "Incipient winding fault detection and diagnosis for squirrel-cage induction motors equipped on CRH trains," *ISA Trans.*, to be published.
[19] M. Rabah, A. Rohan, Y.-J. Han, and S.-H. Kim, "Design of fuzzy-pid controller for quadcopter trajectory-tracking," *Int. J. Fuzzy Log. Intell. Syst.*, vol. 18, no. 3, pp. 204–213, 2018.
[20] A. Joukhadar, M. AlChehabi, C. Stöger, and A. Müller, "Trajectory tracking control of a quadcopter uav using nonlinear control," in *Mechanism, Machine, Robotics and Mechatronics Science*. Cham, Switzerland: Springer, 2019, pp. 271–285, doi: doi.org/10.1007/978-3-319-89911-4_20.
[21] A. Rohan, M. Rabah, F. Asghar, M. Talha, and S.-H. Kim, "Advanced drone battery charging system," *J. Electr. Eng. Technol.*, vol. 14, no. 3, pp. 1395–1405, Feb. 2019.
[22] 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, Jun. 2018.
[23] V. Veselý and A. Ilka, "Gain-scheduled PID controller design," *J. Process Control*, vol. 23, no. 8, pp. 1141–1148, Sep. 2013.
[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[25] A. Rohan, M. Rabah, and S.-H. Kim, "Convolutional neural network-based real-time object detection and tracking for parrot AR drone 2," *IEEE Access*, vol. 7, pp. 69575–69584, 2019.

MOHAMMED RABAH received his B.S. degree in Electronics and Telecommunication Engineering from the AL-SAFWA High Institute of Engineering, Egypt in 2015. He completed his MSc in Electronics and Information Engineering from Kunsan National University, South Korea in December 2017. Currently, he is a PhD candidate and works as a Research Assistant at Factory Automation & Intelligent Control Laboratory, Department of Electronics and Information Engineering, Kunsan National University, South Korea. His research Interests includes UAV's applications, autonomous systems, intelligent control systems, and deep learning.

ALI ROHAN received his B.S. degree in Electrical Engineering from The University of Faisalabad, Pakistan in 2012. From 2012 to 2013, worked as a Development Engineer at Niagara group of Industries, Pakistan. From 2013 to 2015, worked as a Project Engineer for Circle Club, Pakistan. From 2015 to 2016, worked as a Project Manager for Steam Masters, Pakistan and also as a Lecturer at the department of Electrical and Telecommunication Engineering, Government College University Faisalabad, Pakistan. In 2018, completed his M.S. degree in Electrical, Electronics and Control Engineering from Kunsan National University, South Korea. In February 2020, he completed his Ph.D. degree in Electrical, Electronics and Control Engineering from Kunsan National University, South Korea. His Research Interests includes Machine learning, AI, UAV's, Power Electronics, Fuzzy Logic, EV system, Flywheel Energy Storage System.

MOHAMMAD-HASHEM HAGHBAYAN received the B.A. degree in computer engineering from the Ferdowsi University of Mashhad, the M.S. degree in computer architecture from the University of Tehran, Iran, and the Ph.D. degree (Hons.) from the University of Turku (UTU), Finland. Since 2018, he has been a Postdoctoral Researcher with the Department of Future Technologies, UTU. He has published 42 peer-reviewed publications in international conferences and journals. His research interests include machine learning, autonomous systems, high-performance energy-efficient architectures, and on-chip/fog resource management.

JUHA PLOSILA received the Ph.D. degree in electronics and communication technology from the University of Turku (UTU), Finland, in 1999. He is currently a Professor in autonomous systems and robotics with the Department of Future Technologies, UTU. He is also the Head of the EIT Digital Master Programme in Embedded Systems, EIT Digital Master School (European Institute of Innovation and Technology). He represents UTU in the Node Strategy Committee of the EIT Digital Helsinki/Finland node. He has a strong research background in adaptive multiprocessing systems and platforms, and their design. This includes, e.g., specification, development, and verification of self-aware multiagent monitoring and control architectures for massively parallel systems, machine learning, and evolutionary computing-based approaches, as well as application of heterogeneous energy-efficient architectures to new computational challenges in the cyber-physical systems and the Internet-of-Things domains, with a recent focus on fog/edge computing (edge intelligence) and autonomous multidrone systems.

SUNG-HO KIM received the B.S., M.S., and Ph.D. degrees in electrical engineering from Korea University, in 1984, 1986, and 1991, respectively, and the Ph.D. degree from Hiroshima University, Japan, in 1996. 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.