Object detection for drones on Raspberry Pi potentials and challenges

Tran Quang Khoi*, Nguyen Anh Quang2, Ngo Khanh Hieu1,3

1 Department of Aerospace Engineering, HCMUT, VNU-HCM
2 Technical R&D Manager at SII Canada
3 VNU-HCM Key Laboratory for Internal Combustion Engine

*khoi.tq.aero@gmail.com

Abstract. The paper presents preliminary research results about implementing an object detection program on a Single Board Computer. These results are used later to develop applications for drones. The object identification program is developed in Python using the TensorFlow library. The authors have succeeded in implementing and testing this object identification module using the artificial neural network model SSDMobileNet V2 on the Raspberry Pi 3B+. The results in this paper demonstrate the potential of this module for further development in the future. Based on the simulation and real-world results, the authors showed that a good outcome is achievable with limited resources for the AI module. Along with a high-precision object detection feature, this module can also estimate the distance and velocity of the "human" object with good accuracy. Besides, the paper also proposes several solutions to increase the performance and most importantly, the real-time feature of the developed module.

Index Terms— Object detection, Raspberry Pi, SSDMobileNet, Computer vision, Artificial neural network, Convolutional Neural Network.

1. Introduction

Unmanned Aerial Vehicle (UAV) is one of the recent rising research trends, in particular the topics related to Artificial Intelligence (AI)/Machine Learning (ML). The applications of the UAV + AI/DL is being investigated and developed for different industries such as art, transportation, security, etc. For the next generation of UAVs, one of the basic requirements is the ability to localize and identify the interested objects, which is fundamental for more advanced features such as real-time decision-making and high-precision indoor/outdoor navigation.

Even though object localization algorithms can offer a good accuracy, as a trade-off, they usually require powerful computational hardware. This is a real challenge for UAV designers and developers because of the constraints related to Size, Weight, and Power for these extra payloads. In the recent years, with the outstanding development in the computer sciences and other related domains, a new solution known as Single Board Computer (SBC) has been introduced. Thanks to its lightweight yet powerful processing capability (sometimes even with an external on-board graphic processing unit), the single-board computers (computer boards) have gradually overcome the above problem, becoming the best solution for both commercial and research-based UAV. Along with the new advances in hardware,
the development in the field of AI/Machine Learning in recent years has also produced more lightweight, faster, and more reliable neural network algorithms and models, reducing the requirements for the computational resources (memory, clock rate, parallel computation, etc.).

Thanks to the above factors, the implementation of an on-board real-time object detection module using neural networks has become feasible. Some examples for these developments and the related researches can be referred in [1-2]. In this paper, the authors present their recent work on implementing an object detection program capable of estimating the distance and velocity of objects, then performing experiments to determine the applicability of this technology to small drones. The core hardware of this work is a cost-of-the-shelf yet popular Raspberry Pi 3B+ [3], and the AI module is implemented using Tensorflow library developed by Google [4]. Figure 1 below shows an example for the outputs of our work.

The next sections of this paper are structured as follows: In section 2 we make an overview about the computer vision and artificial neural network; Section 3 will show the implementation of the object detection module; Section 4 presents the experiments to determine performance of the integrated solution, along with the results of these tests. As described in detail in this Section, the authors have successfully implemented the program on the Raspberry Pi 3B+. The distance estimation and velocity estimation ability of the program give a good result with low errors (see Fig. 1). After that, Section 5 presents the potentials and challenges for the further research continuing this work in the real-time aspect of the module. Finally, Section 6 will present different future works based on the obtained results presented in this paper.

Figure 1. Object detection and distance estimation, velocity estimation results.

2. Background

2.1. Artificial neural network and object detection
The artificial neural network (or neural network for short) is a mathematical model that was introduced in 1943 to simulate the way in which human biological neurons work. The neural network is made up of basic units, i.e. neurons, which is in fact a group of different basic operations such as Multiplication, Addition, and linear/non-linear Activation A pioneer in the study of neurons is logic unit model made by Warren McCulloch and Walter Pitts in 1943 and the perception model introduced by Frank Rosenblatt in 1957 [5]. Although inspired by biological neurons, artificial neurons and artificial neural networks are completely different in nature. Human brain has about 100 billion neurons working in parallel meanwhile artificial neural networks are essentially just mathematical functions that are executed sequentially by computers. Studies of artificial neural networks favour math and technology rather than biology, although there are some mathematical models for ideas are made by observing how the brain works (see Fig. 2).

Mathematical representation of the neuron output:
Figure 2. Artificial neuron model.

The training of artificial neural networks means finding the appropriate set of weights and deviation coefficients for each neuron in the network so that the final output is as close as possible to the input references (i.e. expectation of the trainer), hence minimizing the objective function (cost function). To train neural networks we use back-propagation techniques in conjunction with optimization algorithms such as Gradient Descent [6].

To solve object detection and identification problems in computer vision, we use a convolutional neural network. The convolutional neural network has a different architecture than a normal neural network. The latter transforms input through a series of hidden layers. Each layer is a set of neurons and fully connected with the neurons on the previous layers. The final layer will represent the prediction of the network. Meanwhile, the convolution neural network (CNN) has neurons that are not fully connected to each neuron of the next layer, but only to a small region (some neurons). This section of the CNN is called the feature extractor, its function is to detect and extract different features in the image. Finally, a fully-connected flatten output layer will show the prediction result [7].

From the fundamental convolutional neural network AlexNet in 2012, there have been different convolutional neural network architectures developed so far (for example SSD and YOLO as shown in Fig. 3). Each network architecture has its own advantages in processing speed or resource consumption or accuracy (see Fig. 4).
2.2. Distance and velocity measurement

The distance estimation from the camera to the subject in the image is based on the image’s formation taken through thin lenses (Fig. 5):

![Image 1](image1.png)

**Figure 4.** CNN structure for Object detection [2].

**Figure 5.** Image captured through a thin lens.

Where:
- $f$: camera lens focal (mm)
- $d$: distance between camera and object (mm)
- $D$: size of object (mm)
- $D_{prj}$: size of object in image (pixel)

The relationship between quantities is as follows:

$$d\ (\text{mm}) = f\ (\text{mm}) \cdot \frac{D}{D_{prj}} \quad (2)$$

$$D_{prj}\ (\text{mm}) = D_{prj}\ (\text{pixel}) \times \text{pixel\_to\_mm\_ratio} \quad (3)$$

$$\text{pixel\_to\_mm\_ratio} = \frac{D_{prj}\ (\text{pixel})}{D_{mm}} \quad (4)$$

In order to calculate the velocity of an object, it is necessary to obtain at least 2 frames has the same object and follow the steps (see Fig. 6):

a. Determine the position of the object in the image and find the box center coordinates in the first frame.
b. Determine the position of the object in the image and find the box center coordinates in the second frame.
c. Calculate the differences in position of box center between 2 frames.
d. Calculate the time difference between 2 frames.
e. Calculate the velocity by taking the ratio of the change in position of box center and time difference between 2 frames.
Figure 6. Bounding box center position change between 2 video frames

*Note: No filters are applied to predict the velocity and position of the bounding box center due to research purpose only.

2.3. Raspberry Pi 3B+

For this work, we focus on the Raspberry Pi (RPi) 3B+, the latest generation of single-board computers of the Raspberry Pi Foundation in the UK at that time. Nowadays, there are thousands of diverse applications based on Raspberry Pi, ranging from Internet of Things to robotics and Industry 4.0 [8][9]. Some important parameters of the RPi 3B+ can be found in Table 1 below.

| Table 1. Raspberry Pi 3B+ configuration |
|----------------------------------------|
| **CPU**                                | Broadcom BCM2837B0 quad-core A53  |
|                                        | (ARMv8) 64-bit @ 1.4GHz            |
| **RAM**                                | 1GB LPDDR2 (900 MHz)               |
| **Connection options**                 | Wifi 802.11 n, Bluetooth 4.1, 40-pin GPIO |
| **Power consumption**                  | 800 mA, 5 V (4 W)                  |
| **Total weight**                       | 50 gr                               |
| **Graphic card**                       | None, but support OpenGL for advanced image processing feature. |

As can be seen from the Table above, RPi 3B+ is very energy-efficient and lightweight. Furthermore, thanks to the new powerful ARM processor, RPi 3B+ becomes a good candidate for the onboard processor for UAV applications.

3. Implementation

3.1. TensorFlow API

TensorFlow Object Detection API is designed specifically to build, train, and deploy different object detection models based on image databases such as COCO, Kitti, Open Images, AVA v2.1 and iNaturalist Species Detection. Each of these models has been pre-trained by computer scientists at Google for various artificial neural network meta-structures such as RCNN, SSD, Faster RCNN, SSDLite, etc. Each neural network meta-structure has different characteristics and depending on the needs, users can choose the appropriate model to use [10]. In addition, TensorFlow also allows users to train their own object detection neural network (from scratch or based on the pre-trained model) to suit each industrial specification. This API provides functions so that users can create their own database and conduct neural network training. However, this job is very time consuming and needs computers with powerful Graphical Processing Unit (GPU) to train or to fine-tune the neural networks.
3.2.  COCO dataset

COCO (Common Object In Context) developed by Microsoft is a fairly large database with 330,000 images in 91 categories of common objects. Over 200,000 of them are labeled and ready for artificial neural networks training [3]. In particular, the most photographed subjects are people (Fig. 7).

![Number of images in each category of COCO](image)

**Figure 7.** Number of images in each category of COCO [3]

3.3. Choose artificial neural network model

Since the algorithm must be low in resource consumption, the author prioritized the processing time of TensorFlow's existing artificial neural network models. As illustrated in Fig. 8, comparison results show that SSDlite_MobileNet_V2 model (SSDlite convolutional neural network architecture using MobileNet feature extractor version 2) has best performance (0.81) with the output is a bounding box, this model is chosen to implement in the object detection module presented in this paper.

![CNN models comparison](image)

**Figure 8.** CNN models comparison

4. Experiment

To investigate the program’s performance on object detection accuracy, distance estimation accuracy and velocity estimation accuracy, we perform 2 types of test: static image and video.

The experimental flow chart is as follow:
4.1. Static image tests

This test focuses on evaluating the ability of the program to detect objects for still images in order to obtain the optimal distance between the camera and the object where the program gives the best results (lowest error in both vertical and horizontal direction).

Experiment process: (see Fig. 9 – 11)

a. Take photos of a person at distances $d = 666f$ (H 2 m), $1,333f$ (H 4 m), $2,000f$ (H 6 m), $2,666f$ (H 8 m), $4,000f$ (H 12 m).

b. At each distance, take pictures at horizontal positions equal to 0.5 m from the center to the left and right.

c. Calibrate the program’s parameters at a distance of $d = 666f$ (H 2 m) so that the distance estimation of the program is best.

d. Repeat step c for the remaining cases.

e. Make conclusions about the effects of object position in horizontal and distance to program's detection accuracy.

f. Make conclusions about the program’s distance estimation accuracy at different distances.

The subject is a human photo (Fig.9). A simple grey background (Fig.10) is generated using Photoshop to compare the performance of the program with real life background (Fig .11) (complex background).
Figure 9. Experiment subject

Figure 10. Subject in complex background

Figure 11. Subject in a simple background
5. Results

The program can identify objects up to 12 m (Fig. 13). At a longer distance, i.e. smaller object; the precision (detection and labeling) of the program degrades gradually. The distance estimation accuracy is relatively good in the range of 2 m - 6 m (error not exceeding 10%).

Figure 12 shows the differences in horizontal axis has a small impact on the accuracy of object detection and distance estimation. The effective range of the program is from 4 m to 6 m. The maximum detection distance of the program is 12 m. And the minimum estimated distance of the program is 2 m.

Figure 14 and 15 show that in the effective range of the program (4 m – 6 m), object’s horizontal position has little effect on the program’s accuracy.

5.1. Video tests

This experiment aims at evaluating the object detection ability of the program to determine the effect of distance and camera shutter speed to the accuracy of the recognition, distance and velocity estimation of the program as well as the program real-time performance. The test object (human image - size 19 cm) is mounted on a single pendulum which oscillates with a magnitude of 5 degrees, the pendulum length (to the midpoint of the object) is 0.52 m. Friction bypassed, parameters are calculated as follows (Fig. 16):

- The oscillation cycle: \[ T = \frac{2\pi}{\omega} = 2\pi \sqrt{\frac{l}{g}} = 1.4466(s) \]  (5)
- Maximum speed at equilibrium point: \( v_{\text{max}} = \omega S_0 = \omega \alpha_0 l = 0.1971 \left( \frac{m}{s} \right) \) (6)

The testing system is placed at different distances (corresponding to the distance of 4 m, 6 m, 8 m in real world) and oscillates (maximum velocity at the equilibrium position equal to 1.58 m/s in real world). The author recorded the system movement for 1 minute at each distance and each shutter speed value, extracted the image from the video recorded. The extracted images are fed to the program to evaluate: accuracy of identification, distance and velocity estimated compared to real world (Fig. 17).

Figure 16. Experiment single pendulum

Figure 17. Result photos extracted from 10 FPS video at distance of 0.5 m

Figure 18. Effect of shutter speed to object detection accuracy

Figure 18 shows that at low shutter speed, the program’s object detection accuracy is lower. This is due to image blur effect on a moving object. At the shutter speed of 1/30 second or faster, the program’s accuracy nearly the same.
Figure 19. Effect of shutter speed to distance estimation accuracy

Figure 19 shows that the shutter speed has little effect on distance estimation accuracy thanks to the method to estimate the distance is based on object recognition. As long as the program recognizes the object, it will estimate the distance based on known sizes of the object.

Figure 20. Effect of shutter speed to velocity estimation accuracy

Figure 20 shows that velocity estimation error reaches maximum value at the longest distance for a shutter speed of 1/10 second. For shutter speeds of 1/30 second, the error does not exceed 9% at all distances. For shutter speeds of 1/50 second, the error does not exceed 8% at all distances. When increasing the shutter speed, the velocity estimation error tends to decrease.

The estimated error of the program is within an acceptable range (not exceeding 10%) at shutter speeds of 1/30 second and 1/50 second. Shutter speed of 1/10 second gives a large velocity estimation error (nearly 24%) and in some cases it is not possible to identify the object, so the distance and velocity cannot be estimated.

5.2. Real-time performance

The processing time of the program determines the capability to use in real time. Real-time is defined as the ability to process all images that the camera obtains in a shorter or equal amount of time that the camera needs to capture the images. For example, a camera can obtain 10 frames per second (10 FPS), the processing time of the program for each image must be less than or equal to 0.1 second to be considered real-time.

The real-time performance is calculated by taking 1 second divided by the total time the program processes all frames obtained from the camera in 1 second.
Figure 21. Effect of value of frames per second (FPS) obtained by the camera to the program's real-time performance

Results: (see Fig. 21)

FPS (frames per second) value has a strong influence on the program's real-time performance. As FPS increases, the real-time performance of the program decreases, because the program needs to handle more tasks.

At the lowest FPS (10 fps), the system takes approximately 14 seconds to process 1 second of video. This is due to the hardware limitations of Raspberry pi 3B + computers.

Similarly, at 30 fps, the program's real-time processing capability only reached 2.29% of requirement. This shows that the program cannot be used to handle situations that require rapid response.

The program can only handle a maximum of about 0.71 FPS. The real-time response of the program decreases linearly as the FPS value of the camera increases.

6. Conclusions

The program can run on Raspberry Pi 3B+ smoothly without any problem and shows good accuracy in object detection and identification for both still images and videos. The estimated distance and velocity accuracy is also quite good, which can be used in different features such as monitoring/avoiding obstacles.

Real time performance is bad even with one of the simplest and well-known object detection models. This is a real challenge because in many situations, the drone must act quickly and need fast response from the program to make decisions.

7. Future work

Based in the experience and obtained results in this study, some of the important future work are: (1) Improve the algorithm, using more compact and low computational resource neural network models to increase the program's real-time performance, e.g TensorflowLite and quantized model; (2) Develop the ability to track and estimate the velocity and distance of many objects simultaneously; (3) Make use of graphic card processing power on single-board advance computers like NVIDIA’s Jetson Nano to improve real-time performance; and (4) Integrate the object detection program to a fully operational obstacle avoidance system for UAVs, especially quadrotors.

References

[1] Liu, Wei, et al, "SSD: Single shot multibox detector," in European conference on computer vision, 2016.
[2] Mathworks.com, "MathWorks.com," MathWork, [Online]. Available: https://it.mathworks.com/discovery/convolutional-neural-network-matlab.html.
[3] Schnerr, G. H. and Sauer, J., "Physical and Numerical Modeling of Unsteady Cavitation
Dynamics," in *International Conference on Multiphase Flow*, New Orleans, USA, 2001.

[4] S. Boyce, "AI-on-raspberry-pi-with-the-intel-neural-compute-stick," 31/1/2019. [Online]. Available: https://hackaday.com/2019/01/31.ai-on-raspberry-pi-with-the-intel-neural-compute-stick/..

[5] "Raspberry pi," Raspberry Pi Foundation, 2019. [Online]. Available: https://www.raspberrypi.org/..

[6] Wikipedia, "TensorFlow," Google, 2019. [Online]. Available: https://vi.wikipedia.org/wiki/TensorFlow..

[7] Nicole Sebe; Ira Cohen; Ashutosh Garg; Thomas S. Huang, *Machine Learning in Computer Vision*, Springer Science & Business Media, 2005..

[8] Long. L. N., Gupta. A, "Scalable massively parallel artificial neural networks," *Journal of Aerospace Computing, Information, and Communication*, pp. 3 - 15, 2008..

[9] Wikipedia, "Backpropagation," [Online]. Available: https://en.wikipedia.org/wiki/Backpropagation..

[10] viblo.asia, "Măng No-ron tích chấp," [Online]. Available: https://viblo.asia/p/mang-no-ron-tich-chap-p1-DZrGNNjPGVB..

[11] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012. [Online]. Available: https://www.cs.toronto.edu/~fritz/absps/imagenet.pdf. [Accessed 3 3 2019]..

[12] Mlab.vn, "Raspberry-pi-la-gi," [Online]. Available: http://mlab.vn/11025-raspberry-pi-la-gi-gioi-thieu-cac-ung-dung-cua-raspberry-pi-3.html..

[13] Reglisse44, "Instructables.com - The drone pi," [Online]. Available: https://www.instructables.com/id/The-Drone-Pi/..

[14] "TensorFlow Object Detection Model Zoo," Google, 2019. [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

[15] "TensorFlow-Object-Detection-on-the-Raspberry-Pi," 25/2/2019. [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi..

[16] ""TensorFlow-Object-Detection-on-the-Raspberry-Pi," 25/2/2019. [Online]. Available: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi.".