Real-Time Object Classification on Aircraft Board using Raspberry Pi 3

Nikita V. Belov[1], Boris Y. Buyanov[2], Vera A. Verba[3]
Department of Intelligent Control Systems and Automation
Moscow Technical University of Communications and Informatics
Moscow, Russia
[1]djnikone2777@yandex.ru, [2]b.buyanov@gmail.com, [3]verba@list.ru

***Abstract*. The approach to solving the problem of classification of objects in real time and its application on an unmanned aerial vehicle are investigated. When implementing this system, a Raspberry pi 3 microcomputer with a video camera connected to it, as well as an artificial neural network of the Fast Yolo architecture is used.**

***Key words:* unmanned aerial vehicle; hexacopter, artificial neural network; Fast YOLO v2; Raspberry pi 3.**

I. INTRODUCTION

This article uses INS architecture Fast YOLO V2 (You Only Look Once), which is a faster development of the YOLO v2 architecture, and allows you to detect objects on the video stream on embedded devices in real time. Firstly, it is used the evolutionary structure of deep intelligence for the of YOLO v2 network architecture and create an optimized architecture at that has 2.8 times less parameters. To further reduce power consumption on embedded devices while maintaining performance, the adaptive movement method is introduced into the proposed fast YOLO fast structure to reduce the frequency of the outputs using Fast YOLO v2 based on the time characteristics of the motion. Fast YOLO architecture reduces the number of pins by an average of 38.13% to detect objects in the video stream compared to the original YOLO v2.

II. DETECTION OF OBJECTS AND THEIR LOCALIZATION

In the industrial Internet of things, a hexacopter with an installed object recognition system can be used for visual inspection of ships and determining breakages, tracking the spread of weeds, analyzing the number of people at enterprises and so on. By transmitting a video signal and communicating with any device with a wi-fi network on board, it can act as a sensor and help in the automation of production.

Detection of objects [1; 5] is one of the most difficult tasks in the field of technical computer vision. The purpose of object detection is to localize various objects in the scene and assign labels to object restrictions. The most common approach [1; 2] to solving this problem is to reuse existing trained classifiers to assign labels to the bounding rectangles in the scene. For example, you can use the standard slip approach [1], where the classifier defines the existence of an object and the associated labels for all possible windows in the scene. However, this approach has significant limitations in terms of not only high computational complexity, but also a high rate of detection errors.

Deep neural networks (DNN) have shown excellent performance in various applications [3; 4], and object discovery is one of the key areas where DNN has far exceeded all previously known approaches. In particular, convolutional neural networks (CNN) have demonstrated the best characteristics when solving the object detection problem. For example, the CNN approach uses an architecture to create bounding rectangle suggestions in an image instead of a moving window approach, and thus, the classifier divides into classes by bounding rectangles. Although the modernized architecture of R-CNN (Region-based Convolutional Network) is capable to obtain maximum accuracy, the entire procedure is slow and cannot be optimized, since each component must be trained individually.

In 2016, the “Detecting an object in one pass (YOLO)” [6] approach was proposed, which mitigated the complexity of computations associated with R-CNN, posing the problem of object detection as a problem with a single regression, where the coordinates of the bounding framework and the probability of classes are computed parallel. Although it was demonstrated that YOLO has significant speed advantages over R-CNN (for example, 45 frames per second on the Nvidia Titan-X graphics processor), it has also been shown that the localization error of YOLO is significantly higher than in architectures such as Faster R-CNN [7].

J. Redmon and A. Farhadi [8] proposed an improved YOLO method (called YOLO v2), where the so-called “anchor boxes” are used to predict bounding rectangles. In addition, compared to YOLO, YOLO v2 does not have fully connected layers in its network architecture. To make the network faster, YOLO v2 uses the new architecture of CNN (Darknet-19), unlike other frameworks that use VGG-16. Darknet-19 requires 8.52 billion floating-point operations, which is significantly lower than VGG-16, for which requires 30.69 billion floating-point operations on each pass. Experimental results for this approach have shown that YOLO v2 can perform object detection with 67 FPS on the Nvidia Titan-X graphics processor, achieving the fastest detection characteristics.

III. PRINCIPLE OF WORK YOLO

The principle of YOLO is the following: at each step you run the classifier to get a prediction of what sort of object is inside the current window. Using a sliding window gives several hundred or thousand predictions for that image, but we will keep only the ones, in which the probability of finding an object is more than 30%.

This approach shows good results, but obviously, it will be very slow, since it requires multiple launch of the classifier. A more effective approach is to predict which parts of the image contain interesting information (the so-called regional proposals), and then we run the classifier only in these areas.

YOLO V2 uses a different approach. This is not a traditional classifier, which is an object detector, a YOLO V2 divides the image into cells 13x13 pixels, as shown in Fig. 1.

Fig. 1. Separation of the image into separate cells [10]

Each of these cells is responsible for the prediction of 5 bounding rectangles. The bounding box describes the rectangle that surrounds the object.

YOLO also calculates a confidence score that shows how reliably it is determined that the predicted bounding box actually covers an object.

The predicted bounding rectangles look as shown in Fig. 2 (the higher the confidence indicator, the greater the thickness of the lines that limit the field).

Fig. 2. Example of forecasting [10]

For each bounding block of cells, the class also predicts. This works just like a classifier: it gives the probability distribution over all possible classes. In the version of YOLO, training was conducted on a PASCAL VOC data set [9], which can detect 20 different classes, such as bicycles, boats, cars, cats, dogs, people, etc.

The confidence estimate for the bounding framework and class prediction is combined into one final score, which indicates the probability that this bounding box contains a certain type of object. For example, a big fat yellow line on the left indicates a dog with a probability of 85% (Fig. 3).

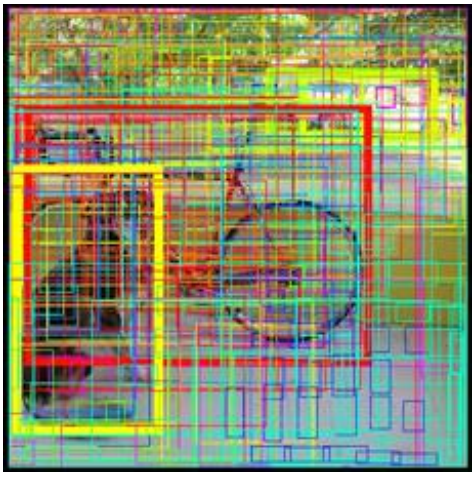

Fig. 3. Selecting objects in the picture [10]

Since there are 13 × 13 = 169 grid cells, and each cell predicts 5 bounding rectangles, it eventually results in 845 bounding rectangles. Most of these cells will have very low confidence ratings, so only those rectangles are saved, for which the final result is at least 30%.

In our case, the final prediction is shown in Fig. 4.

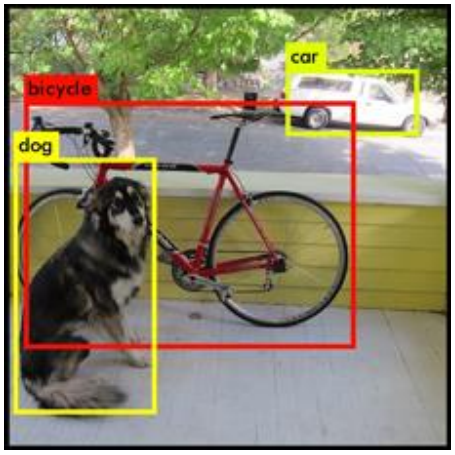

Fig. 4. Final prediction [10]

Of the 845 common restrictive boxes, only these three are preserved, because they give the best results. But even if there were 845 individual predictions, they were all done simultaneously - the neural network worked in one pass.

IV. HEXACOPTER CONSTRUCTION

To test Fast YOLO V2, a UAV was chosen onboard, a hexacopter is capable to obtain high-resolition aerial photographs and stabilized high-definition video.

The following components were used to implement the hexacopter:

- DJI F550 ARF kit;
- 3300mAh 3S 35C LiPo battery with T-connectors;

- DJI DT7 and DR16 transmitter and receiver;
- Eken H9 camera with microSD card;
- FeiYu Tech Mini 3D 3-axis gimbal;
- D-Link DWA-160;
- Raspberry pi 3.

Before the beginning of the project it is necessary to determine the purpose of each part of the hexacopter. Below is the specification of hexacopter elements:

- Frame. The frame provides a convenient way to fasten engines and electronics. DJI Flame Wheel 550 (F550) is used in this project:.
- Brushless motors and propellers. The engines rotate the propellers to create traction for lifting the hexakopter. Engines DJI E305 960KV and propellers DJI 9450 are in this project.
- Electric speed controllers (ESC). ESCs nourish brushless motors and provide a PWM interface that allows the flight controller to monitor the speed and thrust of each engine. DJI controllers E305 960KV are used in this project.
- Flight Controller. The flight controller controls how the hexacopter flies. The controller is a small computer with an inertial measuring block or IMU (which includes a gyroscope and an accelerometer) to support and stabilize a hexacopter: the flight controller also has a barometer, GPS, a magnetometer that allows the flight controller to know at what altitude it is, where he is on land and in what direction it flies, respectively. NAZA-M Lite with GPS kit is used in this project.
- Voltage regulator, flight controller status LED / USB interface. The voltage regulator provides a constant 5V for the flight controller. The status LED transmits information about the state of the hexacopter to the pilot during the flight. The indicators include a battery and a GPS. The USB interface also allows you to configure the flight controller from the PC. Used voltage regulator and status LED are included in the delivery NAZA-M Lite.
- Battery. Lithium-polymer (LiPo) batteries are the preferred method for providing the energy of a hexacopter due to the high energy-to-weight ratio and high maximum discharge currents.
- Transmitter + Receiver. The transmitter receives the pilot commands with a few joysticks and switches and transmits them over the wireless network to the receiver on a hexakopter. The receiver decodes the pilot's commands and sends them to the flight controller. DJI DT7 transmitter with DR16 receiver is used in this project.
- Gimbal + Camera. The camera and the gimbal turn the hexacopter into an aerial photography platform, capable of capturing professional quality frames. The camera is attached to the gimbal, which, in effect, performs opposite movements of the hexacopter, making sure that the camera is always horizontal. Eken H9, FeiYu Tech Mini-3D Gimbal are used in this project.
- Raspberry pi 3. The camera is connected to Raspberry Pi 3 with a 5GHz WiFi transmitter. This allows Raspberry Pi to broadcast live stream from the camera to any device connected via WiFi. The Raspberry Pi 3 has a Ubuntu Mate system, which has a wide range of applications. Model Raspberry Pi 3, D-Link DWA-160 are used in this project.

Video from Raspberry Pi 3 is sent via the Wi-Fi access point to the receiving device. There are probably three ways that you can place a Wi-Fi access point:

- The receiving device and the aircraft are connected to the same network. This approach allows you to use a powerful WiFi-router and increase the transmission range of the signal. The disadvantage is that the router is another object that must be placed on the UAV. The router also requires a separate power source.
- A Wi-Fi hotspot located on the Raspberry Pi 3, to which the receiving device is connected. This solution will work with any device, regardless of whether it can host a Wi-Fi hotspot or not.
- The Wi-Fi hotspot is located on the receiving device. Hosting the Wi-Fi access point on the device eliminates the need to move the Wi-Fi router and additional power source and makes it easy to place a WiFi access point.

In this system, we need to add signal protection functions, such as a password for accessing the wi-fi network wherein the devices are located, and also protecting the radio signal to prevent interception of device control. For this, the transmission frequency changes every 2 ms (i.e., 500 times per second), the transmitted data is mixed with a pseudo-random sequence, for more efficient transmission, and for noise immunity.

Figure 5 shows the final assembled aircraft.

Fig. 5. Collected UAVs

Fast YOLO V2 provides high performance enough to solve the task in real time on a powerful graphics processor, but it is still very difficult to use this approach to detect objects in real time in a video stream on embedded computing devices with limited processing power and limited memory. For example, in various real-world applications, such as real-time output to smart phones or embedded video surveillance, the available computing resources are limited to a combination of embedded graphics processors with low power. Therefore, the detection of objects in real time in the video stream on embedded devices remains a big problem.

After the UAV assembly, Fast Yolo v2 is configured on the Raspberry Pi 3 microcomputer.

V. IMPLEMENTING YOLO ON RASPBERRY PI 3

The Yolo models used on the Nvidia Titan-X GPU:

- yolo.cfg is based on the extraction of the network. It processes images at a rate of 45 frames per second;
- yolo-small.cfg has smaller fully-connected layers, so it uses much less memory. It processes images at a rate of 50 frames per second.

As you can see, Fast Yolo v2 produces more frames per second on the Nvidia Titan-X graphics processor, compared to Yolo v2, which is why it was decided to use it on Raspberry pi 3.

The original YOLO model uses a lot of GPU memory but because is used Raspberry Pi 3, you need to use a smaller version of the YOLO model yolo-small.cfg.

The small version of YOLO uses only 1.1 GB of GPU memory, so it is suitable for Raspberry pi 3.

For implementation, a pre-trained model was used. Coefficients of the network are available on the official website of YOLO [10]. The next task was to import the DarkNet weights in TensorFlow, for this you need to install DarkNet. After you need to run Fast Yolo v2 in test mode and because DarkNet is used on Raspberry Pi 3, 6-12 images should be processed per second.

We present the classification of objects in real time with Raspberry pi 3.

Running Fast YOLO v2 on the test data is only needed for testing the system, in a real system, the video stream from the video camera is used.

An example of how Fast YOLO v2 works on a video stream from a video camera is shown in Fig. 6.

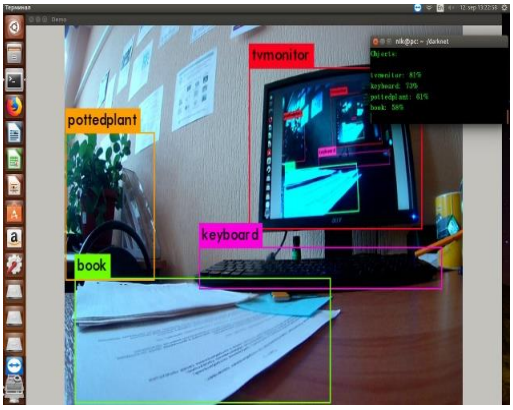

Fig. 6. Example of the work of Yolo

Fast YOLO v2 displays the current FPS and predicted classes, as well as the actual image with the bounding rectangles drawn on top of it.

As you can see in Fig. 7, the FPS (frames per second) is 9.8, which is not enough to use Fast Yolo V2 on Raspberry pi 3 in real time, because to use the FPS real-time system, it must be at least 24.

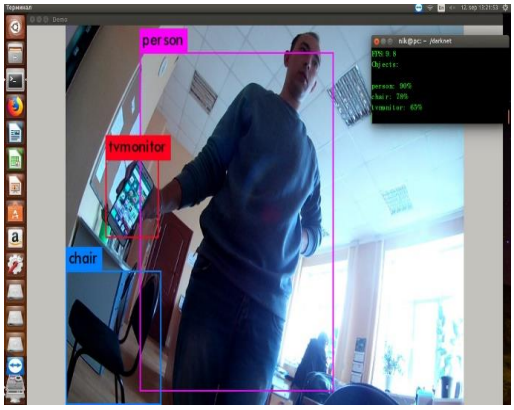

Fig. 7. Example of Yolo algorithm work

VI. CONCLUSIONS

The article discusses the INS architecture of Fast YOLO V2, this is the new INS deep learning architecture for solving real-time object detection problems on a video stream. Although YOLO v2 is considered the most modern architecture and allows you to solve the task in real time on powerful graphics processors, its direct use on embedded devices in real time is not yet possible.

In the work, the hexacopter was assembled and tuned, components were selected and a real-time video stream was transmitted, which was preprocessed on board the hexacopter with the Raspberry Pi 3 microcomputer with the preinstalled Ubuntu Mate 16.04.1 operating system.

Fast Yolo v2 on Raspberry pi 3 allows you to speed up the process of recognizing objects in real time by 33%, relative to using Yolo v2. Fast YOLO v2 can achieve an average run time of 3.3 times faster than the original YOLO v2, but this is not enough to use it in real time on Raspberry pi 3.

REFERENCES

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

[2] Lienhart R., Maydt J. An extended set of haar-like features for rapid object detection // Proceedings. International Conference on Image Processing. Rochester, 2002.

[3] Krizhevsky A., Sutskever I., Hinton G. Imagenet classification with deep convolutional neural networks // NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems. 2012. Vol. 1. P. 1097–1105.

[4] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition // Proc. Int. Conf. Learn. Represent. 2015. https://arxiv.org/abs/1409.1556.

[5] Girshick R., Donahue J., Darrell T. et al. Rich feature hierarchies for accurate object detection and semantic segmentation // Computer Vision and Pattern Recognition. 2014.

[6] Redmon J., Divvala S., Girshick R. et al. You only look once: Unified, real-time object detection // Computer Vision and Pattern Recognition. 2016.

[7] Ren S., He K., Girshick R. et al. Faster RCNN: Towards real-time object detection with region proposal networks // Advances in neural information processing systems. 2015.

[8] Redmon J., Farhadi A. YOLO9000: better, faster, stronger // Computer Vision and Pattern Recognition. 2017.

[9] Pascal. URL: http://host.robots.ox.ac.uk/pascal/VOC/.

[10] Yolo: Real-Time Object Detection. URL: https://pjreddie.com.

[11] Буянов Б.Я., Верба В.А. Использование модуля dnn библиотеки OpenCV 3.3 для распознавания объектов // Инновационные технологии в кинематографе и образовании: Сборник IV Междунар. науч.-практ. конф. / ВГИК. М., 2017. С. 47–56. [Buyanov B.Ya., Verba V.A. Using the dnn module of the OpenCV 3.3 library for object recognition // Innovative technologies in cinema

and education: Collection IV Intern. scientific-practical conf. / VGIK. Moscow, 2017. P. 47–56.]

[12] Буянов Б.Я., Верба В.А. Многомерный статистический и когнитивный анализ крупномасштабных систем // Вызовы глобального мира, Вестник ИМТП. 2014. № 4. С. 23–26. [Buyanov B.Ya., Verba V.A. Multivariate Statistical and Cognitive Analysis of Large-Scale Systems // Challenges of the Global World, IMTP Bulletin. 2014. N 4. P. 23–26].

[13] Буянов Б.Я., Верба В.А. Некоторые вопросы определения пространства состояний параметров сложных систем // Системный анализ в проектировании и управлении: Сб. науч. тр. по итогам XXII Междунар. науч.-практ. конф. СПб.: СПбГТУ, 2018. С. 224–229. [Buyanov B.Ya., Verba V.A. Some issues of complex systems state space parameters determining // System analysis in design and management: Sat. scientific tr. on the results of XXII International. scientific-practical conf. SPb .: SPbSTU, 2018. P. 224–229].

[14] Буянов Б.Я., Верба В.А. Мультиагентные модели сложных социо-технических систем // Системный анализ в проектировании и управлении: Сб. науч. тр. по итогам XXII Междунар. науч.-практ. конф. СПб.: СПбГТУ, 2016. С. 155–159. [Buyanov B.Ya., Verba V.A. Multi-agent models of complex socio-technical systems // System analysis in design and management: Sat. scientific tr. on the results of XXII International. scientific-practical conf. SPb.: SPbSTU, 2016. P. 155–159].