www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

Vol. 03, Issue 01, January 2023, pp : 10-17

e-ISSN : 2583-1062

Impact Factor : 2.265

DEVELOPMENT OF AN UNMANNED AERIAL VEHICLE (QUADCOPTER) WITH REAL TIME OBJECT MONITORING

Mohammed Yakub Adinoyi[1], Salifu Simeon Imaben [2], Alaiyemola Dave Oluwafemi[3], Edoka Romanus[4], Jimoh Ajibola Johnson[5], Osanuwa Rotimi Lawrence[6]

[1]Electrical Electronics Technology Department, Kogi State College of Education (Technical), Kabba, Kogi State, Nigeria.

[2]Physics Department, Kogi State College of Education (Technical), Kabba, Kogi State, Nigeria.

[3]Computer Science Department, Kogi State College of Education (Technical), Kabba, Kogi State, Nigeria.

[4] Electrical Electronics Technology Department, Kogi State College of Education (Technical), Kabba, Kogi State, Nigeria.

[5]Computer Science Department, Kogi State College of Education (Technical), Kabba, Kogi State, Nigeria.

[6]Computer Science Department, Kogi State College of Education (Technical), Kabba, Kogi State, Nigeria.

ABSTRACT

Drones have been frequently used for aerial photography in recent years at significantly cheaper rates. However, the most modern drones are exceedingly error-prone and require precise manual control to take high-quality photos or films. We suggest using quadcopters equipped with object monitoring to power drones equipped with cameras, enabling the drone to perform automatic tracking and detection. We used cutting-edge detection and tracking algorithms to implement the visual part of this project. Our system was put into use on a variety of hardware platforms, including both desktop and GPU (NVIDIA GTX980) and embedded GPU (NVIDIA Tegra K1 and NVIDIA Tegra X1) and evaluated frame rate, power consumption and accuracy on several videos captured by the drone. Our system achieved real time performance at 71 frames per second (fps) for tracking and 1.6 fps for detection on NVIDIA TX1.

Keywords: Quadcopter, Object Monitoring, Drone

1. INTRODUCTION

Modern drones come with cameras and have a lot of potential for application in a wide range of businesses, including surveillance, aerial photography, and other fields. Drones must have sophisticated computer vision and autopilot in order to be widely deployed and to further lower their costs. Object recognition and tracking are crucial in the use of aerial photography in order to capture important things in a scene. The classic issues in computer vision are object recognition and tracking. Drones provide extra difficulties because of their top-down view angles and time restraints, though. Another difficult issue is the integrated technology's strict weight and space restrictions, which prevent drones from running computationally complex algorithms like deep learning with sufficient hardware resources.

Deep Drone is a framework that intends to tackle both problems while running on embedded systems that can be mounted onto drones. For this project, we present our vision system pipeline and its performance along with accuracy on multiple hardware platforms, and we analyzed the trade-off between accuracy and frame rate.

Related work

Deep neural networks, object detection and object tracking are the three major components in our work. We first present an overview of past work, and then describe our improvements.

Deep neural network is the state-of-the-art technique in computer vision tasks, including image classification, detection, and segmentation. AlexNet [1] is the classic network proposed in 2012 that has 8 layers and 60 million connections, and won the Imagenet contest in 2012, spawning a lot of improvements later. After that, VGGNet [2] was proposed, which has 16 layers and 130 million parameters. Both AlexNet and VGGNet have bulky fully connected layers which results in huge model sizes. GoogleNet t [2] is a more compact CNN that consists mostly of layers, it uses the inception model that has 1x1, 3x3 and 5x5 convolution kernels with different scale. It also has multiple loss layers to prevent vanishing gradient problem. ResNet [3] was proposed in 2015 and greatly improved the image recognition accuracy, it adds bypass layers to let the network learn the residual rather than the absolute value. SqueezeNet [4] was proposed recently that is aggressively optimized for model size. It has 50x fewer connections and half the computation than AlexNet but has higher accuracy. After Deep Compression [5, 4], the model is only 470KB and fits well into the last level cache. Fast R-CNN [3] is a Fast Region-based Convolution Network method (Fast R-CNN) for object detection. In this algorithm, an input image and multiple regions of interest (RoIs) are input into a fully convolution network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by

fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

Faster R-CNN [6] made further improvements on Fast R-CNN that introduce a region proposal Network (RPN) that shares full-image convolution features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully-convolution network that simultaneously predicts object bounding boxes and scores at each position. RPNs are trained end-to-end to generate highquality region proposals. Because of the region proposal network are fused and could be trained end to end, the network is faster than fast R-CNN.

YoLo Detector [7] is a new approach to do object detection. Prior work on object detection re-purposes classifiers to perform detection. Instead, YoLo use object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. The unified architecture is extremely fast but not as accurate as Faster R-CNN.

KCF[7, 8] is the kernelized correlation filters used for detection. It uses the characteristic that under some conditions, the resulting data and kernel matrices become circulant. Their diagonalization by the DFT provides a general blueprint for creating fast algorithms that deal with translations, reducing both storage and computation by several orders of magnitude, obtaining state-of-the-art trackers that run at 70 frames per second on NVIDIA TK1 and is very simple to implement.

Fast and Faster R-CNN originally used VGGNet for feature extraction. It is accurate but slow. Drones have limited hardware resource both in memory and in computation power, so we need to have smaller network. In order to run the CNN fast enough on embedded device, we didn't use those off-the-shelf network architectures. Instead, we used a relatively shallow and small network to extract image features and to detection. The architecture is shown in Fig 1.

Even using our small network architecture, the detection frame rate is still low on TK1 mobile GPU. To compensate for the slow speed of detection, we used the cheap KCF tracker, although it's less accurate than MDNet, to track on the bonding box returned by the detection pipeline. Thus we have the accurate but slow Faster R-CNN for detection, and have the less accurate but super fast KCF for tracking. Detection is only called when the confidence of the tracker is below certain threshold, which is very infrequent. This architecture makes the pipeline accurate, robust and fast.

Accuracy is not our sole target in this project. We have a thorough evaluation with respect to accuracy, power consumption, speed, and area of different hardware running detection and tracking algorithm. Balancing these hardware constraints, rather than only optimizing for MAP, is our top priority.

2. METHODOLOGY.

2.1 System Architecture

Our vision system's software architecture is made up of two parts. The first portion of the algorithm is a Convolutional Neural Network (CNN)-based detection technique, while the second part is a tracking algorithm that makes use of the HOG feature and KCF. To ensure slick and real-time performance, these two algorithms are smoothly interwoven. Given that CNN-based detection uses several GOPs each frame and is only invoked to initialize a bounding box for important objects in the scene, detection algorithms, such as Faster RCNN, are expensive to compute. The tracking method, like KCF, may run at a high frame rate and is very cheap to compute in order to follow the bounding box that the detection technique provides. The following pseudo code displays the core algorithm loop, and we talk about the detection

2.2 System Architecture

Two components make up the software architecture of our vision system. Convolutional neural networks (CNN) are used in the first component, a detection technique, and KCF and the HOG feature are used in the second component, a tracking algorithm. The performance of these two algorithms is smooth and real-time because to their seamless integration. The CNN-based detection contains several GOPs per frame and is only invoked to initialize a bounding box for a key object in the scene, making the computation of the detection algorithm, such as Faster RCNN, expensive. To follow the bounding box that the detection method provides, the tracking algorithm, like KCF, is relatively cheap to compute and can operate at a high frame rate. The core algorithm loop is displayed in the following pseudo code, and we go over the detection

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

Vol. 03, Issue 01, January 2023, pp : 10-17

e-ISSN : 2583-1062

Impact Factor : 2.265

Algorithm 1 Detection and Tracking Pipeline for Deep Drone

```
boxFound ← false
while true do
    f ← new frame
    while boxFound == false do
        detection(f)          ▷ Invoke detection algorithm
        if Box is detected then
            boxFound ← true
        end if
    end while
    tracking(f)               ▷ Invoke tracking algorithm
    if Tracking is lost then
        boxFound ← false
    end if
end while
```

2.3 Detection

The initial area of study was identifying a human figure because drones are mostly used for surveillance by collecting photographs of terrain and people. The person with the highest detection score is our target because it was presummated that there would be just one person of interest to track. Yolo detector and Faster RCNN detector have both been used to detect the human figure.

2.3.1 Faster R-CNN

For person detection, a 7-layer convolutional neural network on Faster R-CNN [15] was used. The framework outputs bonding boxes and target classes for detected objects after receiving raw image frames from a video feed. On the KITTI[2] dataset, an internal model was employed. KITTI was chosen because it has a wealth of training examples, including vehicles, people, and bicycles, and can be readily streamlined to accomplish our goal. We changed the KITTI script to only detect people because the human figure is an interesting detection target. In Figure 1, a thorough architecture is displayed.

We calculated the speed (mAP) and accuracy (mAP) of our own model and contrasted them with the norm, as shown in table 1. Although it runs 12 times faster than the baseline model, our own model is marginally less accurate.

Table 1 shows the precision and efficiency of our detection network (runtime is measured on GTX980 GPU

Model	mAP	Runtime
Baseline[15]	65.9 %	2s
Ours	62.0 %	0.17s

2.3.2 Yolo Detector

Yolo detector[14], which is simpler to compile for mobile, was also attempted by us. Although less accurate, it is also a faster option. Yolo detector was not used because, in our experiments, we discovered that it was unable to identify situations in which the human figure was small and distant.

2.4 Tracking

Despite other state-of-the-art tracking algorithms being more accurate and performing better on the VOT 2015 challenge, the KCF method was picked above them [10]. This was due to the demand for a real-time tracking system for the drone that could provide reliable control commands to the drone.

2.5 Hardware Platform

Drones are specialized hardware platforms with constrained weight and area, making compute power difficult. Even if it's faster, we can't afford to employ desktop GPUs for the detection process. We assessed the computation time for detection and tracking in Figure 1 and the power consumption of these hardware platforms is shown in the same table for comparison purposes. The GTX980 is about ten times as fast as the TK1, but it uses twenty times as much power. TX1 uses about the same amount of power but is three times faster than TK1. In conclusion, the TX1 hardware platform would be the best choice for drone detection.The TX1 is ideal with respect to speed and power consumption, the TX1 development board is much larger and much heavier than the TK1 board, making it very hard to put on the drone. However, TK1 is smaller and DJI provides the Manifold box to hold the TK1, which makes it very easy to use.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

Vol. 03, Issue 01, January 2023, pp : 10-17

e-ISSN : 2583-1062

Impact Factor : 2.265

3. MODELING AND ANALYSIS

The block diagram used in implementing this project is shown in Figure 1 below:.

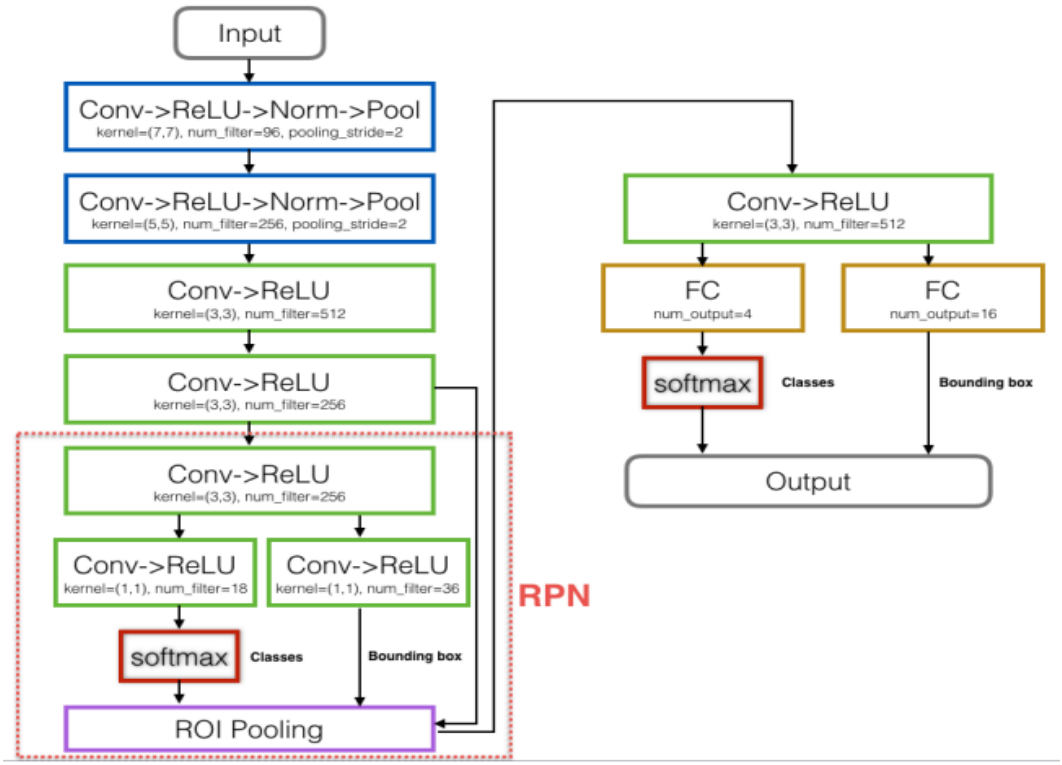

Figure 1: The CNN architecture used for detection

We purchased a small carrier board for TX1 in order to accommodate the development board's huge form factor. In Figure 2, it is displayed. We may unhook the TX1 central board from the whole development board and only connect the central board that includes the actual TX1 chip to this carrier board because it is as little as the heat sink in size. As a result, it is even smaller than TK1. There isn't a free lunch, though. We haven't had a chance to link the carrier board with the drone yet; this could be future work. The interface, particularly the power supply, is incompatible with the DJI drone. The TX1 carrier's TK1 is completely functional, if not speed-optimized.

Figure 2. NVIDIA TX1 and TK1 development board we used

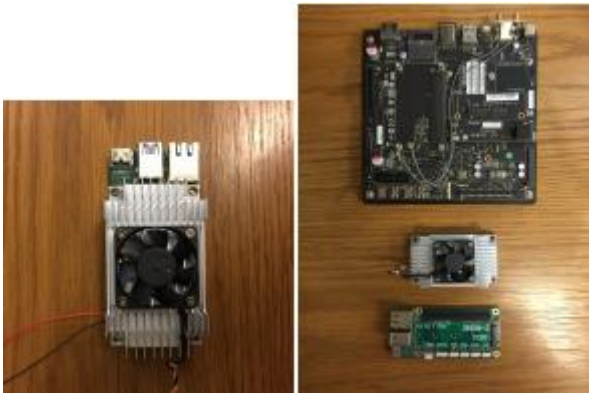

Figure 3. In order to reduce the size of TX1, we bought a small carrier board.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

Vol. 03, Issue 01, January 2023, pp : 10-17

e-ISSN : 2583-1062

Impact Factor : 2.265

4. RESULTS AND DISCUSSION

4.1 Detection

Faster RCNN has a number of tailored region pooling layers and is built on top of Caffe, a deep learning framework requiring numerous dependencies. On all of our desktop and embedded systems, CUDA and Faster RCNN were painstakingly installed. The following difficulties were encountered during installation.

1. We loaded the CUDA 6.5 development kit on our TK1 after flashing it with the most recent version of L4T (Linux for Tegra). However, the TK1 does not support CUDA versions higher than CUDA 6.5. However, if we switch to an earlier branch of Caffe, it won't support the new layers needed by faster r-cnn because the most recent version of Caffe is not backward compatible with CuDNN v3 and earlier. To get around this issue, we deactivated the CuDNN switch during Caffe installation.
2. Python was used to build some of the Faster RCNN libraries into native C++ code. We encountered a compilation problem on gpu nms.cpp, a GPU implementation for non maximum suppression, when installing these libraries onto embedded devices, namely TK1 and TX1. Compiler compatibility issues in embedded systems were determined to be the primary factor.

Eventually, we discovered a workaround by manually altering the produced cpp file's source code, and the compilation was successful. In order to assess testing performance using offline and internet video streams, we lastly trained a 7-layer CNN Caffe model using GTX 980 and translate the model across all platforms.

4.2 Tracking

The original KCF technique is implemented in Matlab for tracking. But because of TK1's hardware constraints (storage capacity and processing power), we use C++ to implement the method. In line with what the original model suggested, the tracker interface offers an initialization method that accepts a frame and a starting bounding box, then builds a model using linear regression from sample patches of the frame with the aid of cyclic shift. When a new frame is received, the tracker's update function is called. It first detects patches of the same size; if the resulting detection score falls short of a certain level, further patches under various scales are recognized. The rescaling factors we utilized were 110% and 90%. Last but not least, we return the result with the highest score (the new positive and negative patches are used here for online learning).

a. Handshake between detection and tracking

We use Python.h to construct a C++/Python binding between the two algorithms as the detection algorithm is built in Python. The C++ main program stores the initialization of the detection algorithm (loading the neural network under Caffe) as a PyObject. The detection function is called with the resultant array after the C++ main program converts the frame (which is stored in a byte array) into a Python array object. After parsing the outcome, it checks to see if the detection result matches a threshold (how confident the object is a person). When importing a Python Module under sudo (which is required to activate and use the DJI drone's camera), an intriguing problem occurs since part of the Python. The NVIDIA TX1 and TK1 development board that we used packages are installed with root permission only, we worked around this problem by command, and hacked the privilege of using the DJI live camera.

b. Offline Detection and Tracking

With the TK1 board put on the drone, we initially tested our detection and tracking module on our own offline video recordings from various angles (a DJI Inspire clip of Mr. Emeka Justus playing in the sand). When utilizing faster rcnn for detection, each frame takes an average of 1.6 seconds (as opposed to 0.6s on TX1, which is tiny enough to mount on). DJI does not currently support the drone). Every frame just needs to be 14 ms for tracking under KCF (71 fps). Running the two films through our detection and tracking module reveals a number of intriguing issues for us. First, the performance of the tracking method will depend on how tight the detection and bounding box results are. If the bounding box is too loose, it may include unrelated subject (in the example above, a significant amount of shadow, for example).

Figure 4 Sub-Image 4), tracking will erroneously believe that it is the unimportant subject that it wishes to track. Second, when the person is posing as a larger figure, detection is less effective. As in the snowboarding video, it is difficult to identify the individual while looking at them from the side because of their helmet, face mask, and heavy clothing. By changing the detection confidence threshold and retraining the neural network with photos taken from various angles and viewpoints, we are able to solve the two issues.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

Vol. 03, Issue 01, January 2023, pp : 10-17

e-ISSN : 2583-1062

Impact Factor : 2.265

Figure 4. Faster From a drone's perspective, RCNN does a great job of people detection. even when the item is distorted and far away (last figure).

Figure 5. Yolo detector's performance falls short of Faster RCNN's. The detector fails if the target is a little person.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

Vol. 03, Issue 01, January 2023, pp : 10-17

e-ISSN : 2583-1062

Impact Factor : 2.265

Figure 6. Yolo detector performs not as well as Faster RCNN. When the target person is small, the detector fails

c. Online detection and tracking on DJI live cam

We then modify the detection and tracking module to work with the live camera on the DJI M100. According to the prior section, the camera reads data as a raw byte array in the NV12 format, which includes three components for each pixel: a part of the luma (the brightness) Add the two color components U and V, as the detection and tracking Each and every algorithm is based on RGB pixel values. In order to create an RGB representation of the frame, we convert the data as follows (clamping means restrict the value within the 0-255 range for RGB)

$$R = clamp(Y + 1.4075(V - 128))$$

$$G = clamp(Y - 0.3455(U - 128) - 0.7169(V - 128))$$

$$B = (int)(Y + 1.779(U - 128));$$

After getting the RGB-encoded frame, we follow a similar procedure to that used for offline video and detect and track the video frame by frame. This step presents a challenge because we are performing detection on a live stream, which means that the object may be moving quickly while we are detecting. As a result, it's possible that the object has already left the bounding box by the time detection has finished examining a frame from t seconds ago. To solve this problem, we initialized the tracker with the frame from t seconds prior as opposed to the current frame. The tracker will subsequently train its positive and negative patches with the proper boundary region. This resolved the issue, barring excessive deformation of the item Δt time frame.

d. Controlling the Drone

Using the OnBoard SDK that DJI has made available, we are controlling the camera. We must first activate the CoreAPI driver by sending it some data. This phase is really problematic because DJI just updated their drone operating system while leaving its Onboard SDK unupdated. To properly activate the updated version of the encryption key, we had to get in touch with the DJI developer who created their Onboard SDK. By providing Gimba AngleData to the camera, we can then take control of it. A gimbal angle data set includes three import fields called yaw, roll, and pitch that correspond to the three degrees of freedom for the camera.

3. CONCLUSION

We introduce Deep Drone, a real-time detection and tracking system that runs on integrated technology and provides the drones with vision. We demonstrated our software design, which combines the more accurate but slower detection algorithm with the quicker but less accurate tracking algorithm to create a system that is both quick and accurate. Additionally, we evaluated the size, runtime, and power consumption of various hardware platforms, and we talked about embedded hardware implementation problems and solutions.

4. REFERENCES

[1] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In Proceedings of the IEEE International Conference on Computer Vision, pages 4310–4318, 2015.

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR), 2013.

[3] R. Girshick. Fast r-cnn. In International Conference on Computer Vision (ICCV), 2015.

[4] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[5] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems, pages 1135–1143, 2015

[6] G. Zhu, F. Porikli, and H. Li. Tracking randomly moving objects on edge box proposals. arXiv preprint arXiv:1507.08085, 2015.

[7] Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015.

[9] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernandez, T. Vojir, G. Hager, G. Nebehay, and R. Pflugfelder. The visual object tracking vot2015 challenge results. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 1–23, 2015.

[10] Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. arXiv preprint arXiv:1510.07945, 2015