Design And Implementation of a Mini Defense Drone

Ahmet Yasir Cilvez, Muhammed Samed Özmen
Electrical-Electronics Engineering, AGU

Prof. Dr. İrfan Alan
Electrical-Electronics Engineering, AGU

***Abstract*— It is aimed to design and develop a ‘’Mini Defense Drone’’ in order to find a solution to one of the military defenses needs. In this project, the Mini Defense Drone detects spy or armed drones to determine their location by using pixels on the location of the screen and tracking. To achieve these goals, a small and light drone was selected. To detect the spy drone by using image processing with YOLO algorithms on Pycharm with getting images from a camera that is used in the mini drone and determining the location of the enemy drone as pixel coordinates. After, determining pixel coordinates, drone tracks according to these coordinates with using PID algorithm and catching spy drones. Also, it is aimed to land the drone without being damaged by creating an emergency landing algorithm for problems that may occur during tracking.**

Keywords*— *Drone, Defense, UAV, YOLO, Spy, Detect, Tracking

I. INTRODUCTION

The areas of use of unmanned aerial vehicles (UAVs) are growing rapidly with the development of technology and the needs of the world. Considering this growth, many different studies will emerge in this field in the future. In this process, previously manually controlled UAVs are now used autonomously. Manually operated UAVs lose their value along with autonomously used UAVs. In addition, autonomous drones, with their artificial intelligence, have an advantage by reacting better than humans. There are several types of UAVs and one of them is the rotary wind drone. Such drones are used for tasks that require flying robots, are difficult to complete, etc. It is used almost everywhere. A major use case is the defense industry that needs espionage, surveillance, elimination, deception, or other military missions. Drones help the military complete such challenging missions. Various technologies are used for autonomous flying, object detection, detection, and data collection in drones. In this project, a mini defense drone was developed and implemented to detect and track other drones that could be enemy drones.

II. METHODOLOGY

In order to drone must complete its mission, the project contains four main parts which are DJI Tello Ryze, object detection, object tracking, and autonomous flying. The reason there are no physical parts to complete, a ready-made drone, DJI Tello Ryze, was preferred in the market available due to budget limits. DJI Tello drone and its developed algorithms are explained following sections. In the development process, all codes were created in Python language in PyCharm environment.

A. DJI Tello Ryze

As mentioned before, to object training and autonomous flying, a programmable drone must be used. Because a programmable custom drone with a flying control card and companion computer was not used, DJI Tello Ryze which allow it to be programmed was preferred.

DJI Tello Ryze has a 720p, 5mp camera on it and it was used to object detecting with computer vision. Also, the drone has a distance sensor under it to autonomously takeoff and landing. The drone can be seen in Figure 1.

Fig. 1. DJI Tello Drone

B. Object Detection – Computer Vision

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

Computer vision primarily involves computers and systems deriving information from data contained within images from digital images, videos, or other visual inputs. It is an artificial intelligence (AI) field that allows it to make suggestions based on these informations obtained afterward and to act depending on the suggestions found. Computer vision can understand data from those with artificial intelligence which is like the human brain helps to observe and understand. [1]

Computer vision basically works like human vision, but instead of the brain, it has artificial intelligence. Humans have many abilities such as separating, identifying, and estimating distances by learning each object cumulatively over time from their birth. On the other hand, artificial intelligence gains these abilities from train visual inputs. After training these visual inputs, the artificial intelligence classifies the objects from the

data that is trained by evaluating in the system and saves them in its memory. [1] After train visual inputs, system getting pictures in real life with cameras. However instead of human, computer or systems which have artificial intelligence need to get more information in a time.

Computer vision needs a lot of visual data to minimize the margin of error. Because computer vision performs data analysis non-stop until it recognizes the images obtained by distinguishing the data of the objects from each data received from the camera. Two main types of technologies which are convolutional neural networks and deep learning are used to object detection [1].

Deep learning is a type of Machine learning. In this method, an automatic analytical model is created by systems. It is a kind of artificial intelligence that systems can learn and define with the help of data with minimizing human intervention [2].

CNN uses image recognition and classification method to detect the visual data received from the camera. The CNN method consists of neurons with many learnable weights and biases. Each neuron in this method receives a large number of data inputs and takes a weighted sum over these inputs [3]. At the begging, it classifies the images which are obtained from the camera according to similarities, and then this data is used to perform object recognition. These algorithms can identify objects in many different domains using the CNN method.

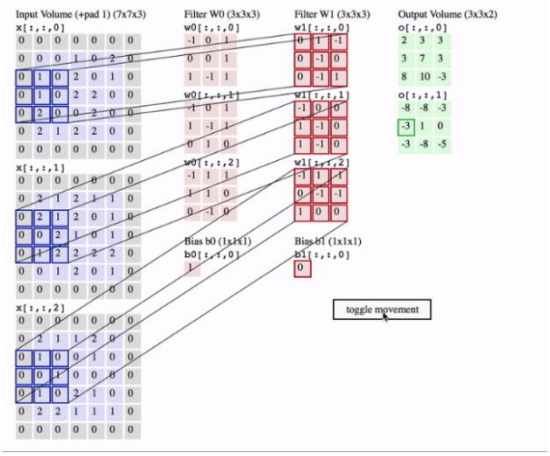

Fig. 2. RGB diagram

The images that are getting by the camera are stored in the system in the form of large pixel grids. Each resulting pixel is stored as a combination of the three primary colors red, green, and blue (RGB) [4].

The system divides the received data into multiple grids of equal size to identify a specific object. The most important reason for that is to separate objects with the same color algorithms. It performs the desired object recognition by minimizing the margin of error by non–stop processing of the data obtained from the grids [4].

Although there are 6 different tools used, OpenCV is the most used. OpenCV is an open-source tool. In this way, it reduces the workload by pulling many codes required during coding through the library. In addition, there is less margin of error in taking the shape of the center point and the rectangle surrounding the object created when defining objects on openCV compared to other applications. Also, OpenCV can be worked on all kinds of operating systems which are Windows, Linux, Android, etc. Also, OpenCV supports almost all computer languages like Java, Matlab, C++, Python, Java, MATLAB, etc. For the reasons mentioned above, openCV is mostly used today [5].

Many different algorithms are used on object detections, and there are 2 common types of algorithms. These are the Yolo and Cascade algorithms. Among these algorithms, Yolo has a fast image capture and identification feature. It can handle an average road with 45 FPS. In addition, it can go up to 155 FPS with fast-Yolo. In this way, Yolo processes the received data faster and obtains more accurate results. Also, Yolo supports OpenCV and can work with any kind of operating system. [6]

Yolo is an algorithm based on CNN. The data created in the Yolo algorithm is run over the DarkNet. There are many different Yolo algorithms available on the internet. Yolo v3 v4 and v5 are the current algorithms used today [6].

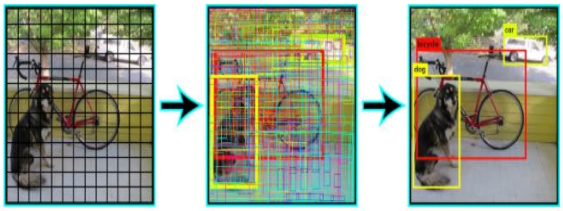

Fig. 3. N numbers grids

As seen in the Fig. 3 above Yolo algorithm divides the data from the camera into N grids of equal size and dimensional SxS matrix. It analyzes each of these N numbers of grids according to their colors and the types of weight and config created. After this analysis, it is responsible for the detection and localization of the desired and defined object [6].

The B bounding box coordinates are predicted from the cell coordinates generated from the data and clusters from the N grids obtained from the image.

However, due to these reasons, it can be defined in objects different from the desired object. In this case, the Non-Maximal Suppression method is used to minimize the margin of error. With this method, B bounding box coordinates with low similarity are eliminated. With this method, points with a low potential to match the object are removed from the system [7].

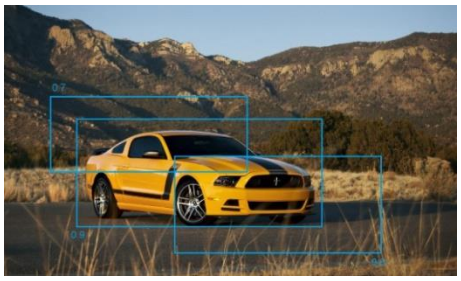

Fig. 4. Non-maximal suppression

As seen in Fig. 4, there are 3 different rectangles with similarity ratios. Similarity ratios are written in all of these rectangles. These similarity ratios are 0.7, 09, and 0.6, respectively. This result was achieved by the Non-Maximal Suppression method. However, this method alone is not sufficient. For this reason, the Intersection Over Union (IoU) method is used for getting more clear data.

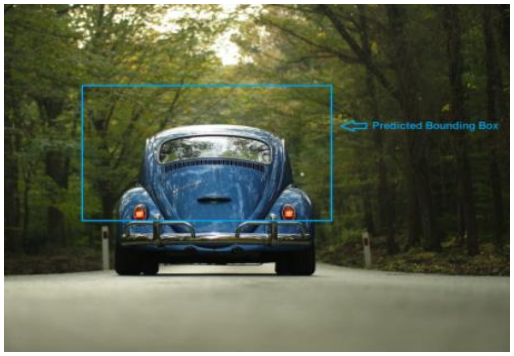

Fig. 5. The Intersection Over Union (IoU)

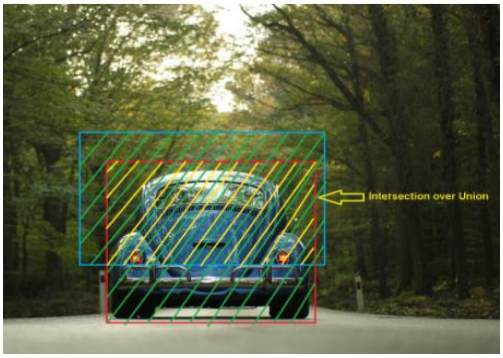

Fig. 6. The Intersection Over Union (IoU)

As shown in Fig. 5 and Fig. 6, the Intersection Over Union matches similar data with the trained images. In this mapping, the areas of the data obtained with the real object are calculated. True, the threshold is kept as high as 0.8 to get the result. By reducing the number of rectangles obtained in this way, the desired object is detected and defined on the system [7].

Training a Specific Object

Firstly, a custom-built weight and config file is required to create a Yolo for a specific object. In order to obtain these files, it is necessary to take and train a large number of photos about that object. The more photos that are trained, the higher the rate of correct results. We used the Labellmg program shown below to train in our project. We convert the files we have to .txt format to create the yolo weight file with the Labellmg program.

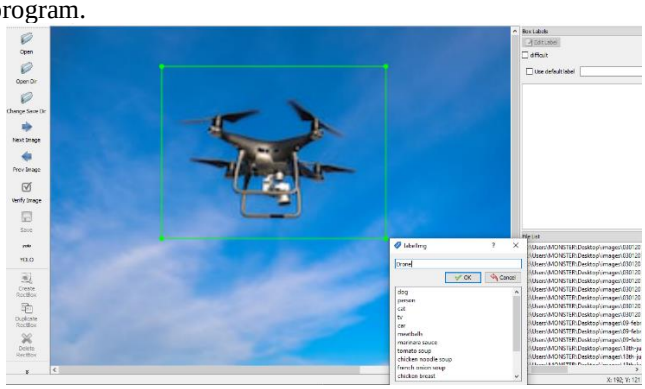

Fig. 7. Training drone on Labellmg

As seen in the Fig.7 above, we need to pay attention that YOLO is written in the commands on the left. Afterward, we take the desired object into a rectangle with the help of the Create RectBox of the photo we have uploaded. Finally, it will ask us to define a name for the rectangle we created and we write the name of the object we have defined there. We do this with all the photos we have. In our project, we found and trained about 1000 drone photos. After completing this process, we output the .txt files we have created with the help of Labellmg as yolov3.weights file via Colab.

Choosing Confidence

Confidence selection, which is one of the most important parameters, should be taken into consideration when Object Detection is performed. To check how the model provides accuracy at various confidence values, need to examine the F1, recall, and precision plots that depend on the confidence value. The confidence value should be determined according to these three charts. F1, recall, and precision calculation formulas can are shown in Eq.1[8].

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (1)$$

$$precision = \frac{TP}{TP + FP} \quad (2)$$

$$recall = \frac{TP}{TP + FN} \quad (3)$$

Eq. 1. Calculation of F1 (1), precision (2), and recall (3).

As can be seen in the Fig. 8 shown below, the higher the confidence value, the higher the precision value.

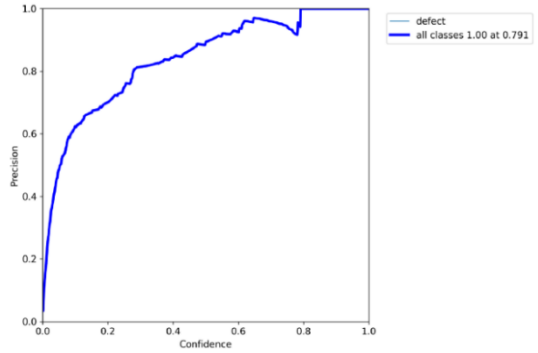

Fig. 8. Relationship between confidence and precision

In the Fig. 9 shown below, the recall value decreases as confidence increases. These two charts show that the confidence value should not be too small or too large.

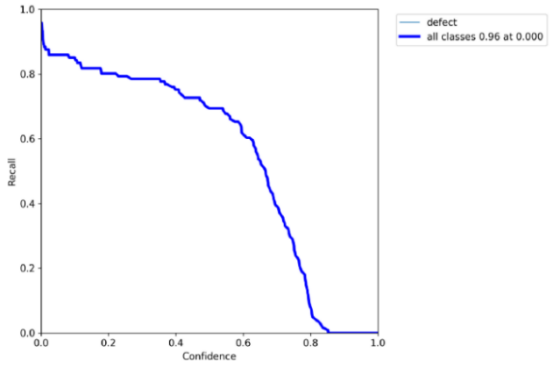

Fig. 9. Relationship between confidence and recall

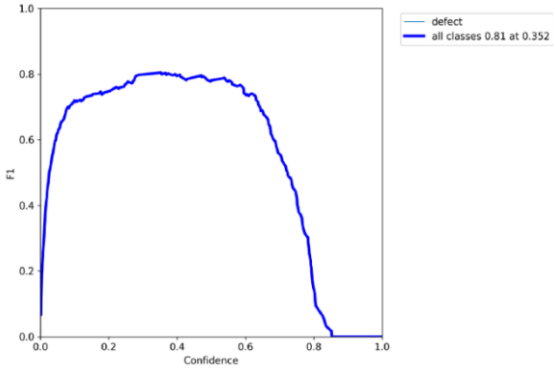

Fig. 10. Relationship between confidence and F1

As seen in the Fig. 10 above, the F1 – Confidence chart was created after the data was taken in the other two charts. This chart shows us in which confidence range we will get the most accuracy. According to the result of this graph, we chose the confidence value to be >0.2 in our project [8].

C. Object Tracking

To track the targeted drone, a tracking algorithm based on object detection with the camera was developed. In this algorithm, the target can be tracked in all x,y, z-axis in the Cartesian coordinate system. The basic idea of the algorithm is taking the targeted drone's rectangle's middle points to coordinate in the x and y-axis and trying to bring the target's middle point to the camera's middle pixels. Also, with the calculation of the area of this rectangle, the algorithm detects whether the drone should approach the target or move away from it. For tracking, yaw angle changing velocity, up-down and forward-backward velocities were controlled and the target drone was tried to fix in the middle point of x and y coordinates of the mini drone's camera.

Even though the drone camera is 1280x720 pixels, the obtained real-time video streaming pixels were arranged to 360x240 pixels to more fast and more accurate detection and tracking. In order to the middle point of the video pixels' coordinates were (180,120) all of the calculations were created due to these coordinates. To arrange the velocity of the drone in yaw angle, the pixels between the target drone's rectangle middle x coordinate and the center point of the drone camera were calculated and velocity determined using PID. The yaw speed calculations code in Python language are located below. Also, the speed in yaw is limited between -100 and 100 as minimum and maximum to secure the drone. The speed values vary 0 - 8 m/s for 0-100 in velocity function.

```
errorx = center_x – 180

sum_errorx = sum_errorx + errorx

speed_yaw = 0.4*errorx + 0.01*sum_errorx +
0.4*(errorx – previous_errorx)

previous_errorx = errorx

speed_yaw = int(np.clip(speed_yaw, -100,100))
```

The example of tracking algorithm is shown in Figure 11.

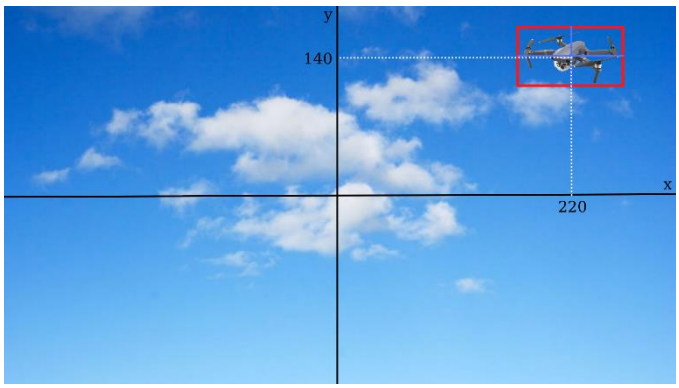

Fig. 11. Tracking algorithm example

To calculate up and down speed, another PID algorithm was used similar to yaw speed. At this time, the summation of the error in the y-axis was also limited to -500 to 500 to obtain a safe flying. The calculations are shown below. Also, the error value was equated to 0 between -5 and +5 intervals to provide stable flying.

```
errory = 120 - center_y
sum_errory = toplam_errory + errory
sum_errory = int(np.clip(sum_errory, -500,500))
speedy = 0.2*errory + 0.01*sum_errory +
0.1*(errory – previous_errory)
previous_errory = errory
if errory <5 and errory >-5:
speedy = 0
```

In the last part of the speed calculations, a PID algorithm was not used for forward and backward speed to provide more secure both the tester and the drone. To calculate speed in forward and backward directions, the area of the target drone's rectangle was used. A tracking interval was determined which equals 40 – 60 cm away from the target. Also, the speed is determined with if-else functions according to tracking intervals in pixels.

```
area = x *y
if area > 6200 and area < 6800:
    forward_backward_speed = 0
elif area > 6800:
    forward_backward_speed = -20
elif area < 6200 and area !=0:
    forward_backward_speed = 20
```

Lastly, all three-speed algorithms of the drone contain a function for staying at the current location when a target is not determined because of the security of the drone and tester in the indoor environment.

D. Autonomous Flying

To track target drones, the mini defense drone has to fly autonomously using information coming from detecting and tracking parts. Tello Drone code library was used to control all of the physical movements of the drone. These movements firstly start with a takeoff command and the drone goes up for 1.5 seconds to get a secure and available tracing position. Then it starts to object detection and the speed information is conducted to Tello's speed function. Lastly, two emergency landing functions were added to the algorithm to obtain safe flying. One of them is if the battery level is critical (below 15 %), the drone lands. Another one is during the drone flying, if the user presses the 'q' button on the keyboard, the drone lands. All of these codes are shown below.

```
me = tello.Tello()
me.connect()
me.streamon()
me.takeoff()
me.send_rc_control(0, 0, 25, 0)
time.sleep(1.5)
me.send_rc_control(0, forward_backward_speed,
speedy, speed_yaw)
if cv2.waitKey(1) & 0xFF == ord('q'):
    me.land()
```

Due to the importance of users should be display critical parameters of drone during flying, these were printed out on the Pycharm console. The monitored parameters and the console picture is shown below Fig. 12.

```
Forward-Backward: -20 Speedy: -8 Yaw Speed: 7 area:  8649 errorX: 9 errorY: -18  battery: 54
Forward-Backward: -20 Speedy: -8 Yaw Speed: 7 area:  8464 errorX: 9 errorY: -18  battery: 54
Forward-Backward: -20 Speedy: -8 Yaw Speed: 6 area:  8281 errorX: 8 errorY: -17  battery: 54
Forward-Backward: -20 Speedy: -8 Yaw Speed: 7 area:  8100 errorX: 9 errorY: -18  battery: 54
Forward-Backward: -20 Speedy: -8 Yaw Speed: 8 area:  8464 errorX: 10 errorY: -18  battery: 54
Forward-Backward: -20 Speedy: -8 Yaw Speed: 7 area:  8281 errorX: 9 errorY: -17  battery: 54
```

Fig. 12. Console output of the algorithm

III. Results And Discussion

In this project, the mini defense drone algorithms were applied to the Tello Drone and the target drone was detected and tracked successfully. The detection of two different drones from their photographs are shown below in Fig. 13 and Fig. 14.

Fig. 13. Drone detection from camera

Fig. 14. Drone detection from camera

However, some limitations were occurred due to the Tello Drone abilities. The first limitation was the detection range of the drone due to the camera's quality. Because of the low camera quality, it can detect a maximum of 5 meters away targets even though it is a 720p 5Mp camera. That's why the algorithm could not be tested more than 5 meters. In addition, due to the weather conditions, all of the tests were made in indoor environments. The second limitation was related to ambiance lights. Because Tello Drone's sensor and camera are affected by light, the drone cannot provide stable flying in places that have low ambient light. The third limitation was the delay between the computer and Tello Drone because there is no onboard computer to make intelligent processes for Tello Drone.

Possible Improvements

The first improvement can be changing the camera with a high quality at least 1080p camera. In this way, the object detection range can be increased. Secondly, due to cameras just giving information about a 2D axis, the depth of the image just depends on assumptions. An additional distance sensor is recommended use to obtain certain and more accurate distance information. The third problem was about computer vision and autonomous commands delays. Because Tello Drone has no components on board to process images and give commands to it, all of the processes are done by the computer which codes are executed. That's why, taking images from a drone with Wi-Fi signals, processing it, and sending the command back cycles take some time and cause approximately 0.7 seconds. This kind of delay is not a problem for a prototype but in the military area, a strong companion computer at least Jetson Nano. The fourth improvement can be replacing the motors with RC high rpm motors by designing and customization all parts of the drone again. In this way, enemy drones can be tracked more easily.

IV. Conclusion

In conclusion, it is aimed to design and implement a mini defense drone to detect and track enemy target drones. In this perspective, computer vision techniques were applied with the help of a camera using a ready-made Tello Drone. Based on the pixel coordinates on the camera of the mini defense drone, the target drone's pixel coordinates were obtained and a tracking algorithm using PID has been developed. As a result, drones could be detected even at close range and followed at the desired tracking proximity. Although some parts of the project are not in the desired place due to physical constraints, the goal has been achieved. The project is open to development in many ways.

References

[1] What is Computer Vision? | IBM. (n.d.). IBM. https://www.ibm.com/topics/computer-vision

[2] Machine Learning: What it is and why it matters. (n.d.). SAS. https://www.sas.com/en_in/insights/analytics/machine-learning.html

[3] Bansari, S. (2021, December 7). Introduction to how CNNs Work - DataDrivenInvestor. Medium. https://medium.datadriveninvestor.com/introduction-to-how-cnns-work-77e0e4cde99b

[4] Kumar, A. (2021, July 6). Computer Vision | How does Computer Vision work. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/everything-happening-in-computer-vision-that-you-should-know/#:%7E:text=Computer%20vision%20uses%20Artificial%20Intelligence,to%20what%20they%20%E2%80%9Csee%E2%80%9D

[5] GeeksforGeeks. (2021, August 5). OpenCV - Overview. https://www.geeksforgeeks.org/opencv-overview/

[6] YOLO: Real-Time Object Detection Explained. (n.d.). V7labs. https://www.v7labs.com/blog/yolo-object-detection

[7] Shivaprasad, P. (2021, December 7). A Comprehensive Guide To Object Detection Using YOLO Framework — Part I. Towards Data Science. https://towardsdatascience.com/object-detection-part1-4dbe5147ad0a

[8] Lebiedzinski, P. (2022, January 6). A Single Number Metric for Evaluating Object Detection Models. Towards Data Science. https://towardsdatascience.com/a-single-number-metric-for-evaluating-object-detection-models-c97f4a98616d