DEEP NEURAL NETWORK FOR AUTONOMOUS UAV NAVIGATION

Jan Klouda
Bachelor Programe (3), FEEC BUT
E-mail: xkloud04@stud.feec.vutbr.cz

Supervised by: Jiří Janoušek
E-mail: xjanou09@stud.feec.vutbr.cz

Abstract: The project deals with autonomous drone control. A neural network is used to create autonomous control for object recognition. This recognition is performed with a ground station, where the computer evaluates the position of the drone and autonomously controls the flight of the drone through the detection of objects.

Keywords: UAV, drone, Python, autonomous navigation, object recognition

1 INTRODUCTION

The article deals with the creation of a fully autonomous drone. The basis of control of this aircraft is the ground station, which receives telemetry data and images from the aircraft. It processes this data and creates an autonomous drone through communication with the flight unit. The ground station contains a neural network that recognizes dozens of objects from the camera image on the drone in real time, this creating an overview of the position of objects. This makes it easy to control a drone and make a various tasks, from tracking a car or a human, through a very precise landing, to reconnaissance or rescue missions. I wrote this whole program in Python and successfully tested it in various situations. Wi-Fi telemetry or a 433 MHz antenna is used for communication between the ground station and the flight unit. I decided to choose a personal computer with a graphics card as the ground station, because the program requires higher performance for smooth running and immediate detection.

2 CONNECTION OF ARTIFICIAL NEURONAL NETWORK WITH FLIGHT UNIT

The connection of the whole system is described in Picture 1 as a block diagram, where the UAV (Unmanned Aerial Vehicle) is controlled by a ground station. The UAV consists of several basic blocks. The flight controller unit here functions as an aircraft control element, calculating data from accelerometer, gyroscope, barometer and GPS coordinates. The flight controller also receives commands from the ground station (personal computer) using wifi telemetry module.

[Diagram of autonomous flight]

Picture 1: Block diagram of autonomous flight
To make the connection, I use a Wi-Fi module connected to the flight controller. I set the flight controller to send flight information (telemetry) and receive control signals via this module. As the Ground Station, on which the computer code runs, I choose a personal computer. To start communication, it is necessary to connect the computer to the created Wi-Fi network, and create MAVLink communication via the UDP (User Datagram Protocol) port. My program first creates this communication channel with the flight controller, and then communicates via MAVLink protocols. MAVLink is a protocol used between drones to transmit control signals, aircraft status information and other configuration messages between the drones and the ground control station. The information is published in a hybrid way, namely publish-subscribe (for sending data streams) and point-to-point (for configuration protocols, mission protocol, or parameters). The so-called dialects, ie message sets supported by a specific MAVlink system, are used to transmit information. Picture 2 describes the structure of the used MAVLink protocol.

![MAVLink Frame – 8-263 bytes](image)

**Picture 2:** Structure MAVLink packet protocol [1]

### 2.1 YOLO NEURAL NETWORK

The use of neural networks for object detection eliminates the disadvantages of cascade detection systems. The difference is that the neural network learns itself, and therefore the monitored object can be detected from different angles. Thus, the accuracy of using neural networks has increased. Thus, a learned neural network is able to detect a person, even if only his hand is visible in the picture. Therefore, I use matrix scales to learn such a neural network, and I gradually adjust them for a given object. Many calculations are required to learn this network and start detection. For example, the Darknet library (used for easier operation of neural networks on a computer) can use the computer's graphics processor to calculate neural networks using the CUDA toolbox, increasing the speed of object detection [2].

The YOLO (You look only once) recognition system works on the same principle as most neural networks, but uses a different approach. It applies a single neural network to the entire image, which divides the image into many frames delimiting objects, at different scales. These frames are created based on the probability for each area. Areas of an image with a high number of frames are thus considered as a very probable area with the occurrence of the searched object. YOLO is a great example of a single-stage detector, and approaches object recognition by considering detection as a regressive problem, taking an input image while learning the coordinates and probability of a bounded frame of a marked class. This means that one convolutional network predicts bounding boxes and probability classes for these boxes simultaneously [3].

Picture 3 illustrates how a program detect a person. This is a direct image of the camera on the drone during the test. The picture shows the marked object, as well as its position in image coordinates, as well as commands sent to the flight unit as "forward" and "right". Based on this information, calculate the relative position of the drone to the monitored object (person). Thus, if a person approaches the drone to a defined distance, the ground station decides that the detected person is too large in the image and sends a command to the flight unit to fly backwards. The same control of the distance between the drone and the object to be observed occurs when the object in the field of view of the camera is too small, and the ground station sends a command to fly forward. If the position of the object leaves the center of the camera's field of view on the drone, ie the person moves to any side, the program adjusts the position and rotation of the drone so that the monitored object is always in the middle of this field of view. Thus, human tracking is ensured. A similar pro-
procedure occurs during landing. The drone begins to descend on the landing surface, and optically monitors where it is located, in this way I created an accurate landing. All these calculations and controls are performed 20x to 30x per second, so that the control is sufficiently fine and thus responds immediately to a change in the position of the object. The biggest slowdown in code is the detection of neural networks, and when using a more powerful graphics processor, this detection becomes faster and control can theoretically speed up.

![Tracking a moving person](image)

**Picture 3:** Tracking a moving person

### 2.2 Code Sample

I wrote the whole program in Python. Most often I used the OpenCv libraries (for image preprocessing and image field creation), the Dronekit library (for creating MAVLink packets and thus ensuring communication between the flight unit and the ground station) and the Darknet library, which uses YoLo neural networks and using this libraries detect objects. Source code 1 describes the detection of an object in each camera frame using a neural network, and then calculates the coordinates of the object in the field of view. Source code 2 describes the creation of the MAVLink protocol for sending control commands directly to the drone flight unit [4].

```python
if frame_resized is not None:
    image = darknet.draw_boxes(detections, frame_resized, class_colors)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    ax = dx = dy = en = w = 0
directionx = directiony = ""
    dy, dx, en, w = darknet.middle(detections, dy, dx)
    if en == 1:
        dx = (width/2) - dx
        dy = (height/2) - dy
        if w < 150:
            ax = gnd_speed
            directiony = "vpred"
        if w > 250:
            ax = -gnd_speed
            directiony = "vzad"
```

*Source code 1:* Source code for calculating the position of a detected object
def set_velocity_body(vehicle, vx, vy, vz):
    msg = vehicle.message_factory.set_position_target_local_ned_encode(
        0, 0, 0,
        mavutil.mavlink.MAV_FRAME_BODY_NED,
        0b000011111000111,
        vx, vy, vz,
        0, 0, 0,
        0, 0)
    vehicle.send_mavlink(msg)
    vehicle.flush()

Source code 2: Source code for sending control signals to the flight unit

3 CONCLUSION

In making this project, i used multiple neural networks and different types of object detection. In the project, i solved the use of neural networks for autonomous control and created an autonomous UAV that is controlled based on object recognition. I connected the neural networks to the flight unit and created a program for autonomous control. YoLo neural networks were the most accurate for my use and detected multiple objects at once, so i could create subconscious places where the drone is located, and then create a program that uses this data to evaluate where the drone should move. The whole system is autonomous, and during the flight the drone is controlled only by computer code commands. The tests are documented on camera. Because communication with the ground unit is required when using the YoLo neural network, i have created a program that uses a similar detection method but is not hardware intensive. This code modification is also suitable for Raspberry Pi microcomputers. The microcomputer is mounted on the drone and the system thus becomes fully independent. I have also successfully tested this modification. I also use this project in my bachelor's thesis, where i evaluate the results of detection, various modifications of my code, and evaluate the results of tests.

REFERENCE


