Object Detection and Mapping with Unmanned Aerial Vehicles Using Convolutional Neural Networks

Stefan Hensel[1], Marin B. Marinov[2(✉)], and Max Schmitt[1]

[1] Department for Electrical Engineering, University of Applied Sciences Offenburg, Badstraße 24, 77652 Offenburg, Germany
[2] Department of Electronics, Technical University of Sofia, 8, Kliment Ohridski Blvd., 1756 Sofia, Bulgaria
mbm@tu-sofia.bg

Abstract. Significant progress has been made in the field of deep learning through intensive research over the last decade. So-called convolutional neural networks are an essential component of this research. In this type of neural network, the mathematical convolution operator is used to extract characteristics or anomalies. The purpose of this work is to investigate the extent to which it is possible in certain initial settings to input aerial recordings and flight data of Unmanned Aerial Vehicles (UAVs) in the architecture of a neural network and to detect and map an object. Using the calculated contours or dimensions of the so-called bounding boxes, the position of the objects can be determined relative to the current UAV location.

Keywords: Computer vision · Object detection · Deep learning · Convolutional neural network

1 Introduction

1.1 Motivation

A significant advancement in development due to scientific research has taken place in the field of deep learning over the past decade. The so-called convolutional neural networks (CNN) are an essential component. In this type of neural network, the mathematical convolution operator is used to extract features or abnormalities. Above all, very positive results can be found in the field of object detection using these networks. The research results in completely new, different application possibilities, all of which require analysis, whereby each application case must be based on an adapted selection of data for the training process [1].

Closely related to neural networks and their advantages for object detection is the associated mapping of objects to enable ongoing data analysis [2, 3]. Any device capable of recording, paired with an option for determining the position, can be able to carry out such a task and to put the detected objects on a local or global map.

D. Perakovic and L. Knapcikova (Eds.): FABULOUS 2021, LNICST 382, pp. 254–267, 2021.
https://doi.org/10.1007/978-3-030-78459-1_19

1.2 Task

This work aims to investigate to what extent it is possible, under certain starting points, to introduce aerial recordings and flight data of a UAV into the architecture of a neural network and to carry out object detection. With the help of the calculated contours, respectively dimensions of the so-called *Bounding Boxes*, the position of the objects relative to the current UAV location should be determined.

Vehicles in public parking lots should serve as objects for object detection. Despite changes in the direction of orientation and the flight altitude of the UAV, these objects should be stored with a high degree of accuracy in a global map [4].

1.3 Approach

To solve the task of detecting and mapping objects, the constant, secure flight operations of the UAV, for whose control option a separate controller is available, must first be enabled and guaranteed. A Python script must therefore be developed which is responsible for receiving the recorded image data and the associated measured values of flight parameters (e.g. positioning data, flight altitude, etc.). These should then be made accessible via the middleware ROS (Robot Operating System) for further components to solve the task.

The CNN YOLO was chosen for the detection of vehicles from aerial photographs. To make this network-aware of this type of data, an efficient training procedure is crucial. To enable this process, sufficient data for this process must be collected in advance with a successful implementation of the UAV script to enable the most robust detection possible with a high probability of detection. With the use of the measurement parameters of the UAV and the YOLO neural network, operated in the background, vehicles are ultimate to be positioned relative to the current UAV position. To solve this, the global positioning data, combined with the results of the object detection, must be included in mathematical calculations. Additional requirements (e.g. independence of the UAV's direction of orientation) are to be implemented.

2 Materials and Methods

2.1 Convolutional Neural Network

The most widespread algorithm among the deep learning algorithms is CNN, a class of artificial neural networks. Due to their impressive results in a competition for object recognition, known as the Image Net Large Scale Visual Recognition Competition in 2012, the performance of the convolution networks is steadily increasing [5].

2.1.1 Building Blocks of the CNN Architecture

The architecture of a CNN essentially consists of three types of layers:

- Convolutional Layer,
- Pooling Layer,
- Fully Connected Layer.

A typical network architecture consists of several repetitions of these types of layers. Different architectures can be created, each of which is advantageous for certain detection tasks. The convolution used in these networks is a linear operator that is used to extract features from an input image. Using a filter core and an element-wise product and summation with a tensor, which is derived from an array of the individual pixel values of the input image, an output tensor is created for each position. The entirety of all output tensors results in a so-called feature map [5].

Figure 1 shows an example of the process of convolution of an input tensor with a filter kernel, from which the feature map is created. The pooling layer represents a method for reducing the dimension of the feature maps to obtain translation independence from small displacements and distortions. Furthermore, the number of parameters to be learned is reduced below this layer. It should be noted that there are no parameters to be learned in the pooling layers [5].

The pooling layer is used to modify the output value of the next layer. This function replaces the output of the network at a certain position with a value that depends on the surrounding value neighborhood. Examples of pooling functions are [6]:

1) Max Pooling: Return of the maximum from a square region.
2) Average Pooling1: Returns the mean value from a square region.
3) L2 pooling: L2 norm from a square region.
4) Weighted Average Pooling: Returns the weighted average based on the distance from the center.

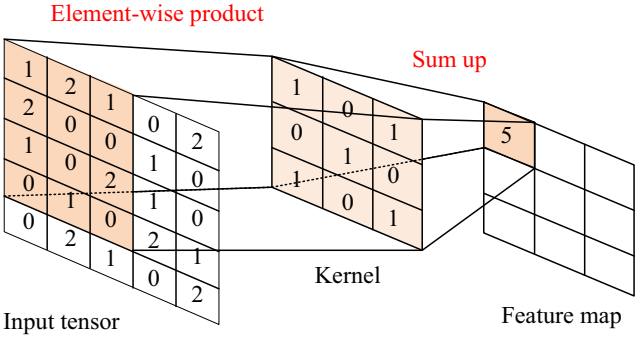

Fig. 1. Example of the process of convolution of an input tensor with a filter kernel of size 3 × 3 [5].

The fully connected layer is applied to the last, final, convolution, or pooling layer. Usually, a one-dimensional array or a one-dimensional vector is connected to form one or more fully networked layers. Each input value with a learnable weight is connected to an output. These outputs are then passed through a specific activation function, which assigns the outputs of the last fully connected layer to a probability of specific class affiliation [5].

2.1.2 CNN – Training

Training a neural network is the process of finding optimal filter cores in convolution layers and the weights that are used within the fully connected layers. For this purpose, training data sets are used which contain the so-called ground truth labels.

These ground truth labels are manually created data sets that contain the exact position of the objects in an image with the corresponding class affiliation. The training is intended to minimize the differences between the predictions and the given Ground Truth Labels [7]. There are essentially two steps that have to be carried out for this [5].

1) Forward Propagation: Calculation of the model performance using initial filter kernels and weights using the calculation of a Loss Function with the help of a training data set.
2) Backpropagation: Optimization algorithm that updates the learnable parameters (filter kernels and weights or hyperparameters) based on the values of the loss function and the gradient descent

The gradient descent is used to change the learnable parameters so that the value of the loss function is minimal. The gradient is the direction of the loss function in which it has the steepest rise. Each learnable parameter is changed so that it points in the opposite direction of the gradient [5]. The gradient results from

$$\mathrm{w} := \mathrm{w} - \alpha \cdot \frac{\partial L}{\partial w}, \tag{1}$$

where w stands for a trainable parameter, α for the learning rate, and L for the loss function. Since the calculation effort for the gradient of the loss function is very high, concerning the learnable parameters, a reduction is achieved using a subset of data records, the so-called mini-batches. This method is called Stochastic Gradient Descent [8].

The following Fig. 2 is intended to clarify the sequence of the gradient descent.

2.2 YOLO - You Only Look Once

The neural convolution network YOLO, which is used in this work, is a real-time capable object detector. The so-called mean average precision (mAP for short) is a standardized metric that provides information about the detection capability concerning all object categories of a neural network. Intersection over Union (IoU) is used to determine the accuracy of the object localization. If IoU is greater than a threshold value of 0.54, the object is declared as successfully detected [7, 9]. In addition to the performance of YOLO, Table 1 shows other common neuronal convolution networks. It can be seen that YOLO still achieves an mAP of 28.2%, mainly due to its short computing time of 22 ms. Although mAPs of up to 37.8% can be achieved with other networks, these require a duration of ≈200 ms for object detection.

YOLO is also available in the fourth version. However, since this version was only published in the course of this work, YOLOv3 will continue to be used in this work. In the following Subsect. 2.2.1, therefore, only the current structure of the neural convolution network is explained, as it is also used in this scientific work

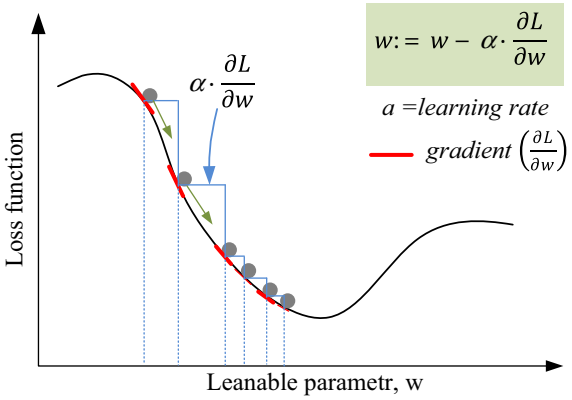

Fig. 2. Illustration of the gradient descent: The gradient of the loss function points in the direction of the steepest rise, the learnable parameters are updated in the negative direction of the gradient (adapted from [5]).

Table 1. mAP and computing time in selected neural convolutional networks [10].

Method	mAP	Time, ms
SSD321	28.0	61
DSSD321	28.0	85
R-FCN	29.9	85
SSD513	31.2	125
DSSD513	33.2	156
FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
YOLOv3-320	28.2	22
YOLOv3-416	31.0	29

2.2.1 Network Structure

Due to a high number of localization errors as well as a low reproducibility of the detections, which the convolution network YOLO has in version 1, various changes have been made. These are mainly intended to remedy both of the problems mentioned, but maintain the classification accuracy of version 1 [11]. To increase the detection capability, various approaches are introduced in version 2, which, due to their success, will continue to be found in version 3 that follows. The neural network consists of 106 convolution layers in total. It is based solely on feature detection on its network, the so-called Darknet-53. Darknet-53 is responsible for extracting features. The entire YOLO convolution network is presented in detail in [12]. The process of detecting features of the entire YOLO network can be divided into 7 successive steps [13].

A special feature of the network is the prediction of bounding boxes on three different scale levels. This should ensure that even small objects can be detected and predicted. The center point of each feature map is responsible for predicting the object. There are three different scale entries in each feature map [10].

$$B(C + 5), \tag{2}$$

B is the number of Bounding Boxes each cell can predict. The value C, in turn, defines the attributes that each grid cell can have [14]:

- Bounding Box coordinates,
- Objectness Score: the probability that an object can be found within the bounding box,
- Class Score: Probability that the object belongs to a certain, predefined object class (e.g. car, bus).

2.3 ROS - Robot Operating System

The Robot Operating System, which uses the publisher-subscriber communication model, is available for communication and data exchange between the UAV and the computer or to enable data transfer in isolation on the latter. Due to their widespread use and frequent use in robot applications, there is a large collection of software tools and libraries available [15]. The platform offers the option of either creating complete software applications for computers, robots, or sensors yourself or using algorithms that have already been implemented. The data exchange between two nodes or users takes place via topics. The data is published via a topic, which can then be subscribed to by one or more nodes [16].

2.3.1 YOLO ROS: Real-Time Object Detection for ROS

The neural network responsible for the detection is a ROS package [17], which is made available on the GitHub development platform. This GitHub repository offers the core structure for the detection of objects in image or video data as well as the possibility to use the corresponding information of the neural network (e.g. coordinates of the bounding box) via ROS for further data processing. For complete implementation or the detection of cars, it is necessary to make changes within the corresponding directories. For this purpose, in addition to the topic of the video file, the launch file must also be supplied with the weights and the network structure of the neural network for the task of detection from a bird's eye view.

2.3.2 YOLO v4, v3, and v2 for Windows and Linux [18]

A separate GitHub repository is used for training the YOLO convolution network, which in addition to the ones for the neural network from Subsect. 2.3.1 necessary weights also provide information about the mAP. In Sect. 3.2 it is explained in more detail which settings or adaptations must be made to YOLO to obtain a solution to the task of the project [10].

2.4 Parrot Anafi

The UAV used in this work is a quadrocopter from the company "Parrot". Due to the multitude of possible uses and the additional software applications that are necessary for the operation of the UAV "Anafi", the manufacturer provides the independent application Olympe. Olympe is an interface based on the programming language Python, which offers the possibility of transmitting simulated or real UAV data to the computer [19]. To get a better overview of the UAV used, some key figures are listed below:

- Size of the ready-to-fly UAV: 175 × 240 × 65 mm
- Weight: 0.320 kg
- Maximum transmission range: 4000 m
- Maximum altitude: 4500 m above sea level
- Maximum horizontal flight speed: 15 m/s
- Maximum vertical flight speed: 4 m/s
- Satellite positioning systems: GPS, GLONASS

An excerpt from the data-sheet is given to clarify the image processing sensor built into the UAV:

- Sensor: 1/24″ CMOS
- Video resolution: optionally 4096 × 2160 with 24 fps, 3840 × 2160 with 24/25/30 fps, 1920 × 1080 with 24/25/30/48/50/60 fps

A controller is available to control the UAV, which, in addition to controlling it, also shows the possibility of making changes to the flight characteristics and camera settings in conjunction with a smartphone. This flight controller is characterized as follows:

- Weight: 0.386 kg
- Transmission system: Wi-Fi 802.11a/b/g/n
- Operating frequencies: 2.4–5.8 GHz
- Live stream resolution: 720p
- USB connection: USB-C (charging connection), USB-A (for connection to smartphone)

The UAV has a mechanical-electrical 3-axis image stabilizer to obtain the most stable aerial recording possible by a camera. Further technical data of the UAV or the controller can be found in the technical data sheet [20].

3 Implementation

To achieve a solution to the task, it is first necessary to transmit the flight and image data of the UAV to the computer. Only the image data is passed on to the YOLO neural network to carry out object detection. The position information, as well as the flight altitude of the UAV, are fed into an exclusively prepared Python script, which merges this data with the detected objects or the bounding box coordinates. The following Fig. 3

should clarify the flow of information, starting with the UAV. It also explains which sub-processes are processed online and offline, whereby these two terms are intended to mean a current or subsequent evaluation of the data.

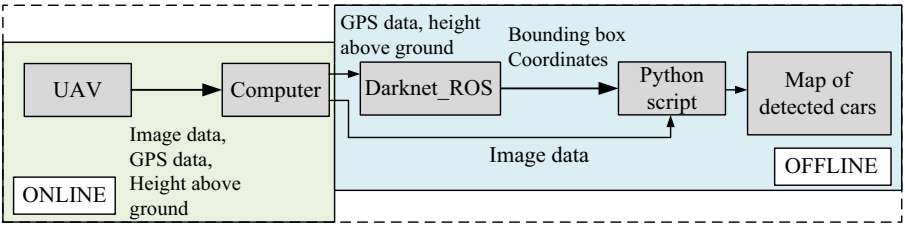

Fig. 3. Clarification of the data flow.

The following subsections go deeper into the sub-processes and are intended to clarify the respective solution approaches.

3.1 UAV Data Acquisition

To receive the flight data of the UAV, which is essentially composed of the image and position data, it is necessary to adhere to a specified flow of information. For this purpose, the data of the UAV are transmitted to the Skycontroller and then are forwarded to the computer. The flight information can be read out with the appropriate script via the bus connection (from USB Type C to USB A). It should be mentioned here that the UAV can only establish a single connection to a single software development kit (SDK for short). For this reason, it is not possible to establish a direct connection from a computer to the UAV and to maneuver it simultaneously with the Skycontroller. As in Sect. 2.4, the sample algorithms to be found online are used to create this Python script, whereby these must be significantly adapted and expanded to include the required functionalities.

3.2 YOLO Neural Network Training

To prepare the YOLO convolutional network as best as possible for the application problem at hand, a data record with ground truth labels of the cars must be created. Since no publicly accessible data sets with cars from a bird's eye view are available during the implementation of this project, these are recorded after successful implementation of the Python script from the previous Sect. 3.1 using the public parking lot of the Offenburg University of Applied Sciences.

The Visual Object Tagging Tool, or VoTT for short, from Microsoft, is used to create the Ground Truth Boxes and the corresponding Ground Truth Labels. This offers the possibility of extracting individual images from a video and drawing the cars contained therein with the appropriate label for each image. If you do this for all recorded videos, the following distributions result for the training, test, and validation data set1 is available:

- Number of training data: 350,
- Number of validation data: 100,
- Number of test data: 50.

Due to the limited number of cars available and the associated insufficient amount of training data, different flight patterns, and data from an altitude in the interval between 10–15 m are recorded. This should keep the neural network as independent as possible of the current yaw angle of the UAV and the size of the objects.

To obtain the weights necessary for the neural network, the GitHub repository YOLO v4, v3, and v2 for Windows and Linux (see Subsect. 2.3.2) is used for training. For this, it is necessary to adjust some parameters within this repository. The configuration file from

```
yolov3.cfg
```

as well as the pre-trained weights of

```
darknet53.conv.74
```

The changes that exist within the configuration file for the training step are shown below. These are carried out according to the specifications of the repository [18]:

```
batch = 64
subdivisions = 64
width = 608
height = 608
max_batches = 6000
steps = 4800, 5400
```

Since this project deals with the detection of only one single class “car”, this fact must also be taken into account in all YOLO layers.

```
classes = 1
```

Furthermore, the size of the filters is adjusted before each last YOLO layer. As already in Subsect. 2.2.1 with Eq. (2), the size should be set to the value 18 due to a single class, because:

$$B(C + 5) = 3(1 + 5) = 18. \tag{3}$$

```
filters = 18
```

To indicate to the neural network, the paths in which the training and validation data are stored, a file must be created as follows. This also contains the path to a file in which the names of the classes are created. For this, the row index stands for the corresponding class that the neural network detects. The different weights that can be found every 1000 iterations are saved in the backup path.

```
classes = 1
train = data_Schmitt/train.txt
test = data_Schmitt/valid.txt
names = obj.names
backup = backup/
```

The images and the .txt files of the same name are stored in the two training and test paths. Furthermore, the following format is to be found in every .txt file, in which each bounding box is assigned a class and the coordinates of this box:

```
<object-class> <x_center> <y_center> <width> <height>
```

To start the training, the following code must be entered and confirmed in a terminal in the relevant management:

```
./darknet detector train data_Schmitt/obj.data ...
cfg/yolov3_MODIFIED.cfg darknet53.conv.74 -map
```

To avoid the problem of overfitting to the specified training data, it is possible, using the validation data, to output the mAP for the stored weights of different iteration steps. If you do this, the following values result:

Table 2. mAP after different iteration steps.

Test run	4000.weights	5000.weights	6000.weights	best.weights	final.weights
1	90.45%	87.84%	90.24%	91.86%	90.24%
2	94.05%	91.35%	93.93%	90.85%	93.93%

It can thus be seen that the mAP with the highest value is present in test run 2. Therefore, these weights are used for the detection of the vehicles.

4 Experimental Results

To achieve the best possible quality of the evaluation and to have a guideline for the implementation, this work adheres to the basic structures of the Standard for Evaluation of the Society for Evaluation e.V. [21].

During the investigations and evaluations of the results, the following questions, among others, should be answered:

- Is the amount of training data sufficient so that sufficient vehicles can be detected?
- Is the precision of the bounding boxes of the neural network sufficient to enable the position of the vehicles to be determined?
- Is the mathematical calculation process for obtaining the positions correctly?

- How precisely are the vehicle positions determined?
- Which optimization potential can be seen?
- Can this project be the basis for further research projects that deal with the detection and mapping of objects?

Since the parking lot of the Offenburg University of Applied Sciences serves as the object for the evaluation, the dimensions of the parking bays are presented in advance. (These apply to all selected scenarios):

- Width: 2.5 m,
- Length: 3.95 m,
- Distance between two parking strips: 5.45 m.

4.1 Mapping of a Section of a Parking Lot with a Stationary UAV

To investigate the mapping accuracy of the underlying mathematical algorithms, an aerial photograph of a car park section at the Offenburg University of Applied Sciences should first be carried out in this section. The UAV remains in the so-called hovering mode during the entire recording, i.e. without or with minimal movement in the x, y, or z components. The following Fig. 4 shows an aerial photograph of this situation, which has already experienced the process of object detection using the YOLO neural network. If you look at Fig. 4, a first insight must be made clear here. Due to variances in the detection and the associated entry of the bounding boxes in the image, it is the case that the horizontal and vertical lines of these boxes do not completely coincide with the vehicle contours.

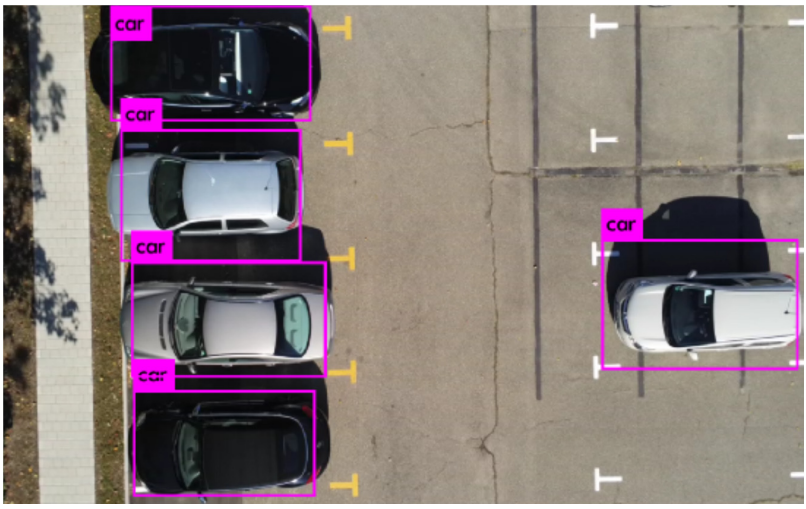

Fig. 4. Presentation of the vehicle positions with the associated bounding boxes drawn by YOLO.

This results in inaccuracies in determining the vehicle. Analogously to this, Fig. 5 shows the result of entering the positions using the self-created Python script which uses mathematical algorithms from [22].

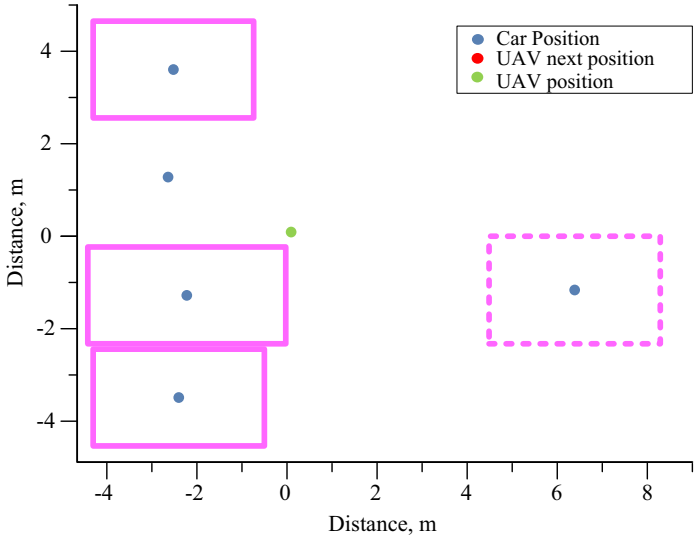

Fig. 5. Result of the vehicle identification with an entry of the drone position.

If one looks at Fig. 5 and the associated dimensions of the Bounding Boxes of the detected cars, an initial plausibility of the dimensions of these boxes can be confirmed. The motor vehicle is used at position $(x, y) = (6, 5 - 1, 3)$ to confirm the correct calculation of the coordinates of the position and the dimensions of the bounding boxes. Due to the selected starting point of the UAV directly on the parking lot marking, it is possible to determine the center of the vehicle with the coordinates just mentioned and thus to calculate $(x_{real}, y_{real}) = (7, 425, 1, 25)$, with tolerances in the determination of the car position, must be taken into account by the dimensions of the parking bay. However, since this is only an estimate, (x_{real}, y_{real}) are assumed to be actual values. This results in an uncertainty of $\left(1 - \left|\frac{6,5}{7,425}\right|\right) \cdot 100\% = 12,45\%$ for the x component and $\left(1 - \left|\frac{-1,3}{1,25}\right|\right) \cdot 100\% = -4\%$ for the y component. Since the centers of the bounding boxes are used to determine the deviations and there is thus a dependency on the accuracy of the detection of the neural network, an increased inconsistency for the x component can be explained.

To answer all the evaluation questions dealt with at the beginning, the performance of the YOLO neural network must finally be addressed. It has been shown that the detection capability, especially (as already mentioned) with slow flight maneuvers, provides considerably more stable and robust results. To generally improve the performance of YOLO, it would be necessary to create a larger amount of training data (training, validation, and test data set as a whole) and to obtain an increase in robustness. Although the mAP is high at 94.05% (see Table 2), this value is calculated using static image data.

5 Conclusions and Future Work

The ready-to-fly drone provided the opportunity to prepare training data for the YOLO neural network. For this purpose, aerial photos of parked cars were taken, which should show the vehicles from as varied as possible angles and different flight altitudes. This procedure is intended to counteract a possible dependency on certain arrangements of the objects and to develop robust detection of the network.

Furthermore, the current behavior of YOLO was discussed and the need for a larger data set for the training process was explained.

If the results of the evaluation are included in the outlook of this work, the potential of the task can be assessed. It turns out that the question of the performance of the detection of the vehicles and the subsequent mapping works precisely within the framework of tolerances. There would be the option of not only performing all mathematical calculations offline afterward but also performing all of these filterings of the redundant results directly after the detection with a powerful, mobile computer.

Acknowledgments. The authors would like to thank the Research and Development Sector at the Technical University of Sofia for the financial support.

This research is partly supported by the Bulgarian National Science Fund in the scope of the project "Exploration the application of statistics and machine learning in electronics" under contract number КП-06-Н42/1.

References

1. Al-Saffar, A.A.M., Tao, H., Talab, M.A.: Review of deep convolution neural network in image classification. In: 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), Jakarta (2017)
2. Saetchnikov, I., Tcherniavskaia, E.A., Skakun, V.V.: Object detection for unmanned aerial vehicle camera via convolutional neural networks. IEEE J. Miniaturizat. Air Space Syst. (2020)
3. Hensel, S., Marinov, M., Schmitt, M.: Experimental setup for investigation and evaluation of a mapping and localization system. In: Proceedings of the 9th FDIBA Conference - Challenges of the Digital World, Sofia, Bulgaria, 28–29 November 2019
4. Wu, Q., Zhou, Y.: Real-time object detection based on unmanned aerial vehicle. In: 2019 IEEE 8th Data-Driven Control and Learning Systems Conference (DDCLS), Dali, China (2019)
5. Yamashita, R., Nishio, M., Do, R., Togashi, K.: Convolutional neural networks: an overview and application in radiology. Insights Imaging **9**, 611–629 (2018)
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge (2016)
7. Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: a survey. ArXiv, p. abs/1905.05055 (2019)
8. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML'15: Proceedings of the 32nd International Conference on Machine Learning, July 2015
9. Tao, J., Wang, H., Zhang, X., Li, X., Yang, H.: An object detection system based on YOLO in traffic scene. In: 2017 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian (2017)
10. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement (2018)

11. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI (2017)
12. Zoev, I.V., Beresnev, A.P., Markov, N.G.: Convolutional neural networks of the YOLO class in computer vision systems for mobile robotic complexes. In: 2019 International Siberian Conference on Control and Communications (SIBCON), Tomsk, Russia (2019)
13. Mao, Q., Sun, H., Liu, Y., Jia, R.: Mini-YOLOv3: real-time object detector for embedded applications. IEEE Access **7**, 133529–133538 (2019)
14. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV (2016)
15. Blasdel, et al.: About ROS.Version: 2020. https://www.ros.org/about-ros/
16. Quigley, M., et al.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
17. Bjelonic, M.: YOLO ROS: Real-Time Object Detection for ROS, 2016–2020. https://github.com/leggedrobotics/darknet_ros
18. Redmon, B.A.: Darknet: YOLO v3 - neural network for object detection (2020). https://github.com/AlexeyAB/darknet
19. Parrot Drone SAS: Olympe - Overview (2019). https://developer.parrot.com/docs/olympe/overview.html
20. Anafi: The flying 4 K HDR camera that you can take (2020). https://www.parrot.com/assets/s3fs-public/2020-07/bd_anafi_productsheet_en_210x297_2018-06-04.pdf
21. Deutsche Gesellschaft für Evaluation e.V., Standards für Evaluationen, Köln: Redaktion: Dr. Wolfgang Beywl, Zimmermann-Medien (2002)
22. Lowhead, J.: Learning GeoSpatial Analysis with Python, 2nd edn. Packt, Birmingham (2015)