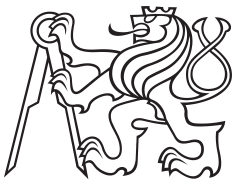
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Implementation of a computer vision algorithm for onboard detection of unmanned aircraft

Lukáš Bauer

Supervisor: Ing. Martin Saska, Dr. rer. nat.
Field of study: Cybernetics and Robotics
Subfield: Robotics
May 2018

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Bauer Lukáš** Personal ID number: **456970**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Implementation of a Computer Vision Algorithm for Onboard Detection of Unmanned Aircraft

Bachelor's thesis title in Czech:

Implementace algoritmu počítačového vidění pro onboard detekci bezpilotních letounů

Guidelines:

The focus of this thesis is to design and implement a vision-based algorithm for detection of UAVs in image from an onboard camera of a UAV.
1. Research possible approaches for feasible, real-time detection of an object class from images obtained by a camera carried by a UAV.
2. Select, implement and optimize one such method as a Robot Operating System (ROS) node interoperable with other flight systems onboard available platforms, for detection and relative localization of neighbor UAVs.
3. Test the method in simulation and on a dataset obtained from real-world flight.
4. Evaluate the precision and computational speed on onboard computers of the available UAVs.
5. Prepare the system for integration into formation control algorithm implemented by other students.

Bibliography / sources:

[1] R. Szelisky, "Computer Vision: Algotrithms and Applications (1st ed.)," Springer-Verlag New York, Inc., New York, NY, 2010
[2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) , pp. 6517-6525, Honolulu, HI, 2017
[3] C. M. Bishop, "Neural networks for pattern recognition" Oxford university press, 1995

Name and workplace of bachelor's thesis supervisor:

Ing. Martin Saska, Dr. rer. nat., Multi-robot Systems, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **12.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Martin Saska, Dr. rer. nat.	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt

Student's signature

Acknowledgements

I would like to thank my family for the support provided to me during my studies, my supervisor for all the help and advice given to me and last but not least, all members of the MRS team for their support and help during the experiments.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date 24. May 2018

Abstract

In recent years, there has been an increase of interest in formation control of unmanned aerial vehicles (UAVs) [1]. Many interesting applications have been studied, including forest fire monitoring [2], or searching through a designated region [3]. During these tasks, UAVs need to maintain the formation. To maintain the formation, relative position of UAVs is needed. In this thesis, the YOLOv2 visual detector was implemented as a ROS node operating onboard of a UAV, to be used in formation control. The detector was modified for relative position estimation from the image and tested in leader-follower experiment both in gazebo simulator and in real world.

Keywords: UAV, visual detector, YOLOv2, UAV formation control

Supervisor: Ing. Martin Saska, Dr. rer. nat.
Multi-robot Systems,
FEE

Abstrakt

V posledních letech byl nárust zájmu o řízení formací bezpilotních letounů (UAV) [1]. Bylo zkoumáno mnoho zajímavých uplatnění, včetně monitorování lesních požárů [2], nebo prohledávání požadovaného území. Během těchto úloh musí letouny udržet formaci. Pro udržení formace je potřeba znát relativní pozice UAV. V této práci byl použit YOLOv2 vizuální detektor, který byl implementován jako ROS node operující přímo na letounu pro použití v řízení formace. Tento detektor byl modifikován pro odhad relativní pozice z obrazu a otestován jak v gazebo simulátoru, tak ve skutečnosti.

Klíčová slova: UAV, vizuální detektor, YOLOv2, řízení formací UAV

Překlad názvu: Implementace algoritmu počítačového vidění pro onboard detekci bezpilotních letounů —

Contents

Project Specification iii

1 Introduction 1

1.1 A brief history of object recognition 1

1.2 Thesis overview 2

Part I
Visual object recognition methods

2 Approaches to visual object recognition 5

2.1 Template matching 5

2.2 Color based 5

2.3 Feature-based methods 6

2.4 Neural networks for visual object recognition 6

2.4.1 Multilayer feedforward neural network 6

2.4.2 Convolutional neural network . 7

3 State-of-the-art CNN detectors 9

3.1 Faster R-CNN 9

3.2 SSD - Single shot multibox detector 9

3.3 YOLO - You Only Look Once . . 10

Part II
Implementation of the YOLOv2 algorithm and experiments

4 Implementation of the YOLOv2 algorithm 15

4.1 Network training 15

4.2 Separating detections and filtering out false positives 16

4.3 Calculating the real-world position of a detected UAV 16

4.3.1 Camera calibration 17

4.3.2 Position of a detected UAV relative to the camera 18

4.3.3 Transforming the object position to coordinate system of the UAV 20

5 Experiments 21

5.1 Performance of YOLOv2 21

5.2 Leader-follower experiment 24

5.3 Precision of the relative position estimation 28

5.3.1 Experimental results 31

6 Conclusion **35**

Appendices

A Bibliography **39**

Figures

2.1 An example of a simple feedforward neural network [5] 7

2.2 Structure of a CNN [6] 8

4.1 Pinhole camera model [31] 17

4.2 Example image from camera calibration process. The distortion of the lens is clearly seen 18

4.3 Example image from camera calibration when holding the chessboard at the left side. The width of the board is almost the same as in front of the camera, even though after undistortion it should appear smaller due to perspective projection 18

5.1 Detected UAV in the gazebo simulator 22

5.2 The detector does not see the UAV on the right 22

5.3 The detector can not separate the UAV from the background 22

5.4 The detector identifies this UAV after it was flying in front of the sun. 23

5.5 The detector detected both UAVs in the image. The UAV on the left is more than 20 meters away 23

5.6 False positives in this image are filtered out. 24

5.7 Multiple detections of the same UAV. These should also be filtered out by the described filter 24

5.8 The desired and ground truth position of the follower UAV in X and Z axis 26

5.9 The deviation from ground truth position of the follower UAV in X and Z axis 26

5.10 The desired and ground truth position of the follower UAV in X and Z axis 27

5.11 The deviation from ground truth position of the follower UAV in X and Z axis 28

5.12 Transformation from the global coordinate system (O) to the UAV frame (O') 30

5.13 Comparison of the relative UAV position estimated from the bounding boxes received from the detector and the relative UAV position calculated from the ground truth 31

5.14 The deviation of the relative UAV position estimated from the bounding boxes received from the detector with respect to UAV position calculated from the ground truth 32

5.15 The deviation of the relative UAV position estimated from the bounding boxes received from the detector with respect to UAV position calculated from the ground truth based on the distance from image center in pixels (L) 33

Tables

2.1 Overview of the various color models and their invariance to various imaging conditions. (+ denotes invariant and − denotes sensitivity of the color model to the imaging condition.) 6

Chapter 1

Introduction

In recent years, there has been an interest for formation control and obstacle avoidance of unmanned aerial vehicles (UAVs) [1]. Many interesting applications have been studied, including forest fire monitoring [2], or searching through a designated region [3]. During these tasks, UAVs need to maintain the formation. This could be achieved, for example, by using GNSS to determine relative position between UAVs from their known absolute positions [4], but there are numerous cases, for example navigation through a forest, when there such systems are unreliable. In those cases, we can use a camera mounted on a UAV to detect and localize neighbors, in order to maintain the formation. To achieve this goal, we need a fast and precise visual object detector, that can run onboard of a UAV. The aim of this thesis is to select and integrate such a detector into the pre-existing ROS[1]-based system used by MRS groub.

1.1 A brief history of object recognition

In the 1980s, a lot of attention was focused on more sophisticated mathematical techniques for performing quantitative image and scene analysis. Image pyramids started being widely used to perform tasks such as image blending and coarse-to-fine correspondence search [8]. Research into better edge and contour detection was also active, including introdution of dynamically evolving contour trackers such as snakes [9] [10].

[1] More about Robot Operating System (ROS) can be found at [7]

In the past decade, there was an emergence of numerous feature-based techniques for object recognition combined with learning, for example by representing an object as a collection of parts arranged in a deformable configuration [11]. Feature-based techniques also dominate other recognition tasks, such as scene recognition [12]. Another trend, which now dominates a lot of the ongoing visual recognition research is the application of sophisticated machine learning techniques to computer vision problems [13]. This trend coincides with the increased availability of large quantities of partially labeled data on the Internet, which makes it possible to learn object categories without the use of careful human supervision [10].

In recent years, convolutional neural networks have gained a significant popularity in the development of artificial inteligence and its applications [14]. They are a useful tool for developing commercial computer vision applications to recognize image objects, to identify human faces in picture, road signs recognition for self-driving cars, and other visual tasks [15].

1.2 Thesis overview

In this thesis, first, various approaches to visual object recognition problem are described, including neural networks. Subsequently the YOLOv2 detector that is used for the onboard detection and other detectors that have similar accuracy or speed of the detection are described. In the second part of this thesis, modifications to the YOLOv2 detector and various experiments performed to test its functionality are presented.

Part I

Visual object recognition methods

Chapter 2

Approaches to visual object recognition

2.1 Template matching

Template matching is a very straightforward process. This method is matching stored template images with given image to determine objects in the input image. This can be performed both on color and greyscale images [16].

This technique can either be based on pixel to pixel mathing or feature based matching. In feature based approach, features of template images are compared to fragments of the given input image [16].

2.2 Color based

Another simple and efficient object detection technique is to represent and match images based on their color histograms. However, when the illumination circumstances are not the same as in template images, the object recognition accuracy degrades significantly. There are also many color spaces that can be used. Commonly used well-known color spaces include: RGB (red-green-blue), CMY (cyan-magenta-yellow); (normalized color) rgb. We can also use intensity I, hue H and saturation S. In table 2.1, we can see differences in invariance of these color models to different conditions of the image [17].

	viewing direction	surface orientation	highlights	illumination direction	illumination intensity
I	-	-	-	-	-
RGB	-	-	-	-	-
rgb	+	+	-	+	+
S	+	+	-	+	+
H	+	+	+	+	+

Table 2.1: Overview of the various color models and their invariance to various imaging conditions. (+ denotes invariant and − denotes sensitivity of the color model to the imaging condition.)

2.3 Feature-based methods

Feature-based methods compare features extracted from an input image with features extracted from template objects [18]. Popular feature extractors include, for example, Speeded Up Robust Features extractor (SURF) [19], which has been used for example for face detection [20], or neural network feature extractors such as ResNet 101 or Darknet-53, which are used by some of the state-of-the-art neural network detectors [21] [22].

2.4 Neural networks for visual object recognition

2.4.1 Multilayer feedforward neural network

This type of neural network is one of the simplest types used for object recognition. This network has one input layer, followed by n hidden layers and one output layer. All neurons of layer *i* are connected to each neuron in layer *i-1*. Each of these connections has assigned weight, that defines how sensitive the neuron is to the output values of the neurons in the previous layer. Each neuron, excluding neurons in input layer, also have a bias number, that strenghtens or weakens the activity of neuron. [23]

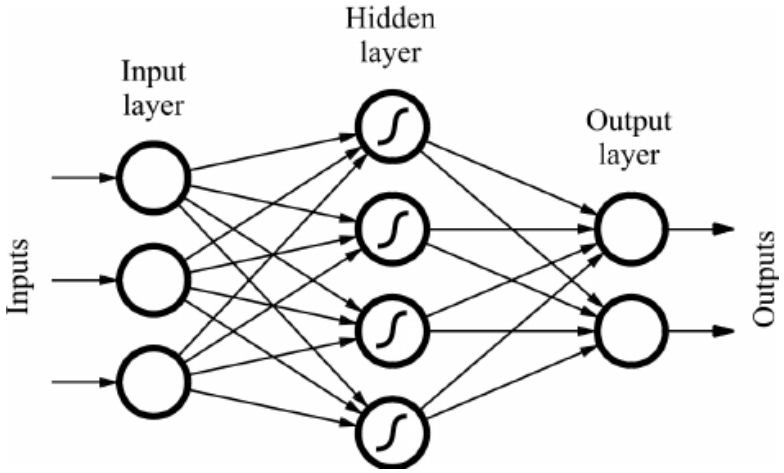


Figure 2.1: An example of a simple feedforward neural network [5]

When used in image recognition, the input to the network would be the pixel value for each pixel in the image, so the input layer would have one neuron for each pixel in the input image. The neurons in the output layer can, for example, hold the probability values for each object class. The object with the highest output value is then chosen as the result of detection. To get correct results for each image neural network has to be trained [24].

Training the network means minimizing the cost function of this network. One of the methods used for minimizing this value is backpropagation. This method modifies the weigths of the connections between layers *i* and *i-1* and the biases of the neurons in layer *i-1* based on the errors in the value of neurons in the layer *i* [23]. This means, that error propagates backwards through network.

2.4.2 Convolutional neural network

Convolutional neural network (CNN) is a special type of multilayer feed-forward neural network [14]. Thse are composed of multiple stages. The input and output of each stage is called the feature map [25]. There are two main modules of a CNN ; the feature learning module and the classification module [14]. The feature learning module can contain multiple 3-layer stages, composed of a filter bank layer, a non-linearity layer and a feature pooling layer [25].

Filter bank layer extracts visual features from an input image. The con-

volution $conv(\bullet)$ is a dot product of the input image, I and the convolution kernel, K. the output is a convolved feature map

$$f_c = (I \otimes K)(i, j), \tag{2.1}$$

where $\otimes$ denotes a two-dimensional discrete convolution operator. The convolution kernel slides over the input image to produce a convolved feature map as output [14].

First the convolutional layer detects low level features from an image. The output from a convolutional layer is called the activation map. The next convolutional layer applies its filters on the activation map from the current layer. This results in the detection of higher level features that are an abstraction of several lower level features found by the previous layer.

Non-linearity layer traditionally consits of a sigmoid function, such as the hyperbolic tangent. However, a more useful function for image recognition is the rectified sigmoid $R_{abs} : |g_i \tanh()|$, where g_i is a trainable gain parameter [25].

The feature pooling layer subsamples the feature map to reduce its spatial dimensionality, thus producing a more compact feature representation [14].

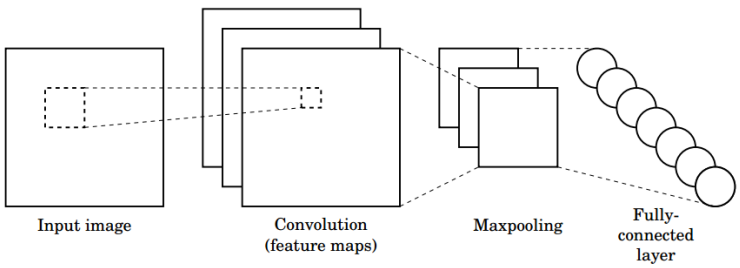


Figure 2.2: Structure of a CNN [6]

Convolutional neural networks are one of the best performing in visual recognition tasks. Recently, they became an important technique for computer vision researchers in design of more effective and sophisticated visual recognition applications [25].

Chapter 3

State-of-the-art CNN detectors

3.1 Faster R-CNN

Faster region-based convolutional neural network (Faster R-CNN) is one of the most accurate state-of-the-art detectors available today [22]. This algorithm uses a region proposal algorithm to formulate a hypotheses about the object locations. These algorithms are usually slow and can not be used in on the fly detection tasks. Faster R-CNN upgrades on its previous version, called Fast R-CNN [26] by combining it with Region Proposal Networks, which are implemented on a Graphics Processing Unit (GPU) and share convolutional layers with the detector while other region proposal methods are implemented on the CPU [22].

3.2 SSD - Single shot multibox detector

Single shot multibox detector (SSD) uses similar approach to YOLO. It does not use bounding box proposal methods, which significantly improves its speed. Instead, the algorithm uses small convolutional filter to predict object cathegories and offsets in bounding box locations, using separate filters for different aspect ratio detections, and applies these filters to multiple feature maps from the subsequent layers of the network in order to perform detection on multiple scales. With these modifications, high accuracy can be achieved

using relatively low resolution input, further increasing the speed of the detector. The SSD is faster and has better accuracy than the original YOLO (66.4% mAP at 21 FPS for YOLO and 74.3% mAP at 59 FPS for SSD in the Pascal VOC2007 test [27]) [28]

3.3 YOLO - You Only Look Once

YOLO detection system is a visual detector, that can recognize multiple classes of objects in an image. Unlike other, similiar detectors, YOLO uses only one convolutional neural network to predict all bounding boxes and probabilities for each object class. All these predictions are simultaneous, so YOLO processes given image in only one pass. Thanks to this ability, YOLO is fast in object detection and can be used in real time detection. [29]

YOLO also reasons globally about an image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time and therefore it implicitly encodes contextual information about classes as well as their appearance [29].

The system operates as follows. Before the detection, the input image is scaled to 448 x 448 pixels. It is then divided into an S x S grid. The cell that includes the center of an object is responsible for detection of that object. Each cell then predicts n bounding boxes and the confidence score for each bounding box. These confidence scores reflect the confidence of the model, that the box contains an object as well as the accuracy of this box. [29]

YOLO detection network has 24 convolutional layers, followed by 2 fully connected layers. Fast YOLO has only 9 convolutional layers instead of 24. This is the only difference between these two versions. [29]

A newer version, YOLOv2, has many advantages compared to the older version. One advantage is the ability to modify the resolution of the input image and thus tune between accuracy and speed of the detector [30]. The structure of the network has also changed. YOLOv2 has 19 convolutional layers and 5 maxpool layers [30].

At the time of writing of this thesis the third version of You Only Look Once (YOLO) detector has been published. YOLOv3 uses a new network for feature extraction. This network has 53 convolutional layers, because of

which it is called Darknet-53. The structure of the network can be seen in [21]. This detector has similar accuracy as SSD, but it is three times faster [21]

For this thesis, the YOLOv2 is used for its precision, speed, as well as for the ability to be easily optimized for the specific task.

Part II

Implementation of the YOLOv2 algorithm and experiments

Chapter 4

Implementation of the YOLOv2 algorithm

For this project, an implementation of YOLOv2 based on the darknet neural network framework written in C++, CUDA and OpenCL was used, together with the Robot Operating System (ROS) framework for implementation on a UAV. The YOLOv2 yolo-voc.cfg configuration file was used, but with slight modifications. Under the net section, the width and the height of an input image was changed to be 640x360 pixels, which is the quarter of the resolution of the camera used in experiments (1280x720 pixels). This way, the aspect ratio of the image is preserved. Under the region section, the number of classes was changed to 1, because the detector will be used only for detecting other UAVs. Finally, the number of filters under the last convolutional section was changed to 30.

4.1 Network training

A YOLOv2 image label is in the following format:

$$[c][x][y][w][h], \tag{4.1}$$

where c is the class number, x is the position of the center of an object along the horizontal axis, y is the position of the center of an object along the vertical axis, w is the width of the object and h is the height of the object all expressed in pixels. For training of the network, a set of 2500 manually labeled images was used. These images are all from the same recording, so they have insufficient variation required for proper training of the network.

The ratio between images used for training and images used for testing was set to 80/20. With this setup, the network was trained for 4000 iterations. The weights that were obtained this way, were then used in the experiments.

4.2 Separating detections and filtering out false positives

Due to the imperfection of the detector, not all bounding boxes correspond to different UAVs. Some of these bounding boxes can belong to the same UAV, falsely detected multiple times, while others can be random erroneous detections. Some of these false positives are filtered out by keeping the last known position for each UAV. At the start of the program, the maximum number of UAVs expected to be detected by the network is given to the program. When a new bounding box is produced by the network, it gets assigned to the closest UAV from the previous frames. If there is no such UAV, the bounding box is assigned to a newly detected UAV. If the maximum number of detections is reached, any new bounding boxes that are not close enough to any UAVs are ignored until one of the previously known UAVs is lost for sufficient time to be cleared. This duration was initially set to 1 second, but due to low performance onboard of a UAV, it was increased to 4 seconds.

This way, multiple detections of the same UAVs are all assigned to this UAV and most of the false positives are filtered out.

The drawback of this solution is that false positives detected in multiple frames can be detected as UAVs, and real UAVs can get filtered out instead.

4.3 Calculating the real-world position of a detected UAV

For calculating the direction and distance of a detected UAV, the pinhole camera model was used. In this model, a scene view is formed by projecting 3D points onto the image plane using a perspective transformation. In OpenCV framework, cameras are described by a camera matrix and distortion

coeficients. Camera matrix is in the following format:

$$C = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{4.2}$$

where f_x and f_y are camera focal lengths expressed in pixels for x and y axis and c_x and c_y are the optical centers expressed in pixels. Distortion coeficients are used to correct the distorition of the camera lens. All these parameters are obtained by camera calibration process.

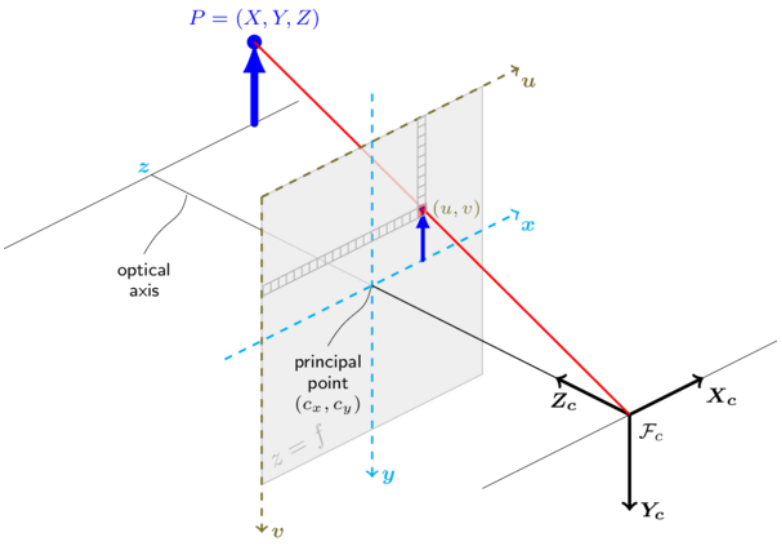


Figure 4.1: Pinhole camera model [31]

4.3.1 Camera calibration

A ROS package called Camera Calibrator was used for camera calibration. The calibration is done by positioning black-white chessboard pattern in front of the camera at various positions. We used 8x6 chessboard with 52 mm squares.

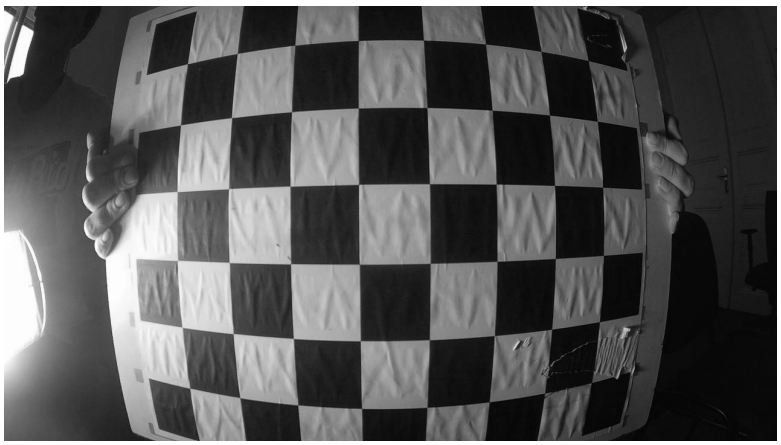


Figure 4.2: Example image from camera calibration process. The distortion of the lens is clearly seen

Figure 4.3: Example image from camera calibration when holding the chessboard at the left side. The width of the board is almost the same as in front of the camera, even though after undistortion it should appear smaller due to perspective projection

4.3.2 Position of a detected UAV relative to the camera

Real world lenses usually have some distortion, mostly radial distortion and slight tangential distortion. In order to calculate the direction of a detected UAV correctly, the detected point from an image must first be undistorted. The OpenCV function `undistortPoints` was used to undistort the point in the center of the bounding box and thus to move it to the position it would have in a pinhole-type camera image. This point was then used to calculate the direction of the detected UAV with respect to camera. The projection of the real world object in an image can be calculated using the camera matrix

as

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = C \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{4.3}$$

where x, y are weighted position of an object in the image, X, Y, Z are coordinates of an object with origin of the coordinate system in the camera with X axis heading to the right, Y axis heading down, and Z axis heading out of the camera, w = Z and C is the camera matrix. The direction vector pointing towards the object from an image position can be calculated as

$$\vec{v'} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = C^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{4.4}$$

This vector can then be normalized to obtain a direction vector with a size of 1

$$\vec{v} = \frac{\vec{v'}}{|\vec{v'}|}. \tag{4.5}$$

In order to calculate the distance of a detected object from the camera, triangle similarity was used. The ratio between real object width and its distance from the camera is the same as the ratio between its width in the image and the focal length.

$$\frac{L}{d} = \frac{l}{f}, \tag{4.6}$$

where L is the real width of the object in meters, d is the distance of the object in meters, l is the width of the object image in pixels and f is the camera focal length in pixels. The distance of the object is then

$$d = \frac{fL}{l}. \tag{4.7}$$

Height of the object could also be used, but it is more dependent on the orientation of the detected UAV and on the relative position of the detected UAV with respect to the camera. Because our UAVs have relatively round shape, their orientation has minimal influence on the width of their bounding boxes.

Once the direction vector towards the detected object and its distance from the camera are obtained, the position of the detected object relative to the camera can then be calculated as

$$\vec{p} = d\vec{v}, \tag{4.8}$$

where $\vec{p}$ is the position of the detected object with respect to the camera, d is the distance of the detected object and $\vec{v}$ is the normalized direction vector.

This vector is then rotated so that X axis is heading out of the camera, Z axis is heading up and Y axis is heading left using rotation matrices

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \\ R_z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.9}$$

to match the coordinate system of UAV with the camera. The position of the detected object in the new coordinate system is then

$$\vec{p'} = R_z R_x \vec{p}. \tag{4.10}$$

4.3.3 Transforming the object position to coordinate system of the UAV

Some of the used UAVs have the camera rotated in yaw with respect to the UAV frame. For those cases, this system is further rotated around Z axis using the transformation matrix

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.11}$$

where α is the camera yaw angle. The position of the detected UAV relative to observer UAV with the camera is then

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = R_z(\alpha) \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}, \tag{4.12}$$

where x, y, z are the coordinates of the detected object relative to the UAV with the camera and x', y', z' are the coordinates of the detected object in the camera coordinate system.

Chapter 5

Experiments

5.1 Performance of YOLOv2

The original YOLOv2 is written on the darknet neural network framework, written in CUDA [1]. This implementation runs at 18 - 20 fps on PC with Intel Core i7 4770K @ 4GHz, Nvidia GeForce GTX1070 and 16 GB of RAM. Because our UAVs do not have CUDA capable GPU, this implementation could not be used for experiments, so OpenCL port of the darknet framework was used instead. In this version YOLO runs at 14 - 16 FPS on the same PC and at 1 FPS on Intel Iris GPU onboard of our UAV. The expected delay between the input to the network and the detected bounding boxes was 1 s, however due to implementation issues the delay at the time of the experiments was approximately 7 s. This was fixed afterwards.

When testing YOLOv2 on desktop PC, the detector managed to detect all UAVs in the image in majority of the cases. The cases where the detector failed to successfuly detect all UAVs were usually when the UAV was flying in front of the sun, or was hidden in the background, as can be seen in figures 5.2 and 5.3.

In the gazebo simulator, the detector managed to perfectly detect and track the UAV.

[1] More about darknet can be found at `https://pjreddie.com/darknet/`

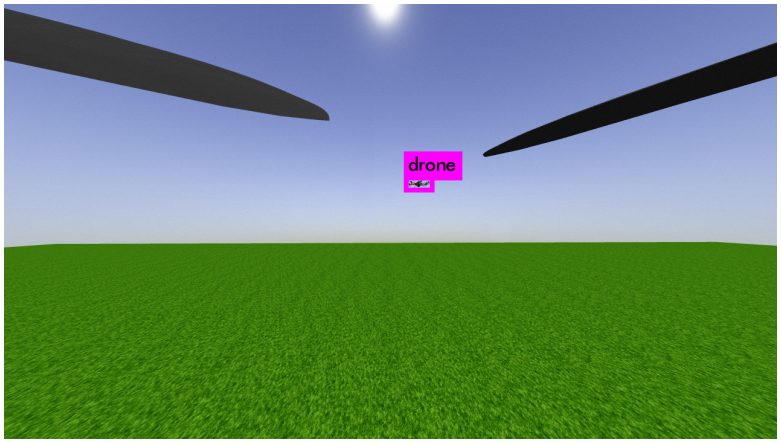


Figure 5.1: Detected UAV in the gazebo simulator

Figure 5.2: The detector does not see the UAV on the right

Figure 5.3: The detector can not separate the UAV from the background

Figure 5.4: The detector identifies this UAV after it was flying in front of the sun.

When testing YOLOv2 onboard of the UAV, the accuracy of the detector decreased compared to tests on PC, due to heavy performance load on the UAV. Thanks to the filter described in the previous section, the detector can still be used for tasks that do not require the production of precise bounding boxes around UAVs.

Figure 5.5: The detector detected both UAVs in the image. The UAV on the left is more than 20 meters away

Figure 5.6: False positives in this image are filtered out.

When the UAV was moving in the image, the onboard detector sometimes created multiple bounding boxes around the UAV. However, when running the detector on the desktop PC, this does not happen.

Figure 5.7: Multiple detections of the same UAV. These should also be filtered out by the described filter

5.2 Leader-follower experiment

To test the functionality of the detector, a simple leader-follower experiment was performed. In this experiment, one UAV follows a pre-set trajectory and another UAV is set to follow the first. Because the relative position estimation, described in section 4.3, was not implemented at the time of the experiment, the Y axis was the same throughout the whole experiment.

The algorithm used for following the leader was a simplified version of the algorithm described in [32], implemented in parallel by other Multi Robot Systems team member at Czech Technical University. The method uses the position of the leader in the image to compute the correct speed and direction of the follower UAV. The thrust gain $m(k)$ for the follower UAV in the time step k is

$$m(k) = 1 - e^{-\frac{1}{2}\widetilde{x}_t(k)^T(\sum_m)^{-1}\widetilde{x}_t(k))}, \tag{5.1}$$

where $\widetilde{x}_t(k) = [\frac{x_t(k)}{y_t(k)}]$ is the position of the leader in the camera image of the follower and $\sum_m \in \mathbb{R}^{2x2}$ is a covariance matrix. The thrust $m(k)$ will be maximal when the leader is on the border of the camera view and zero, when it is in the center of the image [32].

The thrust direction is then obtained as

$$\begin{aligned} v_X(k) &= sgn(\widetilde{x}_t(k))m(k) \\ v_Z(k) &= sgn(\widetilde{x}_t(k))m(k), \end{aligned} \tag{5.2}$$

where $v_X(k)$ is the velocity in the X axis and $v_Z(k)$ is the velocity in the Z axis. Furthermore we need to calculate the desired setpoint that is given to the follower UAV. This setpoint is calculated as

$$\begin{aligned} X(k) &= X(k - \Delta k) + v_X(k)\Delta k \\ Z(k) &= Z(k - \Delta k) + v_Z(k)\Delta k, \end{aligned} \tag{5.3}$$

where $X(k - \Delta k)$ and $Y(k - \Delta k)$ is the position of the follower UAV in the previous time step and Δk is the sampling period. By increasing the sampling period, we can increase the speed of the algorithm, but decrease its robustnes.

In all the experiments, the ground truth position was obtained from Real-time kinematic (RTK) and rangefinder.

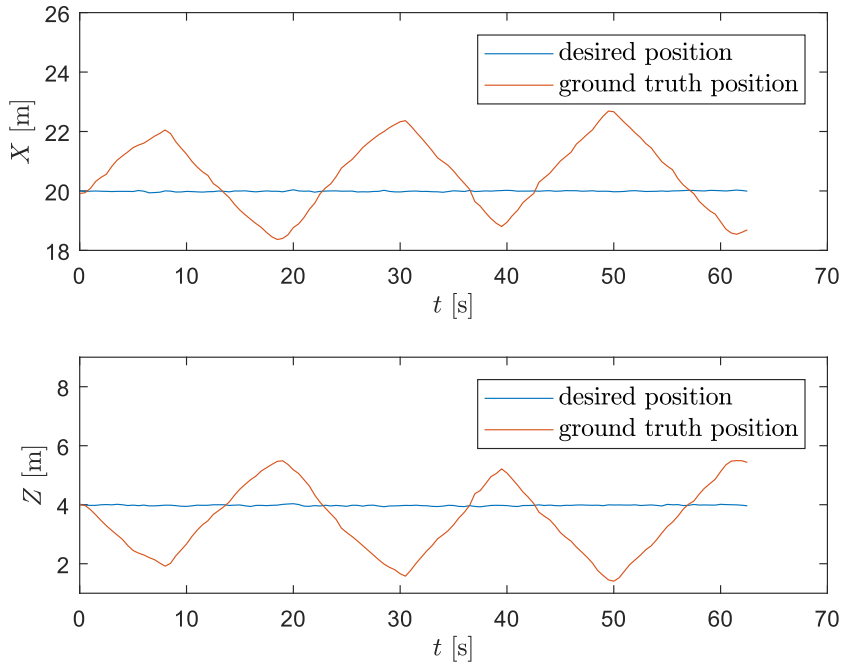


Figure 5.8: The desired and ground truth position of the follower UAV in X and Z axis

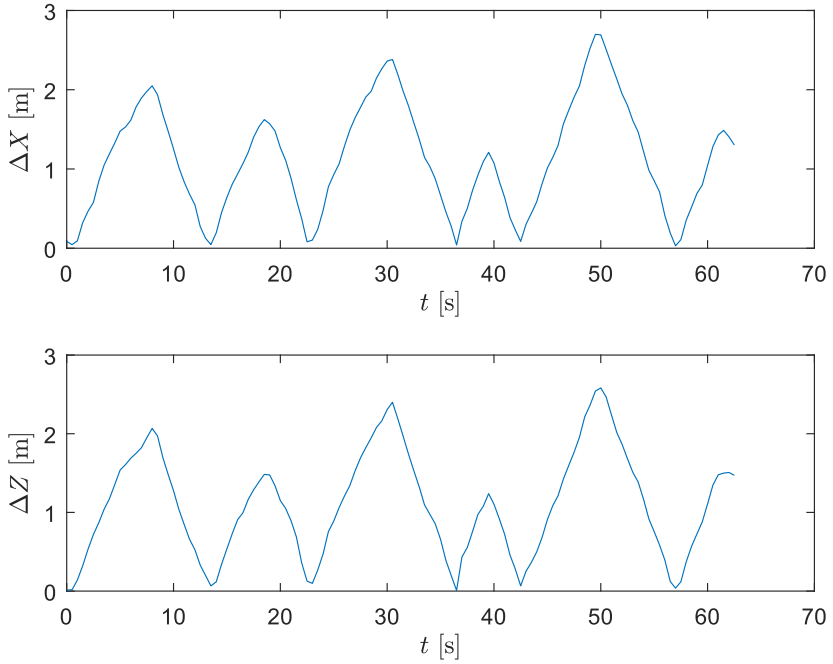


Figure 5.9: The deviation from ground truth position of the follower UAV in X and Z axis

In the first experiment, the follower oscillated around the target position, because of the 7s delay of the detector. For the second experiment, the sampling period of the algorithm was decreased to partially compensate for this delay.

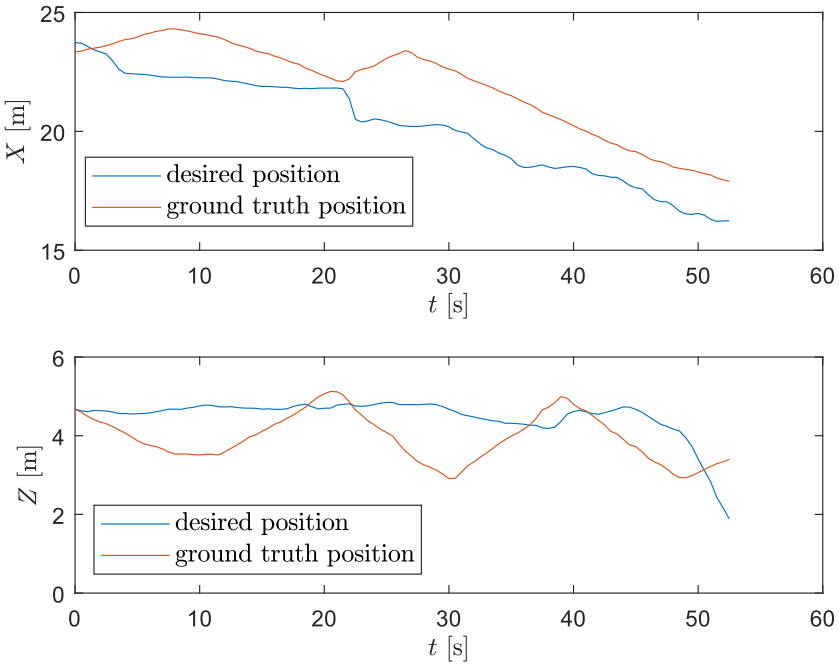


Figure 5.10: The desired and ground truth position of the follower UAV in X and Z axis

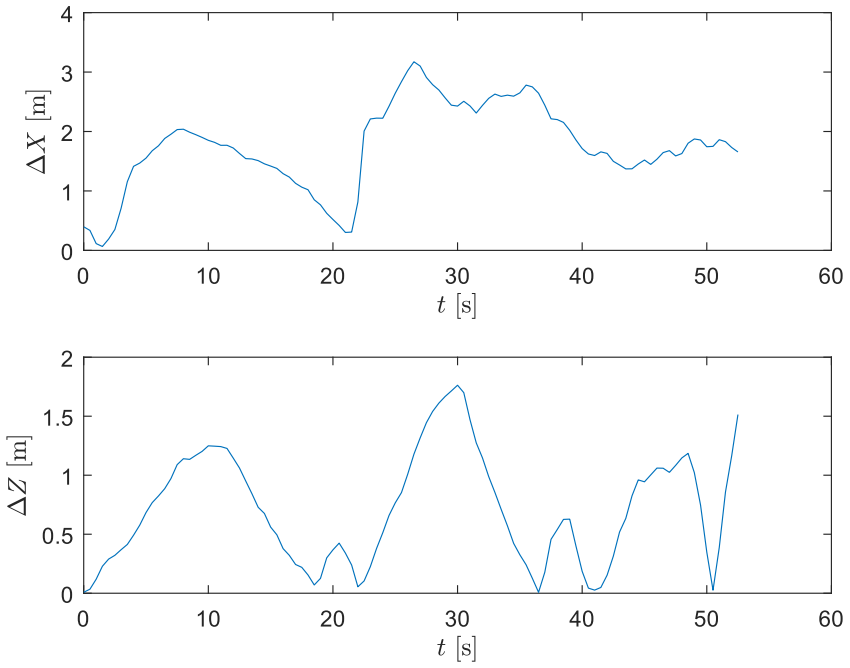


Figure 5.11: The deviation from ground truth position of the follower UAV in X and Z axis

In this experiment, the deviation caused by the delay is still noticable, as seen in 5.10 and 5.11. The follower UAV still manages to track the leader. The video from the whole experiment can be found at `https://www.youtube.com/watch?v=XYZEmq6qC78`

5.3 Precision of the relative position estimation

Because at the time of the experiments this functionality was not yet implemented, the relative position estimation was tested on the desktop PC on video recorded during the leader-follower experiment. This test was performed by comparing the absolute position of the leader UAV obtained from differential GPS data transformed to the coordinate system of the follower UAV and the relative position of the leader UAV estimated from the bounding box produced by the detector.

In order to transform the absolute position of the leader UAV to the coordinate system of the follower we first need to define the transformation

matrices between the global coordinate system and the coordinate system of the follower. These matrices are

$$
\begin{aligned}
R_z(\alpha) &= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
R_y(\beta) &= \begin{pmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
R_x(\gamma) &= \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
T(x,y,z) &= \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix},
\end{aligned}
\tag{5.4}
$$

The relative position of the detected object can then be calculated as

$$
\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = (T(O_x, O_y, O_z) R_z(\phi) R_y(\psi) R_x(\theta))^{-1} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \tag{5.5}
$$

where x, y, z are the absolute coordinates of the leader, O_x, O_y, O_z is the position of the follower UAV and x', y', z' are the coordinates of the leader UAV relative to the follower, ϕ is the yaw of the follower, ψ is pitch and θ is its roll.

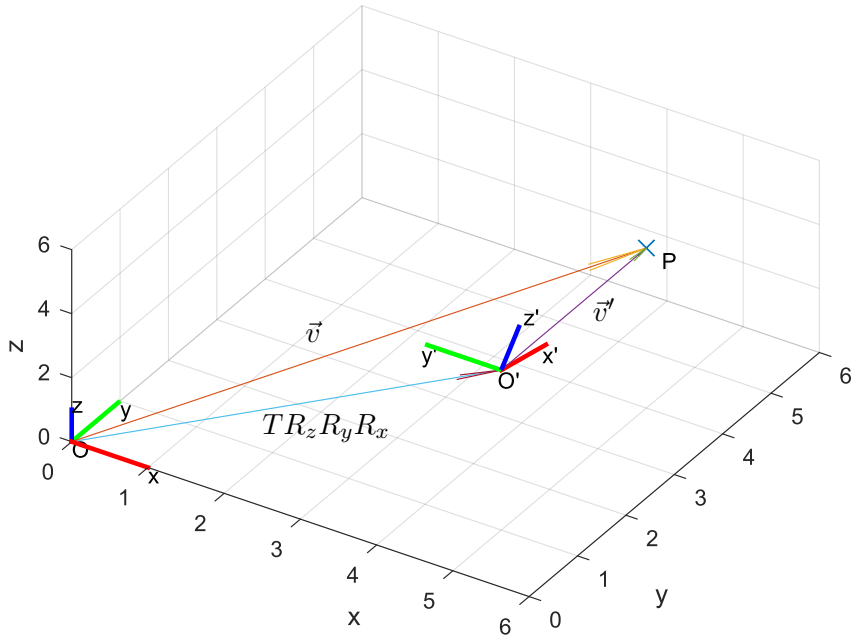


Figure 5.12: Transformation from the global coordinate system (O) to the UAV frame (O')

5.3.1 Experimental results

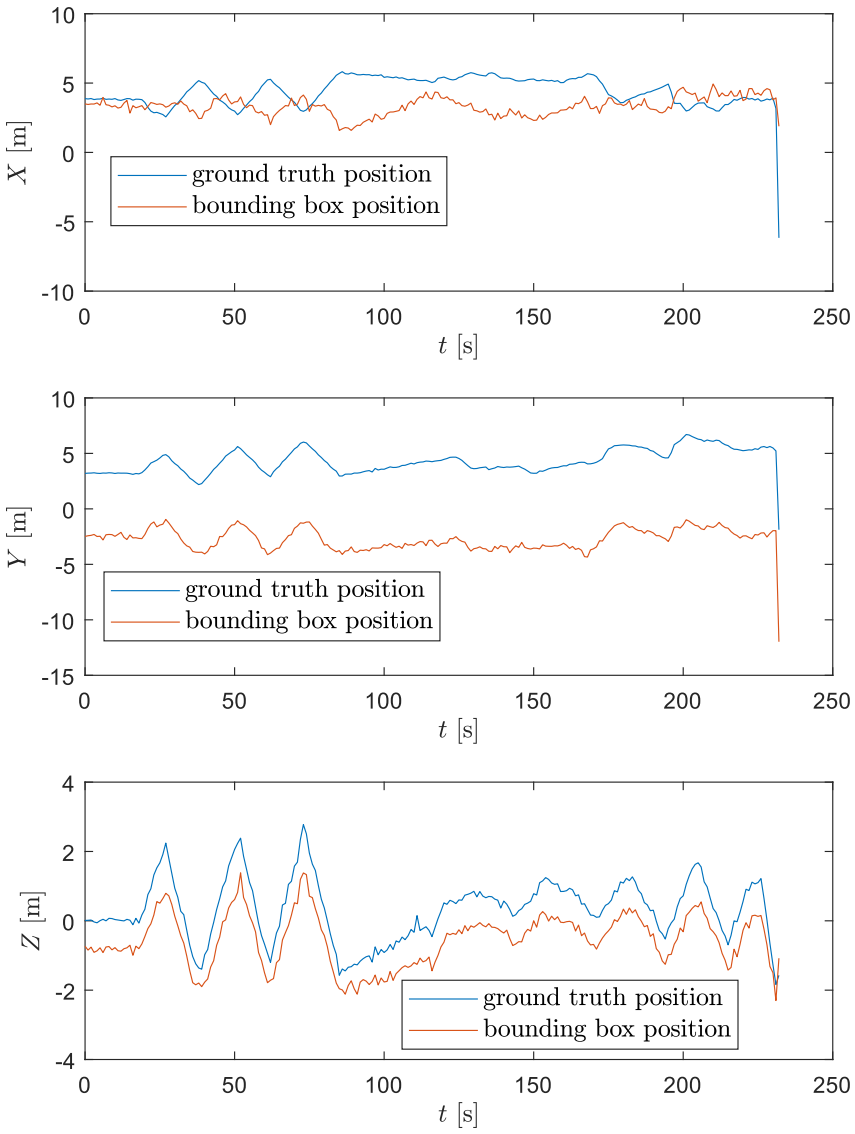


Figure 5.13: Comparison of the relative UAV position estimated from the bounding boxes received from the detector and the relative UAV position calculated from the ground truth

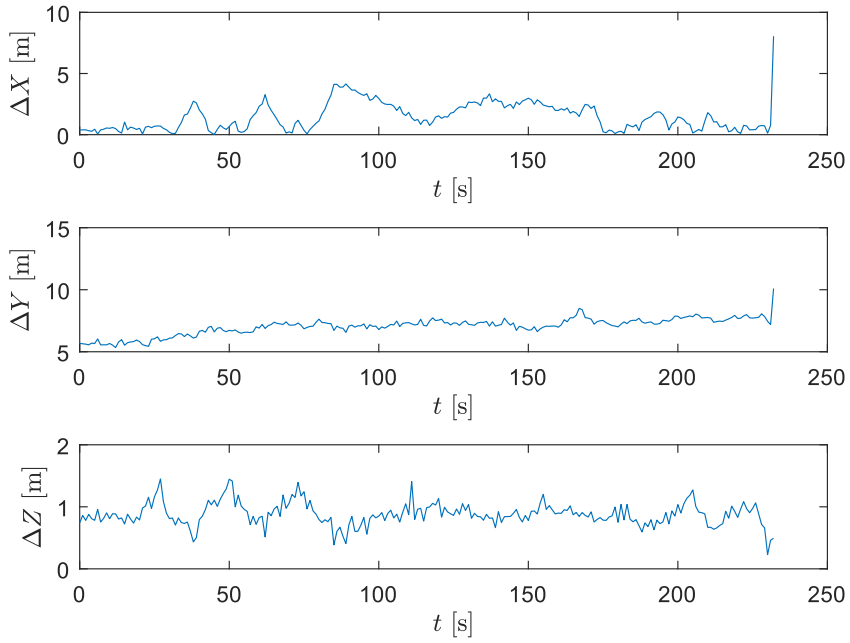


Figure 5.14: The deviation of the relative UAV position estimated from the bounding boxes received from the detector with respect to UAV position calculated from the ground truth

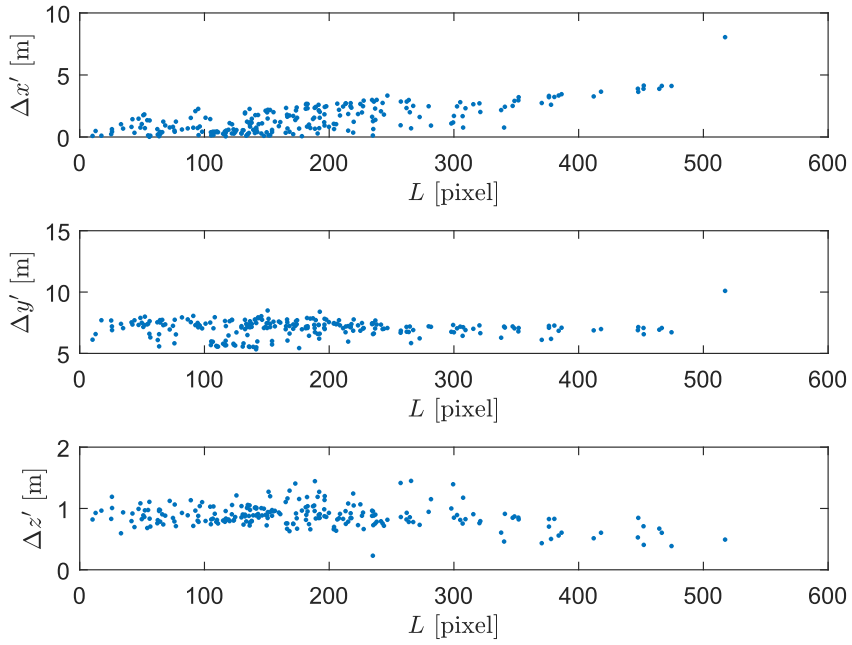


Figure 5.15: The deviation of the relative UAV position estimated from the bounding boxes received from the detector with respect to UAV position calculated from the ground truth based on the distance from image center in pixels (L)

As can be seen in figures 5.13 and 5.14, the position estimated from the bounding box obtained from the detector follows similar trajectory to the position from ground truth in y' and z' axis, but with a constant offset. In x' axis the deviation is smaller. This deviation can be due to inaccurate bounding boxes produced by the detector, error in camera calibration, or inaccurate angle and position readings of UAVs. Because the YOLO detector produces less accurate bounding boxes on the onboard GPU, this inaccuracy would likely increase in such a situation.

As can be seen in figure 5.15, the deviation does not increase with the distance of the detected UAV from the center of the camera image in y' and z' axis. For the x' axis, more data is needed.

Chapter 6

Conclusion

In this thesis we have described various algorithms of visual object detection including some of the state-of-the-art CNN detectors that could be usable for onboard detection of unmanned aircraft. From these detectors, the YOLOv2 system, which proved to be fast, precise and easy to modify was chosen for implementation. The current onboard version of the detector is able to process images at 1 FPS. The bounding boxes are produced 0.9 s after receiving the image.

We have explained the modifications done to the YOLO system for its use in UAV formation control. One of these modifications is filteration of false positives and assigning each bounding box to the correct UAV using the position of each detected UAV in previous frames. Another addition is the estimation of the relative position of the detected UAV with respect to the UAV with the camera.

The detector was tested in leader-follower experiment, together with simplified formation control algorithm, described in [32], implemented in parallel by other MRS team member as part of his Bachelor's Thesis. Due to 7 s delay of the detector, the follower could not center in on the leader, but managed to follow it.

The detector is usable for formation control that does not depend on precise estimation of the relative position between the UAVs, but with the current available dataset the detector is functional only under very specific conditions and even then numerous false positives and inprecise bounding boxes are produced. These false positives are mostly filtered out by the filter described

in 4.2.The UAV that runs the detector also heavily from the computational load. This could be solved by addition of a dedicated GPU.

Appendices

Appendix A

Bibliography

[1] X. Wang, V. Yadav, and S. N. Balakrishnan, "Cooperative uav formation flying with obstacle/collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, pp. 672–679, July 2007.

[2] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small uavs," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 3530–3535 vol. 5, June 2005.

[3] R. W. Beard and T. W. McLain, "Multiple uav cooperative search under collision avoidance and limited range communication constraints," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 1, pp. 25–30 Vol.1, Dec 2003.

[4] S. J. Comstock, "Development of a low-latency, high data rate, differential gps relative positioning system for uav formation flight control," tech. rep., AIR FORCE INST OF TECHNOLOGY WRIGHT-PATTERSON AFB OH GRADUATE SCHOOL OF ENGINEERING AND MANAGEMENT, 2006.

[5] R. Quiza and J. Davim, "Computational methods and optimization," 01 2011.

[6] "Introduction to convolutional neural networks." `http://www.vaetas.cz/posts/intro-convolutional-neural-networks/`. Accessed: 2018-05-24.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[8] A. Rosenfeld, "Quadtrees and pyramids for pattern recognition and image processing," vol. 2, pp. 802–811, 01 1980.

[9] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International journal of computer vision*, vol. 1, no. 4, pp. 321–331, 1988.

[10] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[11] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International journal of computer vision*, vol. 61, no. 1, pp. 55–79, 2005.

[12] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International journal of computer vision*, vol. 73, no. 2, pp. 213–238, 2007.

[13] I. Pitas, V. Calhoun, and K. Diamantaras, "Guest editorial: Special issue on machine learning for signal processing," *Journal of Signal Processing Systems*, vol. 61, pp. 1–2, Oct 2010.

[14] M. Y. W. Teow, "Understanding convolutional neural networks using a minimal model for handwritten digit recognition," in *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, pp. 167–172, Oct 2017.

[15] L. Deng and D. Yu, *Deep Learning:Methods and Applications*. Now Foundations and Trends, 2014.

[16] R. A. Khushboo Khurana, "Techniques for object recognition in images and multi-object detection," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 2, apr 2013.

[17] T. Gevers and A. W. Smeulders, "Color-based object recognition," *Pattern Recognition*, vol. 32, no. 3, pp. 453 – 464, 1999.

[18] J. W. Howarth, H. H. C. Bakker, and R. C. Flemmer, "Feature-based object recognition," in *2009 4th International Conference on Autonomous Robots and Agents*, pp. 375–379, Feb 2009.

[19] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, pp. 404–417, Springer, 2006.

[20] J. Li, T. Wang, and Y. Zhang, "Face detection using surf cascade," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 2183–2190, IEEE, 2011.

[21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[23] C. Stergiou and D. Siganos, "Neural networks." Available at: `http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html`.

[24] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 11 1995.

[25] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 253–256, May 2010.

[26] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[27] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, June 2016.

[30] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, July 2017.

[31] "Camera calibration and 3d reconstruction." `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`. Accessed: 2018-05-24.

[32] F. Poiesi and A. Cavallaro, "Distributed vision-based flying cameras to film a moving target," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2453–2459, Sept 2015.