Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems

Implementation of a neural network for autonomous trail following

Bachelor's Thesis

Yevhenii Kubov

Prague, May 2022

Study programme: Cybernetics and Robotics
Branch of study: TODO

Supervisor: Ing. Matouš Vrba

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor.

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kubov** Jméno: **Yevhenii** Osobní číslo: **492322**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Implementace neuronové sítě pro autonomní sledování cesty

Název bakalářské práce anglicky:

Implementation of a Neural Network for Autonomous Trail Following

Pokyny pro vypracování:

Seznam doporučené literatury:

[1] A. Giusti et al., "A Machine Learning Approach to the Visual Perception of Forest Trails for Mobile Robots," ICRA, 2016.
[2] Seungho Back, Gangik Cho, Jinwoo Oh, Xuan-Toa Tran and Hyondong Oh , "Autonomous UAV Trail Navigation with Obstacle Avoidance Using Deep Neural Networks," JIRS, 2020.
[3] N. Smolyanskiy, A. Kamenev, J. Smith and S. Birchfield, "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," IROS, 2017.
[4] Bruna G. Maciel-Pearson, Patrice Carbonneau, Toby P. Breckon, "Extending Deep Neural Network Trail Navigation for Unmanned Aerial Vehicle Operation Within the Forest Canopy," TAROS, 2018.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Matouš Vrba katedra kybernetiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **06.01.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Matouš Vrba podpis vedoucí(ho) práce	prof. Ing. Tomáš Svoboda, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.

Datum převzetí zadání Podpis studenta

Abstract

The problem of trail following using image from monocular camera, attached to an unmanned drone or ground vehicle is tackled in this thesis. A system solving the task of flying through the forest along the man-made dirt trail is presented. It is accomplished by using a classification deep convolutional neural network for determining which direction is the camera pointed, relative to the trail. It was implemented to run in real-time onboard MRS multi-rotor drone. Performance and robustness was tested in simulations, followed by real-world experiments. The implemented system showed good practical results and can be used as a starting point for more complex navigation and surveillance applications.

Keywords Unmanned Aerial Vehicles, Robotic Perception, Deep Learning, Convolutional Neural Networks

Abstrakt

Tato práce je zaměřená na problematiku sledování lesní cesty pomocí obrázku z monokulární kamery, připevněné na bezpilotní helikoptéře nebo pozemním vozidle. Je představen systém, řešící úlohu navigace podél stezky v lese. To bylo dosaženo s využitím klasifikační hluboké konvoluční neuronové sítě pro určení směru natočení helikoptéry vzhledem k cestě. Systém byl implementován pro běh v reálném čase na bezpilotní helikoptéře MRS. Výkon a robustnost byla otestována v simulaci a následně během experimentů v reálném světe. Implementovaný systém prokázal dobré praktické výsledky a může být použit pro jako výchozí bod pro komplexnější navigační a průzkumné aplikace.

Klíčová slova Bezpilotní Prostředky, Robotické Vnímání, Hluboké Učení, Konvoluční Neuronové Sítě

Abbreviations

Contents

Chapter 1

Introduction

1.1 Unmanned Aerial Vehicles

Flying drones, also called UAVs (Unmanned Aerial Vehicles) have been used since the first half of the 20th century [9]. They were still controlled by a human operator, but allowed for dangerous tasks to be performed remotely, like aerial target practices. Originally designed for military purposes, they evolved into a vast industry. Since that time, electronic hardware has become much more compact, power-efficient, cheap and advanced. Also, aircraft designs have strongly evolved after decades of research, battery power improved, software got advanced as never before. Nowadays, all those factors allow for a wide usage of compact and relatively inexpensive drones in many industries and research.

Drones can be equipped with different payloads and sensors, depending on their task (Fig. 1.1). For example, thermal cameras are used for border control, detecting animals in wildlife, or forest fires. LIDARs can be used for 3D mapping of the area, for navigation in obstructed environment. Digital cameras are good for crops analysis, area mapping, surveillance, monitoring of different constructions or power lines.

Figure 1.1: MRS MAV equipped with firefighting and thermal imaging modules.

UAVs are also important for SAR (Search and Rescue) missions. Benefits of using them in such operations is that drones are usually portable, have small deployment time and generally require only minimal qualification to operate them, thanks to flight computers with advanced software. They can be used instead of human teams, reducing the risk for their life or health in dangerous environments. UAVs may be also cheaper than alternatives, especially manned aircraft, and faster than rescue teams or ground vehicles. They are usually not only less expensive, but also provide data logging from all sensors, including GPS and associated image data from high resolution cameras. This data can be used even for later review. For SAR missions usually two types of drones are used: fixed-wing aircraft and multi-rotor helicopters. First ones provide higher speed, flight time, but are less agile and take longer time to take off and land the vehicle. Multi-rotors have the ability to hover, fly close to the ground, in buildings, caves and other complex terrains. In this thesis, a multi-rotor MAV will be used as an airframe.

1.2 Convolutional neural networks

Neural networks (also called artificial neural networks) are algorithms, whose design was inspired by biological neural networks in human and animal brains. Their structure consists of nodes called neurons, and connections between them. Every connection has its weight. Neurons are grouped into layers, each layer has its unique number of them. Neurons which receive value (stimuli) on input from outside of the network are input neurons. Those who receive values from other neurons, process this data and output the result to other neurons are hidden neurons. Those who output the result outside the network are called output neurons. Mentioned weights of connections between neurons can be adjusted through the learning process. During it, a network learns to identify needed shapes in the input and produce a correct decision according to the task.

Convolutional neural networks have been experiencing tremendous growth during the last 10 years, allowing for many previously unsolved problems to be tackled [1]. They are used in military, healthcare, aerospace, social media, science and other applications. This approach allowed for faster and more accurate analysis of many diseases, even on early stages, using measurements performed on the patient, which are then fed to a neural network. In aerospace and automotive engineering neural networks are often used as component fault detectors and for improved guidance systems. In electronics their capabilities help to expose failures when producing the chips, synthesise voice, compress data and solve many other tasks. In military they help to identify hostile objects or enemies.

There are different types of neural networks. Several examples are presented in this chapter.

1.2.1 Segmentation networks

Among the most popular types are segmentation neural networks. Their goal is to divide an image into multiple segments. In such architecture, each pixel refers to some class or object type. This type is often used for biomedical applications. Good example is U-Net architecture, frequently used in light microscopy [7].

1.2.2 Recurrent networks

This type of networks is used for problems, such as language translation and speech recognition. They are designed to handle sequential data on input (Fig. 1.2).

1.2.3 Generative adversarial networks

GAN networks consist of two neural networks, that contest with each other. Each network's gain is another network's loss. One network called discriminator identifies how much

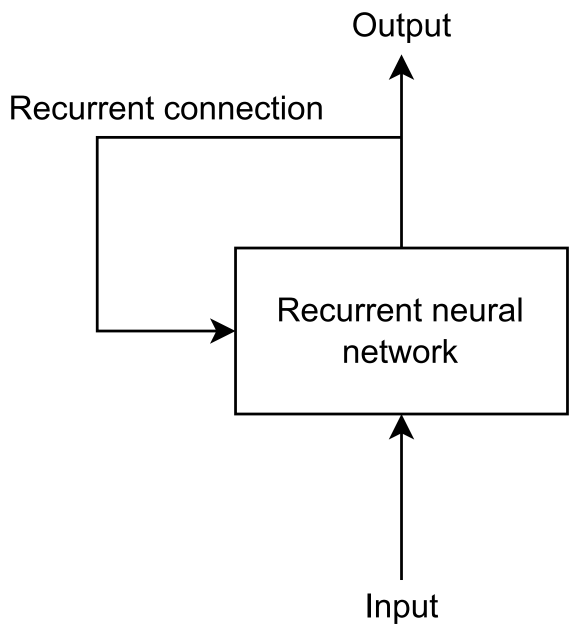

Figure 1.2: Example of a recurrent network architecture.

the input image is "realistic", while other (generator) generates this input and adjusts it to "fool" the discriminator.

1.2.4 Classification networks

One of the most popular are classification neural networks. In such use-case given an input image, a neural network must identify to which class does an image belong. Good example is Alexnet architecture for classification on 1000 classes subset from the ImageNet dataset [11]. Back in 2012 it won the ImageNet large-scale visual recognition challenge. This type of a neural network I will use in this thesis. Architecture is from [6].

1.3 Trail following

In this thesis, the problem of trail following using image from monocular camera onboard a multi-rotor MAV is tackled, including the implementation of a working algorithm, solving the task. Following the man-made forest path is natural for humans, because it is usually the most efficient way to get through such complex terrain. Such policy, in most cases, minimises the travel time and possible injury to a person (Fig. 1.3). The same applies to robots. Human paths are freely passable, unlike random trajectories in a forest, and it is a reason to stay on them.

Trail following is an important task for autonomous navigation of robots. Suggested use-cases are search and rescue missions, efficient navigation through forests and mapping of the area [TODO: REF]. Motivation for this task is a situation when there is no opportunity to communicate with and control the drone manually or when an autonomous mission is highly preferred. The goal is to allow for a quadcopter or unmanned ground vehicle (UGV) to navigate through a forest using computer vision techniques. Having the image from the

Figure 1.3: Random path in a forest is generally challenging to pass.

onboard camera, the vehicle should determine which direction to travel when flying through a forest, utilising the trail. It must strictly follow the human path.

Algorithms solving related problems like lane-following, lane-departure and lane-assist for cars on public roads, were introduced in 1990's [12] and are commonly used in personal vehicles since the early 2000's [3]. But there is a clear distinction between the lane on a road and a forest trail. In the first case, the lane is marked with contrast symbols and lines, making the task solvable by simple segmentation algorithms, based on in-image contrast and colour variance, image saliency [10]. Forest trail images provide smaller amount of distinguishable features (Fig. 1.4) and it may be challenging even for humans to determine the direction of travel [6].

(a) Image of a trail from the dataset [6].

(b) Image of a road taken by myself.

Figure 1.4: Difference between a forest trail on Fig. 1.4a and a road on Fig. 1.4b.

One of the first effectively solved by neural networks, but still topical problems is classification. Neural networks are able to learn and then identify features, corresponding to predefined classes, and combinations of those features [11]. This makes it possible to effectively classify, which class an image belongs to.

Treating the trail following task as a classification problem is a different approach that specifically tackles it. For this approach classes like "Left", "Right" and "Straight" can be introduced [6]. It allows to estimate the current direction of a drone, given the probabilities of these classes and to plan the trajectory of the MAV accordingly.

(a) Class "left".

(b) Class "right".

(c) Class "straight".

Figure 1.5: Example images of different classes

This thesis focuses on implementation of the trail-following algorithm using classification convolutional neural network. The implementation should run online in the Gazebo simulation environment as well as onboard a MAV in a real-world deployment. Therefore, delay of the algorithm must be sufficiently small. The task is to design an algorithm that autonomously provides a MAV with a safe direction and speed of movement. The MAV is equipped with PX4 flight computer, Intel NUC companion computer, and Intel RealSense camera. The environment may contain obstacles, but is assumed that the trail is obstacle-free.

Implemented neural network can be used as an entry point for more sophisticated surveillance and navigation algorithms. For more complex environments a possible enhancement is usage alongside with obstacle avoidance algorithms and lateral correction [2], [4], [5].

Chapter 2

Methodology

TODO: Describe back propagation of the error (+formulas)

TODO: Loss function

TODO: Trajectory generation (2nd section, first one is about system...)

In this thesis an architecture suggested by Giusti, Guzzi, Ciresan, *et al.* will be used. It consists of 4 convolutional layers, each followed by hyperbolic tangent activation and max pooling layer, and then a fully connected layer with 200 hidden neurons. Network processes images from a camera, attached to a vehicle. Input layer is formed by $3 \times 101 \times 101$ neurons. Therefore, input RGB image must be anisotropically resized to a size 101×101 pixels to be fed directly to the network. After going through all the hidden layers and the softmax output layer it produces 3 probabilities of each class, which sum to 1. Based on these probabilities it is possible to determine at which direction is the camera most probably pointed. Given the fact that side cameras are pointed 30° from the centre, interpolation is also possible based on these probabilities.

TODO INSERT... TODO: scheme + details in subsections

2.1 DNN

2.1.1 Convolutions

Convolution is a fundamental mathematical operation used in a wide range of image processing techniques. In the context of Convolutional Neural Networks, convolution of an input matrix I with a so called "kernel" matrix K is applied to obtain an output matrix O. The kernel is typically smaller and is applied to submatrices of I to extract features corresponding to K in different regions of I. The convolutional layer of a CNN typically also contains a bias term w_0. The kernel matrix K and the bias w_0 are parameters that are learned during the training phase using the backpropagation algorithm, described in section 2.2.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ x_{r1} & x_{r2} & x_{r3} & \dots & x_{rn} \end{bmatrix}, \tag{2.1}$$

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}. \tag{2.2}$$

Convolution is performed by "stamping" a kernel onto input data, starting from the upper left angle, and thus creating a linear combination of input array members and kernel weights. Let the input data be (2.1) and convolutional kernel be (2.2). Then, the result of the first application of convolution kernel will be:

$$\begin{aligned} O(1,1) = x_{11} \cdot w_{11} + x_{12} \cdot w_{12} + x_{13} \cdot w_{13} + \\ x_{21} \cdot w_{21} + x_{22} \cdot w_{22} + x_{23} \cdot w_{23} + \\ x_{31} \cdot w_{31} + x_{32} \cdot w_{32} + x_{33} \cdot w_{33} + w_0 \,. \end{aligned} \tag{2.3}$$

And the equation for convolutional kernel stamp starting on i-th row and j-th column of the input image (where the convolution is defined) is:

$$O(i,j) = w_0 + \sum_{k=1}^{m} \sum_{l=1}^{n} I_{i+k-1,j+l-1} \cdot K_{k,l} \,, \tag{2.4}$$

where kernel has m rows and n columns, i runs from 1 to $M-m+1$, j runs from 1 to $N-n+1$ [TODO: REF]. Kernel is K, input image is I. To produce the output array, filter must slide through the whole input image.

2.1.2 Max-pooling

Max-pooling is an operation, applied to some part of the input data array, taking only the maximum value of this area. It is typically used in the form of a rectangular filter, which slides through the whole image. It produces only one output value from each filter-sized input area, thus can be used for downsampling of the image, taking only the most significant values to the output. In this way, the learning process of the neural network is sped up because the amount of learnable weights is decreased. Also, better resistance to distortions and affine transformations is obtained [8].

2.1.3 Softmax

Softmax is widely used in neural network architectures as the last layer. It has same amount of outputs as inputs. Softmax function may have any real values on input, including positive, negative, zero, but its output values are always in the range $[0, 1]$ and they always sum to 1. These properties allow the output to be in form of "probabilities" of each corresponding input value. This layer normalises the output, which is from R^n to probability distribution.

The softmax function is defined as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum\limits_{j=1}^{n} e^{z_j}} \,, \tag{2.5}$$

where z_i are elements of the real input vector, $\sigma(\vec{z})$ is and output, n is the number of outputs.

2.2 Backward propagation of error

Backward propagation is the way to calculate the gradient of the loss function $\frac{\partial J}{\partial \mathbf{w}}$ with respect to the vector of weights $\mathbf{w}$. Calculated gradient has the same length as weights vector.

It means how much each weight affects and contributes to the value of the loss function. This knowledge is used to change the weights in a way that minimises the loss.

First part of backpropagation is a forward pass. Given the input data and the weights, output of the neural network is calculated and compared with ground truth through the loss function. Then, backward pass starts. Partial derivatives are calculated sequentially through each layer, starting from the last one, and after multiplication give the total gradient $\frac{\partial J}{\partial \mathbf{w}}$.

Used in this thesis architecture consists of convolutional layers, max-pooling layers and fully connected layers. Max-pooling chooses one input with the maximum value and feeds it directly to the ouput, other inputs are ignored. It doesn't have learnable parameters affecting the gradient. In convolutional layer backpropagation is the convolution of input feature map with upstream gradient.

Chapter 3

Implementation

TODO: system image/scheme

3.1 System hardware

3.1.1 Pixhawk flight controller

Pixhawk is a low-cost advanced flight computer with open-source hardware. There are different variations of form factors, featuring different amount of input/output ports. Pixhawk is very flexible in terms of attachable peripherals, stable and well-tested. Most essential sensors like accelerometers, gyro, digital compass (magnetometer) and barometer are already part of the main board. MRS vehicles have these boards flashed with open-source PX4 autopilot software. Features like advanced regulators, estimators, interface for controlling the drone and others are already implemented in this software, usually only minor tweaking is needed. Thus an abstraction from the MAV hardware is created, allowing the vehicle to be controlled using high-level commands.

3.1.2 Intel NUC companion computer

NUC is a compact high-performance, yet power-efficient computer, capable of running demanding AI and machine learning software. It is possible to install the whole MRS system on it, including ROS software to command trajectories, speed and other parameters to a flight controller and act as a main high-level computational unit. ROS is run inside Ubuntu operating system on this board, so other software can be used simultaneously. Different peripherals can be connected to a computer via USB. For this thesis a RealSense camera is connected, from which my algorithm (written in Python programming language) is receiving images for processing.

3.1.3 Intel RealSense D435 camera

D435 is a powerful camera capable of taking normal RGB pictures as well as depth images. It has a wide field of view, which is perfect for robotic applications. Also its stereo imagers feature global shutter, which is important in low-light conditions or during fast movements. The camera consists of 4 camera modules: right imager, IR projector, left imager and RGB module. However, for this thesis only the RGB module will be used. Its sensor has a resolution of 2 MP and produces 1920×1080 images at the frame rate of 30 frames per second.

3.2 Used software tools

3.2.1 Python

I have chosen the Python programming language for this task. It is a high-level language with simple syntax, that features high user-readability and supports object-oriented approach. It is dynamically-typed and offers garbage collection feature. Thanks to these properties, it is considered to be one of the easiest programming languages to use. However, algorithms, written purely in Python are usually inefficient compared to the same ones written in low-level languages. But on the other hand, it has a lot of available powerful frameworks, especially for machine learning and neural networks, which are internally implemented in faster low-level languages like C++. So, the speed reduction will not be critical in this case. In this thesis a relatively powerful onboard PC is used so it is not a problem. But in a situation where the fastest possible execution time is needed, another language should be considered.

3.2.2 PyTorch framework

PyTorch is a powerful open-source machine learning framework for Python. It is often used for computer vision tasks, contains many pre-implemented modules and features, which makes it usable in a wide variety of applications and for most neural network types. It is actively developed and maintained. Framework is written in Python, C++ and CUDA languages, but Python is used mainly as an interface for it. Low-level operations are not implemented in Python due to its relative slowness. PyTorch is capable of running on GPU to accelerate tensor computing, however, CUDA-capable Nvidia GPU must be used.

3.3 Dataset

For this task, I have chosen dataset from the authors of DNN architecture [6], but any similar dataset can be used. Requirement is that every image must be labeled to one of the three classes. Used dataset was acquired by a hiker, wearing three head-mounted cameras: one pointing straight ahead, one pointing 30 degrees left, one pointing 30 degrees right. Originally images were labeled corresponding to the expected action: images from the camera pointed to the left are of class "turn right" (TR), from the forward camera are of class "go straight" (GS), from the right camera are of class turn "left" (TL) (see Fig. 1.5).

However, I decided to change the classes definition to be more intuitive: in my code, there will be classes LEFT (for the camera pointed 30 degrees left), STRAIGHT (forward camera), and RIGHT (for the camera pointed 30 degrees right). Dataset is divided in such a way that approximately 75% of it is used for training, and 25% for validation. So, the neural network estimates the current direction relative to the trail, and based on this knowledge, further decision can be made.

3.4 Neural network code

The neural network is trained for 90 epochs, with a batch size of 512, but even larger size should be considered if the GPU has enough memory. Adam optimizer is used, with an initial learning rate of 0.005. Also, a scheduler is set, for reducing the learning rate by 5% on each epoch. The criterion for training is the cross-entropy loss.

Algorithm

After the initialization of the neural network model, optimizer, scheduler, and criterion, the program enters a 90-epoch loop. In each epoch the data loaders are created for training and validation, each receives its part of the data from the pickle file, where the whole prepared dataset is stored. Then starts the training part where the forward pass and the back-propagation are calculated, optimizer step is done. After that validation part begins and the scheduler step is performed. In the end, the weights of the neural network are saved for further usage.

3.5 Program for preparing the dataset

The downloaded dataset was structured into 14 folders (sub-parts), each containing folders “lc” (left camera), “rc” (right camera), “sc”(straight-pointing camera). Inside each one, there is a folder with the name ending “.frames”. Images from it must be moved one level up and the “.frames” folder must be deleted.

Inside the code, a user must set the path to the dataset, select whether preparation of training or validation part is needed, and specify the range of sub-parts to be used (for example, 1-9 for training and 10-14 for validation).

For each selected sub-part program iterates through all images from the dataset, performs transformations so that the images correspond to the required format. In this case, they are resized to 101 by 101 pixels. Then it appends the transformed image to the data array and its label to the label array. After that, these arrays are saved to the python dictionary under fields ‘data’ and ‘label’. In the end, this dictionary is saved to the pickle .pkl file as “trn.pkl” or “val.pkl”, according to the choice of part in the previous paragraph.

Pickle is a tool for serialization of Python objects, which is handy in this case, because the whole prepared, ready-to-use dataset can be transferred as a single file.

3.6 Navigation algorithm

TODO: Describe how do I calculate the yaw rate to follow the trail.

Chapter 4

Simulation

4.1 Software

For simulation of such complex application a dedicated software is needed. The main tool for simulation and further usage in a real drone is Robot Operating System (ROS). It is a framework for robotic applications which offers abstraction from hardware, and even contains already implemented functions for communication between drones, sensors, cameras and other used hardware, both real and virtual. The communication is performed using high-level messages. For every hardware unit a node is created. These nodes can subscribe (receive information) or publish to some topic. Topic represents a virtual pipe, through which the information is transferred. For example, there is a program running onboard drone PC. Program is subscribed to the topic `"/image"`, where the image is obtained from the node `"/camera"` and receives image from it. Then, it processes the image, makes some decision after it and commands the speed using `"/speed"` topic of the node `"/drone"`. [TODO: REFERENCE]

TODO: IMAGES OF ROS NODES AND TOPICS

Another tool used for simulation is Gazebo. It offers real-time graphical visualisation of the ongoing experiment. Very realistic scenarios can be created in this simulator, its engine allows for shaders, different lighting conditions, and even physics simulation. Therefore, a virtual model of the use-case scene and conditions is usually designed, including the drone itself. Such model makes it possible to conveniently test designed software before proceeding to real-world experiments as the cost of a mistake in the real world can be high.

4.2 Simulation setup

For trail following simulation I decided to use the MRS pre-configured setup for one-drone simulation [TODO: REF]. In my case I will run the system inside the singularity container. Default simulation does not include the onboard camera. Thus the `enable_mobius_camera_front` parameter was added to the startup config `session.yml`. I selected the "Baylands" world for simulation, because it has a section of forest path in it.

The code must be also modified to be run in simulation. Only needed change is the topic, from which the images are received. It should be `"/*uav_name*/camera/image_raw"TODO`

To start the simulation I run the script `start.sh` and wait for initialisation of all windows. Then in tmux I can switch to the free window where no process is currently running and there startup my python code.

4.3 Simulation results

TODO:IMAGES

During the experiment in simulation, performance of the implemented algorithm was tested in close to real-world conditions. UAV successfully managed to follow the trail without getting lost. There are some oscillations during the heading correction, but they can be removed by using more complex heading regulation. In fact, part of the algorithm, responsible for heading correction, performs as a proportional regulator, which sets the angular speed of the drone to the expected value. For further improvement of the program, it is possible to implement a PD (proportional and differential) regulator, PID (proportional, integral, and derivative) regulator [TODO: REF], or modify the code to send heading commands instead of angular velocity commands, which will use internal regulator of the flight controller.

Chapter 5

Experiment

TODO: Describe experiment, review results

5.1 Neural network performance test

For evaluation of the neural network performance in real-world forest conditions I decided to take a walk with the on-board computer and the camera, all powered from the battery. Data was outputed in real-time to my laptop and I was able to evaluate the prediction correctness.

TODO: PHOTOS

Testing showed good neural network performance. Accuracy of determining the ”left” and ”right” classes in situation where it is also possible for humans was 100%. There was once a situation though, where the neural network was outputting small probability of ”straight” class, when looking straight. Probabilities of ”left” and ”right” were same, close to 50%. But the issue was clearly dependent on tilt angle of the camera, after tilting it a few degrees down, problem was solved. Possible source of issue could be not only the neural network fail-case, but also lens flare. TODO

5.2 Complete tests on vehicle

Chapter 6

Conclusion

TODO: Conclusion

Chapter 7

References

[1] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[2] S. Back, G. Cho, J. Oh, X.-T. Tran, and H. Oh, "Autonomous uav trail navigation with obstacle avoidance using deep neural networks," *Journal of Intelligent & Robotic Systems*, vol. 100, no. 3, pp. 1195–1211, 2020.

[3] W. Chen, W. Wang, K. Wang, Z. Li, H. Li, and S. Liu, "Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review," *Journal of traffic and transportation engineering (English edition)*, vol. 7, no. 6, pp. 748–774, 2020.

[4] B. G. Maciel-Pearson, P. Carbonneau, and T. P. Breckon, "Extending deep neural network trail navigation for unmanned aerial vehicle operation within the forest canopy," in *Annual Conference Towards Autonomous Robotic Systems*, Springer, 2018, pp. 147–158.

[5] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 4241–4247.

[6] A. Giusti, J. Guzzi, D. Ciresan, F.-L. He, J. P. Rodriguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, and L. Gambardella, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, 2016.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241.

[8] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *International conference on rough sets and knowledge technology*, Springer, 2014, pp. 364–375.

[9] J. F. Keane and S. S. Carr, "A brief history of early unmanned aircraft," *Johns Hopkins APL Technical Digest*, vol. 32, no. 3, pp. 558–571, 2013.

[10] P. Santana, L. Correia, R. Mendonça, N. Alves, and J. Barata, "Tracking natural trails with swarm-based visual saliency," *Journal of Field Robotics*, vol. 30, no. 1, pp. 64–86, 2013.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[12] P. H. Batavia, *Driver-adaptive lane departure warning systems.* Carnegie Mellon University, 1999.

Chapter A

Appendix A