Efficient Detection of Wire Frame Objects on Micro-Air Vehicles

by

P. Duernay

in partial fulfillment of the requirements for the degree of

Master of Science

in Embedded Systems

at the Delft University of Technology,

to be defended publicly on Tuesday January 1, 2013 at 10:00 AM.

Supervisor:	Prof. dr. ir. D. M. J Tax	
Thesis committee:	Prof. dr. C. F. Guido de Croon,	TU Delft
	Dr. E. L. Brown,	TU Delft
	Ir. M. Scott,	Acme Corporation

This thesis is confidential and cannot be made public until December 31, 2013.

TU Delft Delft University of Technology

Preface

Preface...

P. Duernay
Delft, January 2013

Contents

Summary

Glossary

IR Infrared

LIDAR Light Detection And Ranging

EWFO Empty Wire Frame Objects

FoV Field of View

CNN Convolutional Neural Network

GPS Global Positioning System

GPU Graphical Processing Unit

MAV Micro-Air Vehicle

DR Domain Randomization

TO Target Object

DSC Depthwise Separable Convolution

IMU Inertial Measurement Unit

FPV First Person View

IROS International Conference of Intelligent Robots

i.i.d. independently identically distributed

PCA Principal Component Analysis

mAP Mean Average Precision

NED North-East-Down

CAD Computer Aided Design

WRN Wide Residual Network

FCN Fully Convolutional Network

Yolo You only look once

RPN Region Proposal Network

SSD Single Shot Multibox Detector

SVM Support Vector Machine

Chapter 1

Introduction

Micro-Air Vehicles (MAVs) such as a Quadrotor-MAV displayed in Figure 1.1 are an emerging technology that supports society in a wide range of consumer, industrial and safety applications. For example MAVs are used to deliver medicine [45], fight fires [24] or even find survivors in disaster situations [21].

Especially in emergency scenarios the fast and safe flight of MAVs is crucial to deliver help quickly and save human lives. However, due to the complexity of such missions as well as the difficulty to control an MAV in disaster scenarios, often multiple human operators are required in order to ensure safe operation [34]. With humans in the loop a constant connection between the MAV and the operators is required which not only uses energy and requires infrastructure but also significantly increases the reaction time. Enabling MAVs to fly more autonomously could allow less human operators to control more MAVs and thus to improve the support in emergency situations.

A major challenge on the way to the full autonomous flight of MAVs is the accurate estimation of the MAV's state within its environment. The system is highly dynamic so position and orientation can change rapidly. At the same time noise introduced by motor vibrations makes the position estimation with only on-board Inertial Measurement Units (IMUs) too inaccurate [33]. Light Detection And Ranging (LIDAR)-sensors can capture long and wide range 3D information but the sensors are typically heavy and require a significant amount of energy. Infrared (IR) sensors can cover distance information but are often limited in their Field of View (FoV) as well as in their range. External infrastructure like Global Positioning System (GPS) and optical tracking systems can provide accurate measurements but there is no guarantee that such systems are present in real world applications. Cameras on the other hand are cheap, lightweight and can measure long range distance information. This makes them a suitable choice as a sensor for on-board state estimation on light

Figure 1.1: An example of a Quadrotor-MAV-Platform that is used in this thesis.

MAVs [8].

However, the signal delivered by the camera is high dimensional and can not directly be interpreted as position or orientation measurements. Computer Vision algorithms are required to interpret the image and extract relevant information. Instead of designing an algorithm manually, the image processing is learned by using annotated examples. In particular Deep Learning based methods aim to combine whole Computer Vision pipelines into one mapping that transforms the raw input image into a task dependent output. Experiments have shown how Deep Learning based methods outperform traditional Machine Learning approaches and manually crafted algorithms in a lot of examples [37]. This made them the predominant choice for almost any vision task.

The hereby used Convolutional Neural Networks (CNNs) are designed in a hierarchical way, using multiple layers that are evaluated sequentially. An example architecture is displayed in Figure 1.2. The input image I consisting of the individual pixels $I_1..I_5$ is evaluated by the model. Each layer applies its corresponding nodes N and transforms the input signal until the output O is reached. Thereby each N can apply a non-linear transformation parametrized during the learning process. By stacking more layers on top of each other (deepening) and increasing the number of nodes per layer (widening), highly non-linear functions can be modelled. Various experiments have shown the superior performance of particularly deep/wide models [13, 14, 46, 57]. However, this model flexibility assumed to be the reason for their superior performance also leads to immense requirements in computational resources. For example a state-of-the-art Computer Vision model [14] contains 60.2 million parameters and one inference requires 11.3 billion floating point operations [51].

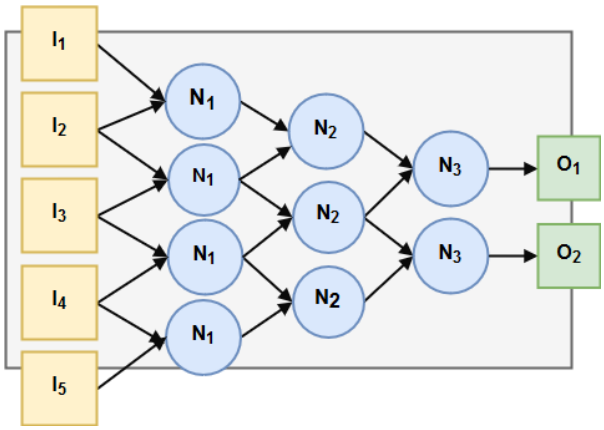


Figure 1.2: Simple example Architecture of a CNN with three layers.

Robotic platforms like MAVs usually need to follow real-time constraints but that have limited resources in terms of processing power and battery life. Here, the deployment of such methods is still an open challenge. Despite a lot of research, that addresses to reduce the number of computations in Deep Learning models [11, 17, 26, 44, 57, 58] the investigation of relatively shallow models with less than ten layers received only little attention by the research community.

This work investigates the deployment of a Deep Learning based Computer Vision pipeline on a MAV. The method is applied in the challenging scenario of Autonomous Drone Racing at the International Conference of Intelligent Robots (IROS) 2018. Within the race court several metal gates are placed and need to be passed one after another. Detecting the gates allows to estimate the MAV's relative position and to calculate the flying trajectory. An overview of the race court and the racing gates at the IROS 2016 Autonomous Drone Race can be seen in Figure 1.3.

Reference to Current Method once it is published

The thesis builds on previous work by Ozo et. al which uses a manually crafted image processing method to detect the racing gates. Although fast to execute the method is very sensitive to illumination changes.

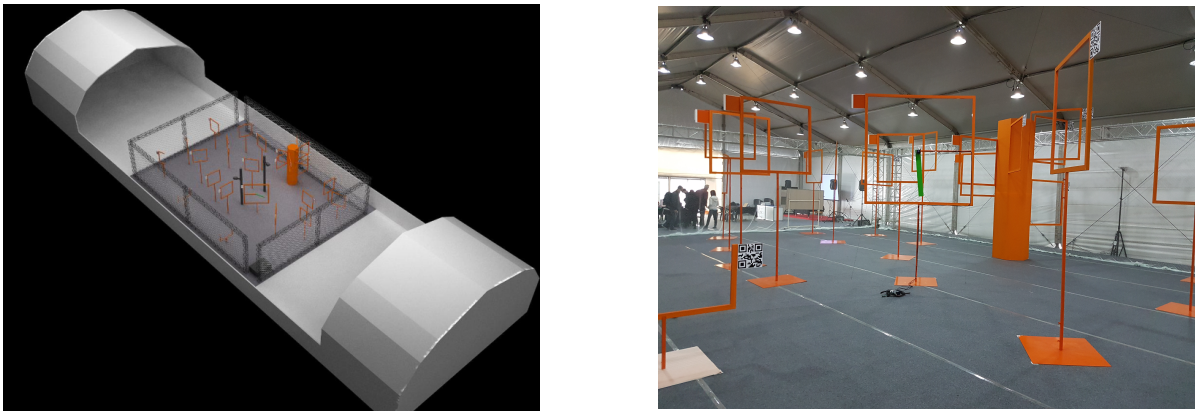


Figure 1.3: Example Images of the IROS 2016 Autonomous Drone Race

Moreover, the algorithm fails when the objects are too far away or the frame is very thin. In order to develop a more robust method, this thesis investigates a learning based approach to the detection of racing gates.

Object Detection is one of the most intensively studied topics in Computer Vision. However, the objects investigated are usually solid and contain complex shapes. For example a pedestrian consist of body parts and a face. A box that surrounds the object mostly contains parts with distinctive shape an/or texture. A Computer Vision model can use these features for detection. The racing gates in contrast are of different nature. As can be seen in Figure 1.3 a box that surrounds the object would largely contain background. Hence, this part can not be used as a hint whether an object is present. Instead it can contain other objects even other gates that might distract a detector. Additionally, the object parts themselves are of very thin structure and can be hardly visible. Thus, a detector needs to make use of fine-grain structures, while ignoring the majority of the image. This introduces a particular vision task that even humans have a hard time at solving [1] and that affects the training and design of a Computer Vision pipeline that aims to detect these kind of objects.

This thesis defines a class of objects as **Empty Wire Frame Objects (EWFO)** studies methods for their detection. The definition is given as follows:

Definition - Empty Wire Frame Objects

1. The object does not consist of complex but only basic geometric shapes like corners, lines and edges.
2. The object parts themselves are thin structures.
3. The object parts can be spread over large parts of the image.
4. The object parts are sparse. The bounding box around the object is largely occupied by background.

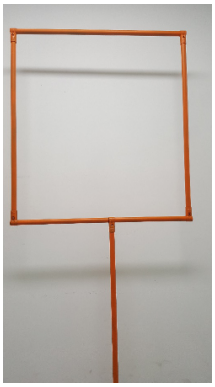
The detection of EWFO is studied in two examples which can be seen in Figure 1.4 . To the best of the authors knowledge EWFO have not been particularly addressed in Computer Vision. In [9] and [27] the authors also detect racing gates, however the used objects contain more structure than the ones investigated in this thesis. Jung et al. present a framework to detect similar objects in [22] and [23] but they do not study the particular effects of the object shape. Chapter 4 particularly addresses the implications of the object shape in modelling a Deep Learning based detection system for EWFO.

replace this with nicer images

A drawback of Deep Learning based vision systems is their need for vast amounts of annotated examples, which is not available for many applications. Racing gates for example are not an object that appears often in everyday life and therefore not many example images exist. To this end no publicly available dataset can be

[1] The unconvinced reader can try to count the number of gates visible in the right image of Figure 1.3

used to train a Computer Vision system for EWFO. Since the object is transparent, it is particularly crucial that the training set covers a large variety of backgrounds. Otherwise, it is likely that a model uses the background for prediction and only works in a particular domain. Although building a race court, placing it in a large variety of scenes and annotating taken images is theoretically possible, it is a cumbersome and very time consuming task.



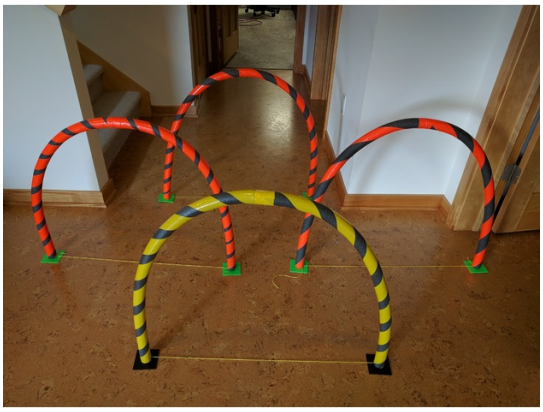


Figure 1.4: Example Images of the Empty Wire Frame Objects investigated in this thesis.

Moreover, the view of a First Person View (FPV) camera mounted on an MAV can be substantially different from the high quality images that are commonly used Machine Learning datasets, as well as between different MAV-platforms. For example the often used wide-angle lenses, used to increase the FoV of the visual sensor, lead to image distortion that changes the appearance of objects and the scene. Moreover sensor bias has been shown to significantly influence the performance of neural networks [1, 7]. A Machine Learning based vision system needs to take these implications into account when selecting the training data.

Alternatively to annotating real images the training data can be synthetically created. This not only allows the rendering of a large amount of backgrounds, it also enables the theoretically infinite generation of training data. Moreover, it allows to model and incorporate domain specific properties like lens distortion or motion blur into the training process. However, a graphical engine can never fully model the real world which needs to be considered when generating synthetic data. Chapter 3 studies the generation of artificial data to train an object detector for EWFO.

Without an accurate detection of the racing gate, the MAV is not able to determine its current position and thus to calculate its flying trajectory. On the other hand, with an algorithm that requires less computational resources a lighter MAV can be built. This allows faster and more aggressive trajectories as well as longer battery life. Hence, the trade-off between accuracy and inference speed is of particular interest for this application and is addressed in Chapter 5.

1.1 Research Question

This section summarizes the research question addressed in this thesis. Furthermore it describes how the question is split in multiple subquestions that are addressed in the individual chapters.

How can the detection of empty wire frame objects on an MAV with resource and time constraints, be modelled and learned from synthetic data?

RQ1 How can data be generated to train a detection model for EWFO detection on a MAVs?

RQ2 How can a detection model represent EWFOs?

RQ3 What are the trade-offs in detection performance and inference time when a detection model for EWFOs is deployed on a MAV?

RQ4 Can the gained insights be used to build a lightweight and robust detection model for racing gates in the IROS Autonomous Drone Race?

1.2 Results/Contributions

Put some results at the end.

1.3 Outline

Refactor contributions once done

Figure 1.5: Thesis Outline

The thesis is structured as displayed in Figure 1.5. Chapter 2 describes the metrics and systems used for evaluation. Chapter 3, Chapter 4, Chapter 5 and Chapter 6 address the individual research questions. Each chapter contains an introduction to the topic, the methodology used in this thesis and experiments that have been carried out. Chapter 3 describes methods to generate synthetic data for machine learning. It concludes with the datasets used for the remaining parts of this thesis. Chapter 4 describes object detection and evaluates current methods in the application for wire frame objects. Chapter 5 illustrates and evaluates measures to reduce computations and optimize an object detection system for a particular hardware. It investigates the trade-off between detection performance and inference time. Chapter 6 describes how the gained insights are used to develop a detector for racing gates at the IROS 2018 Autonomous Drone Race. It also compares the current method to a traditional image processing method in terms of speed and detection performance. Chapter 7 discusses the overall results and formulates a conclusion.

Chapter 2

Evaluation Metrics

describe metrics and hardware

Chapter 3

Synthesizing Data for Object Detection on MAV

Machine learning relies on the assumption that a general concept can be extracted from a limited set of examples. In Supervised Machine Learning, the set of examples X is augmented with a set of labels Y that describes task dependent information that is present in the sample. By defining and optimizing a loss function L the concept is extracted and represented by a hypothesis h. For the optimization procedure different models can be applied. They differ in model assumptions, as well as the amount of parameters that are optimized during training.

The bias-variance trade-off relates the amount of annotated examples used for training and model complexity. It states that with a given amount of training examples, a more complex model might be able to improve in that particular training set, but it will therefore loose performance in other datasets (reduced bias). On the other hand a more simple model will maybe not capture the full information of the training set, but it will have a more stable performance across datasets (reduced variance). The only way to use a more complex model without increasing the variance is to increase the amount of training data.

In recent years Deep Learning showed a vast improvement in a lot of Computer Vision Tasks compared to more traditional approaches. A reason assumed to be responsible for this improvement is the big model size as well as the large amounts of data used for training.

Example of amount of images typically used

However, in many practical applications the gathering of such large amounts of training data is impractical. Firstly, because next to the recording of images also annotation of the samples is required. Secondly, because from a lot of applications not even suitable samples exist.

Therefore it has been a long interest in Computer Vision to automatize the data generation process. By synthetically generating samples with corresponding labels, the theoretical amount of training data is infinite. In addition, it can allow to generate data for domains that are uncommon and where samples are difficult and expensive to obtain.

Yet bears the generation of data its own challenges. First and foremost because the generation process in itself is based on model assumptions. If these do not sufficiently capture the real world, a model trained in such an environment might be heavily biased and perform poorly in the real world. Secondly, because the generation of visual data requires a lot of computational resources. Despite recent advances in Computer Graphics can virtual environments not yet fully capture the real world.

The domain of MAV-Vision is a particular field where samples are not easy to obtain. One reason is the fact that different sensor and lens vary depending on camera and MAV thus an image of the same scene can

Figure 3.1: Example of the Cyberzoo dataset. On the left an image while the MAV is hovering, on the right an image during a turn manoevre.

appear substantially different. Hence, a training set obtained from one MAV might not be applicable for another MAV. While it is possible to remove such effects in post-processing, this can lead to information loss and requires on-board resources.

Furthermore, MAV typically record video data. While such data provides a lot of information the real time storage given the limited resources is not an easy task. In addition, the manual labelling of these videos is very work intense. Finally, as MAV are brittle vehicles and mistakes in development can lead to damage on hardware, engineers and researchers often use simulators to evaluate their systems before transferring them to the real work. Thus the basic infrastructure required to generate data is often already available. Hence, the use of data generation is particularly useful for the MAV domain.

The detection of EWFO is an example of the case mentioned before. In everyday life these object do not appear and thus not many examples exist that can be used for training. To this end no annotated dataset is publicly available. In addition, EWFO are largely occupied by background. Thus if samples are taken from only one race court the variation in backgrounds is quite limited. A model trained on such a set is likely to use the background as a cue and does overfits to this domain. This makes the creation of a dataset for EWFO particularly cumbersome. Therefore, data generation would benefit the development of Computer Vision models for EWFOs.

In Chapter 3 example images of the target domain of this work are displayed. The images are taken during a test flight at a test environment. The left image shows an example when the MAV is hovering and thus is in a very stable position. The object in this case is clearly visible as a single orange square. In contrast the right image shows a close up example during a turn manoeuvre. Here it can be seen how the used wide angle lens causes distortion and thus the lines to appear as circular shape. Furthermore, the large parts of the image including the horizontal bars of the object in the back appear blurred due to the circular velocity of the MAV. In addition, the light conditions of the environment significantly influence the appearance of the object.

Since complex Computer Vision models benefit from large training sets which are particularly hard to obtain for EWFO detection on MAVs, this work addresses the generation of data for this application. The aim is to create a synthetic source domain S in such a way that the trained model h performs well in various target domains T of the real world. The performance is evaluated by applying the learned model on real world datasets and calculating a performance metric m.

The amount of annotated examples consists of 2 datasets obtained during test flights and manually labelled. These are described in the following:

1. *Cyberzoo.* The dataset consist of 200 images taken during a test flight at the cyberzoo at TU Delft Faculty of Aerospace. The race court consist of two gates which are passed in circular fashion.
2. *Testarea.*

insert once done

As only very little examples of the real world are available the datasets are only used for evaluation. However, as the real data only covers a subset of possible application domains additional domain shifts are simulated and their effects on performance is studied.

This chapter focuses on data generation while the exact model is described in Chapter 4. The relevant question to be investigated in this chapter is the following: **How can data be generated to train a detection model for EWFO detection o**

RQ1.1 What are the implications of the properties of EWFOs when synthesizing data?

RQ1.2 How can target domain knowledge be included in the data generation process?

RQ1.3 How can data be generated in a way that the trained model is robust against expected domain shifts?

RQ1.1 will be answered by comparing the implications of domain properties on EWFO to a more complex object. RQ1.2 will be answered by modelling aspects of the target domain and investigating how their incorporation in the data generation process affects performance. RQ1.3 will be answered by simulating various domain shifts and studying the effects on model performance.

The remaining parts of this chapter are structured as follows: Section 3.1 discusses relevant related work. Section 3.2 describes the used methodology. Based on the gained insights Section 3.2.5 formulates several hypotheses to be investigated. Section 3.3 outlines the experiments conducted to evaluate the formulated hypotheses. Section 3.4 describes the obtained results. Section 3.5 discusses the results and Section 3.6 answers the research question.

3.1 Related Work

Data generation has been studied in a wide range of publications. Methods vary from changing low level properties of the image over using CAD models in combination with real background up to rendering full 3D-environments. Often various combinations of synthesized and real data are applied.

3.1.1 Low-Level Image Augmentation

A common part of current Computer Vision pipelines is to augment a given data set by transforming low level properties of the image. By artificially increasing variations in the input signal, a model that is more invariant to the augmented properties shall be obtained.

Krizhevsky et al. [25] use Principal Component Analysis (PCA) to incorporate colour variations. Howard [16] shows how several image transformations can improve the performance of a CNN-based Classification model. The proposed pipeline includes variations in the crop of the input image as well as variations in brightness, color and contrast. In CNN-based Object Detection Szegedy et al. [47] uses random scaling and translation of the input image, as well as random variations in saturation and exposure. Liu et al. [30] additionally crop and flip each image with a certain probability.

Since most methods use image augmentation and Krizhevsky et al. [25] mentions it to be the particularly reason for superior performance at ILSVRC2012 competition it can be assumed to be beneficial for Computer Vision models. Unfortunately, none of the publications measures the improvements gained by the different operations. This work includes a subset of the proposed techniques and measures its effect on model performance.

While the aforementioned approaches add artificial variation to the input data, Carlson et al.[2] augment the image based on a physical camera model. The proposed pipeline is applied for Object Detection and incorporates models for sensor and lens effects like chromatic aberration, blur, exposure and noise. While being of minor effect for the augmentation of real data (0.1% - 1.62% Mean Average Precision (mAP)70) the reported results show an improvement when training on fully synthesized datasets. Here the reported gains vary between 1.26 and 6.5 % mAP70.

Low-level image augmentation is a comparatively cheap method to increase the variance in a dataset. However, it cannot create totally new samples or view points. Furthermore, it cannot change the scene in which an object is placed. Therefore it needs a sufficiently large base dataset that is augmented. This work addresses the case when no real training data is available. Hence, low-level image augmentation is incorporated in the training process but can not be the only method applied.

3.1.2 Augmenting Real Images with CAD - Models

Computer Aided Design (CAD)-models describe the 3D-shape of an object. In a lot of industrial applications such models are available as they are also used to produce the object. For Computer Vision it can be placed on real images to artificially create or increase the size of a dataset.

Peng et al.[36] study the use of CAD-models in the context of CNN-based Object Detection. The authors particularly address how modelling of image cues like texture, colour and background affects model performance. The experiments show how the used CNNs are relatively insensitive towards context but use shape as primary, texture and colour as secondary most important features. This enables competitive performance even when the object of interest is placed only on uniformly covered backgrounds. However, the study only covers solid objects such as birds, bicycles and airplanes. EWFO are substantially different and we hypothesize that other image cues must be relevant.

Madaan et al.[32] study the segmentation of wires based on synthetic training. As wires similarly to EWFO only consist of thin edges, the application is quite close to this work. However, the experiments focus on a single domain, namely sky images and thus the variations in background are comparatively small. We hypothesize that EWFO are particularly sensitive to such variations and address the application in multiple domains.

Hinterstoisser et al. [15] propose to use a base network that has been trained on real images and to continue training on images with CAD-models. During training the base network is frozen and only the last layers are updated. The method does not use real data but requires a suitable base network. As most available feature extractors (further discussed in Chapter 4) are of a size that is computationally prohibitive for MAV the method is not really applicable for this work.

The use of CAD-models in combination with real backgrounds allows to generate totally new view points for the object of interest. Furthermore, the image background consists of real data and thus the synthetic textures only concern the rendered object. However, the geometric properties like perspective as well as the physical properties like object placement are violated and therefore create an artificial scene. Despite this fact, literature shows that such images can benefit model performance in various cases. Yet, most of the approaches still use real data and/or focus on solid objects with rich textures and complex shape. We hypothesize that since EWFO do not provide these kind of structures the results do not apply in the same way. Hence, we incorporate the method to generate data and investigate how it can be applied for the detection of EWFO.

3.1.3 Fully Synthesizing Data

Training models only in a simulated environment is common for Control tasks. In Computer Vision poor quality of graphic engines and long rendering time made the method less popular. However, advances in Computer Graphics and faster processing technologies enabled the generation of more realistic images and led to the creation of fully synthesized datasets[10, 41]. Various studies tried to incorporate such data in their training process.

Johnson-Roberson et al. [20] train an Object Detection model entirely in simulation. The results show an improvement towards data annotated by humans especially when using vast amounts of simulated data. However, the created environment is highly detailed and therefore requires a lot of engineering work.

There should be more examples for this

In contrast [43, 48, 49] use a relatively simple environment but a high degree of randomization to address the reality gap. The aim is to learn an abstract representation by strongly varying textures, light conditions and object locations. Tobin et al. introduced this technique as Domain Randomization (DR). The drawback of the approach is that a too high degree of randomization may omit pattern in the target domain that could otherwise be exploited by the model.

Training a model in a fully synthesized environment enables the full control of all properties present in an image. The object of interest can be placed according to physical laws, shadows fall correctly and geometric properties of an image are followed. However, if the graphical models do not fully capture the detail of real world objects, the generated data might look too artificial. Methods in literature address this problem in two ways: (1) Creating a virtual environment that resembles the real world. Although a lot of engineering effort and processing power is required, a lot of properties of the real domain can be incorporated; (2) Creating a lot of variance in the data generation to obtain an abstract representation. While bearing the risk of omitting properties of the target domain, this method requires less engineering effort.

This work addresses the application of MAVs in GPS denied scenarios. Such scenarios cover a wide range of possible environmental conditions and a full modelling of all these possible domains is beyond the scope of this work. On the other hand, we hypothesize that a model with a high degree of abstraction might perform well across domains but poor in a particular domain. Hence, we investigate this trade-off for the detection of EWFOs.

3.1.4 Transfer Learning

The field of transfer learning particularly addresses domain shifts in the modelling process. Hence, a common application is the learning from synthetic data.

A common approach in CNN-based models is the incorporation of a domain classifier in the model. By augmenting the data with domain labels, the classifier learns to distinguish the two domains. Subsequently a gradient reverse layer is applied and thus the weights are updated in such a way that a domain agnostic representation is learned. Examples of the approach can be found in [4] [56].

While the aforementioned approaches require labelled samples from the target domain, Peng et al. [35] propose to include task-irrelevant samples and a source classifier. As a result no samples of the target domain are required.

While transfer learning provides the theoretical framework as well as methods to deal with domain shifts, it does not allow to generate data. Furthermore, it often requires samples of the target domain. This work addresses the case when no real data is used for training. The field is interesting to be incorporated in the data generation pipeline investigated in this thesis but it can not be used as a start off point. Hence, the use of transfer learning in the modelling process is denoted as future work.

3.1.5 Generative Adversarial Networks

write [19]

3.2 Methodology

In the following, methods used in this thesis are described. The different steps are implemented in a data generation pipeline and the effect of the individual steps are studied in various experiments. An overview can be seen in Figure 3.2. The individual steps are described in the following.

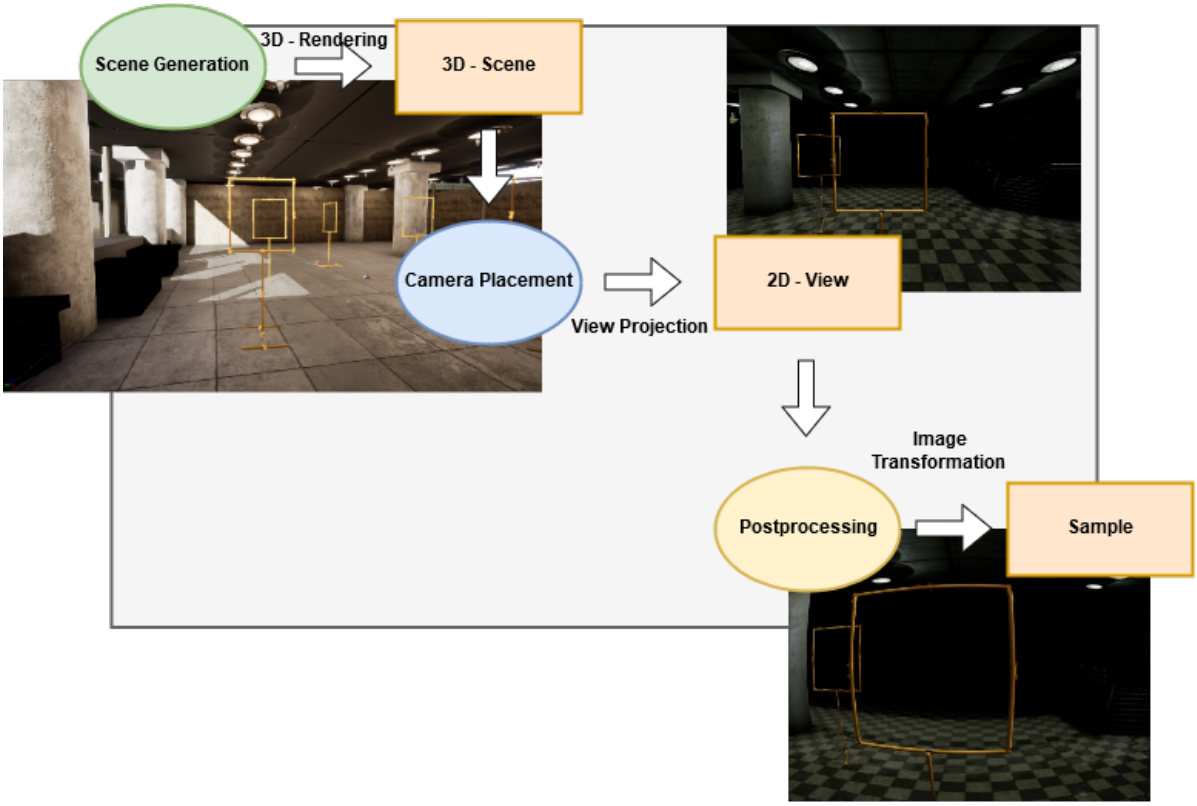


Figure 3.2: Overview of the data generation process. The environment model determines background and lightning conditions and produces a 3D-Scene. The motion model determines the camera pose and location and thus the view point. Projecting the 3D-Scene on the image plane of the camera delivers the 2D-View. A final post processing step incorporates sensor and lens effects.

3.2.1 Scene Generation

The first step creates a scene in which the object is embedded. Therefore it determines the context as well as light conditions.

Pasting a CAD-Model on Real Images

Inspired by [12, 36, 42] a 3D-Model of the object of interest is pasted on real images.

TODO The 3D-Model is provided by TODO and a tool to render these models is implemented using *OpenGL*. A scene

with black background is created and several objects are placed on a virtual ground plane with different rotations to each other. Additionally light sources are placed at different locations and with different intensities.

After creating a scene the black background is replaced with an image obtained from a dataset. As the light conditions between background image and rendered scene are different, the created image can be quite artificial. Hence a post-processing step applies a Gaussian blur kernel along the edges of the CAD-model. This leads to a better embedding of the object in the background. Example outcomes can be seen in Figure 3.3

The parameters and locations of objects and light sources as well as the selected backgrounds can be varied when creating a dataset.

Figure 3.3: Examples for pasting the 3D-Model on various backgrounds. A scene with background is rendered (left), subsequently the black background is replaced with an image from a dataset.

Full Rendering of a Scene

Following [10, 20, 41, 48, 49] the second approach to create a scene is to fully render the environment using a graphic engine, namely the *UnrealEngine* including the *AirSim* plugin.

The *UnrealEditor* allows to manually create environments and provides high quality rendering of objects, textures and light conditions. In total three indoor environments are created. This resembles GPS-denied scenarios as they are targeted in this thesis. Within the environment light conditions, background textures, object locations can be changed manually.

In total three base environments are created. Examples can be seen in Figure 3.6. The three environments are described in the following:

1. *Basement:* The environment is a room without windows, only containing artificial light sources.
2. *Daylight:* The environment is a room with windows along all walls that allow daylight to illuminate the room. The windows can lead to strong variations in the contrast between different parts of the object.
3. *IROS2018:* The environment resembles the room of the IROS Autonomous Drone Race 2018. The light sources stem from a window front at one side of the room, as well as artificial light sources at the ceiling. Depending on the view point, the object might appear against bright or dark background.

Figure 3.4: *Basement* Environment.

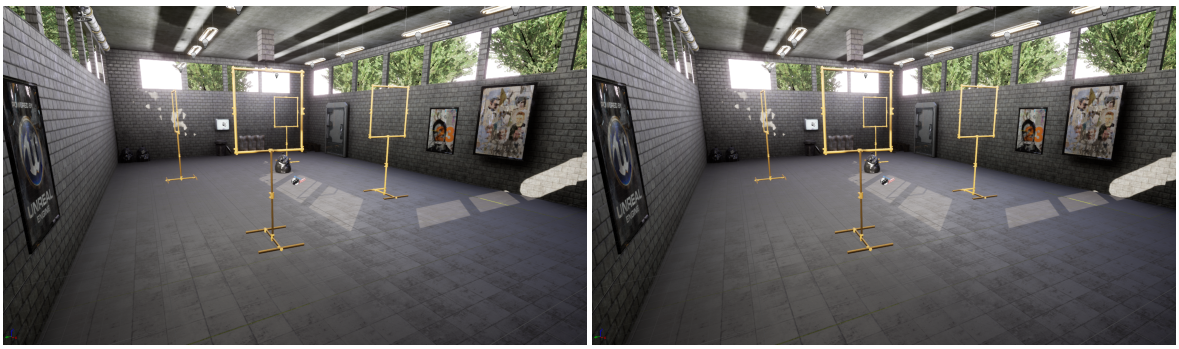


Figure 3.5: *Daylight* Environment.

Figure 3.6: *IROS2018* Environment.

3.2.2 Camera Placement

The second step places the camera in the scene and creates a 2D-image. Hence, it determines the perspective of the image on the scene. The camera pose is determined by:

$$\text{Translation: } t = [xyz] \text{ and Rotation: } r = [\phi, \theta, \psi]$$

Where the North-East-Down (NED) coordinate system is used originating from the initial position of the camera.

Random Placement

A straightforward way of placing the camera is the random placement in the scene. The value for each dimension of r and t are drawn from a probability distribution. The chosen distributions have to follow certain limitations, for example the gate should still be visible in most images.

Quad-rotor Model

r and t follow the motion model of a quad-rotor MAV. The development of this model has not been done within this thesis but is summarized here for completeness: .

put shuos model here

Using this motion model, the camera follows a certain trajectory through the 3D-Environment. Storing the current image at a frequency of 2 Hz creates the corresponding samples.

3.2.3 Post-processing

In the final step low-level image transformations are applied. Steps include sensor model based an artificial augmentation.

Model-based augmentation

The applied pipeline is strongly based on [2]. However, on the camera of the MAV we assume that the raw image signal can be processed. Hence, the model for information loss due to post-processing is not included. Instead all images are converted into YUYV-colour space, a format that is obtained from most visual sensors as well as the target platform of this thesis. The pipeline is extended by two models: (1) A model for lens distortion since MAV often use wide angle lenses to increase their FoV; (2) A model for motion blur since fast camera movements are to be expected. The individual models are described in the following.

Lens Distortion Lens distortion is a form of optical aberration which causes light to not fall in a single point but a region of space. For MAVs commonly used wide-angle lenses, this leads to barrel distortion and thus to straight lines appearing as curves in the image.

The effect is applied using the model for wide-angle lenses from [52]. It models the removal of lens distortion as combination of radial and non-radial part, that is approximated with a second order Taylor expansion:

$$\begin{pmatrix} p'_x \\ p'_y \end{pmatrix} = \begin{pmatrix} p_x(1+\kappa_1 p_x^2 + \kappa_1(1+\lambda_x) p_y^2 + \kappa_2(p_x^2+p_y^2)^2) \\ p_y(1+\kappa_1 p_x^2 + \kappa_1(1+\lambda_y) p_y^2 + \kappa_2(p_x^2+p_y^2)^2) \end{pmatrix} \tag{3.1}$$

Where:

- p'_x and p'_y are the undistorted coordinates.
- κ_1 controls the primary distortion (default 0)
- κ_2 controls the secondary distortion (default 0)
- λ_x and λ_y controls asymmetric distortion (default 0)

Applying the lens distortion to an image is done using the inverse of Equation (3.1). However, as there is no closed form solution, so the Newton-approximation is used:

$$p_i = p_{i-1} - \nabla p^{-1}(f(p)_i - p') \tag{3.2}$$

Where f is the function defined in Equation (3.1).

An example with $\kappa_1 = 0.5, \kappa_2 = 0.5$ is displayed in Figure 3.9. It can be seen how the previously straight lines appear as circular shape.

Chromatic Aberration. Chromatic Aberration is caused when different wavelengths of light do not end up in the same locations of the visual sensor. This leads to a shift in the colour channels of the image.

Similarly to [2], chromatic aberration is applied by scaling the locations of the green channel, as well as applying translations on all channels. The model can be implemented as affine transformation of the pixel locations for each channel:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S & 0 & t_x \\ 0 & S & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.3}$$

An example is displayed in Figure 3.8. It can be seen how the red and green channel are shifted relative to each other. Thus two bars appear in the image.

Motion Noise. Motion noise is caused when light falls in different locations of the images sensor due to a fast movement of the camera. It leads to blurry images based on the sensor motion.

The phenomenon depends on camera properties as well as the motion of camera and objects. Although a full modelling of this process might benefit the learning process, it requires a complex pipeline and is computationally expensive. Therefore a strong simplification is used, namely a one-dimensional Gaussian filter:

$$K_v = \begin{pmatrix} \dots \\ \mathcal{N}(\mu-1) \\ \mathcal{N}(\mu) \\ \mathcal{N}(\mu+1) \\ \dots \end{pmatrix} \quad K_h = \begin{pmatrix} \dots & \mathcal{N}(\mu-1) & \mathcal{N}(\mu) & \mathcal{N}(\mu+1) & \dots \end{pmatrix} \tag{3.4}$$

Where $\mathcal{N}$ is a Gaussian-PDF with mean μ and variance σ, K_v models vertical motion blur, K_h horizontal motion blur. The size of the kernel is chosen by k.

An example for vertical blur is displayed in Figure 3.11. It can be seen how particularly horizontal lines appear softer.

Out-of-focus Blur. Next to motion, sensor noise can lead to blurry images. For the blur operation a 2D Gaussian kernel is applied on the input image with:

$$k = \frac{1}{2\sigma_x\sigma_y\pi} e^{-\sqrt{\frac{x^2+y^2}{2\sigma_x\sigma_y}}} \tag{3.5}$$

Exposure. Exposure is the time the sensor records light in order to create an image. Over- and Underexposure are caused when this time is too short or too long, leading to too dark or too bright images.

Following the model from [2]:

$$I = f(S) = \frac{255}{1 + e^{-AS}} \tag{3.6}$$

where A is a constant term for contrast and S the exposure.

The image can be re-exposed with:

$$I' = f(S + \Delta S) \tag{3.7}$$

where S is obtained from :

$$S = f^{-1}(I) = \frac{\ln(\frac{255}{I} - 1)}{-A} \tag{3.8}$$

An example for overexposure is displayed in Figure 3.12. It can be seen how lighter areas appear particularly light, while dark areas remain dark.

Noise. text

3.2.4 Artificial Augmentation

Inspired by [16, 30, 47] the application of several artificial image transformations is studied. The overall goal is to generate more variations in the input signal and thus to make the model more invariant against changes in those properties. We do not use image scaling or translation for augmentation as it is easier to incorporate such variations using the motion model.

Brightness. In order to obtain a model that is more robust against illumination changes image brightness is alternated. Therefore a scaling on the V-channel in HSV-colour space is applied. The scaling is drawn from a uniform distribution.

Grayscale. By transforming a subset of samples into grayscale images, the model is forced to learn more color invariant features.

Histogram Equalization. Changes in the environmental conditions can also affect contrast. By applying a histogram equalization on a subset of images, variations in contrast are achieved. describe

Flip. By mirroring the image vertically, more variations in object locations are achieved. The operation is applied with a certain probability.

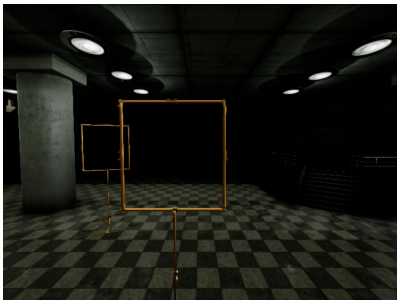


Figure 3.7: Original Image.

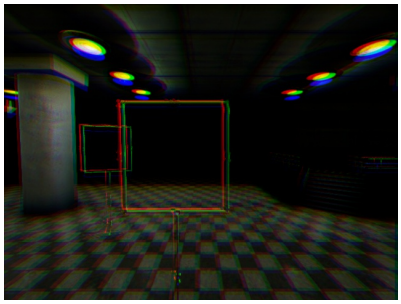


Figure 3.8: Chromatic Aberration.

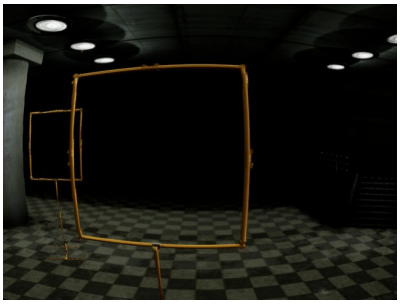


Figure 3.9: Lens Distortion.

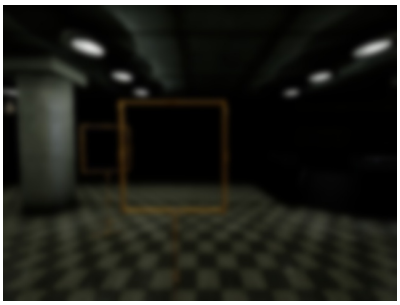


Figure 3.10: Out-of-Focus blur.

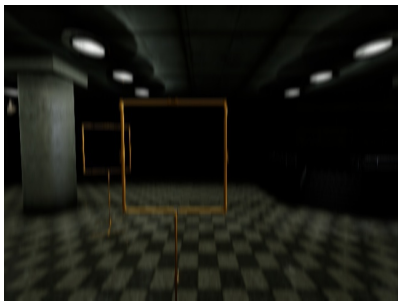


Figure 3.11: Vertical Motion Blur.

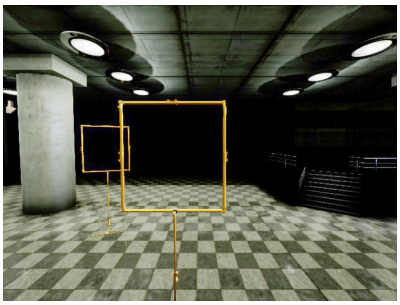


Figure 3.12: Exposure.

3.2.5 Hypothesis

visualizing cnns

Most of described approaches in Section 3.1 focus on objects that have complex structures and therefore provide robust features that are independent of the rest of the scene. For example a face contains eyes and a nose, whose appearances are influenced to some extent by light conditions but relatively independent from image background. It has been shown in how object detectors exploit this kind of structure in the learned representation. We hypothesize that a model to detect such complex objects is less domain dependent and can therefore be more easily transferred to domains with other environmental conditions. That is the performance drop Δm of a model trained in S and applied in T where S and T are different in terms of environmental conditions will be larger for an EWFO than for a more complex object.

If the environmental conditions have an high impact, their modelling is particularly important in the data generation process. We hypothesize that, as pasting the image on random backgrounds fails to capture these conditions, it is not a sufficient method to train a model for the detection of EWFOs. That is the performance drop Δm of a model trained in S and applied in T where S is generated by pasting a 3D-Mesh on random backgrounds and T is modelled using full environment rendering will be prohibitively large.

From that it follows that in order to generate data for EWFO Object Detection it is particularly important to address the domain shift. In literature two ways to approach this problem on the data level can be found: (1) providing a lot of variance in the training data to obtain a representation that is robust against domain shifts; (2) including target domain knowledge in the training data to obtain a representation that is tailored to the target domain. We hypothesize that there is a trade-off between these two approaches: a model that performs well across domains will perform poorer in a particular domain but better in other domains compared to a model that is trained for that particular domain. Hence, we hypothesize, if knowledge of T is included when generating S the performance in T will improve.

In the data generation process this can be addressed on several levels:

1. **Scene Generation.** A model trained for a particular room, with particular lightning conditions will perform better in that room but perform poorly in other rooms than a model trained on various rooms

with various lightning conditions. By modelling the environment of the real data, the performance on the real data can be improved.

2. **Camera Placement.** A model trained using the quad-rotor model will perform better on real data than a model trained using random camera locations.

3. **Postprocessing.** A model trained using a post-processing pipeline that models the real-world sensor will perform better on the real data than a model that is trained on using varying parameters in the post-processing pipeline.

The hypotheses are summarized in the following:

$\mathscr{H}_1$ The performance drop Δm of a model trained in S and applied in T where S and T are different in terms of environmental conditions will be larger for an EWFO than for a more complex object.

$\mathscr{H}_2$ In contrast to a more complex object, the performance drop Δm of a model for the detection of EWFOs trained in S and applied in T where S is generated by pasting a CAD-Model on real backgrounds and T is modelled using full environment rendering will be prohibitively large.

$\mathscr{H}_3$ A model trained in S_0 where $S_0 \in S$ and applied in T_0, where $S_0 = T_0$ will perform better in T_0 than a model that is trained in S but perform worse in T_1 where $T_1 \in S$ and $T_1 \neq S_0$.

$\mathscr{H}_3$ By including properties of T in S where S is an artificial set T is the real data, the performance m_T of a model can be improved.

3.3 Experiments

In order to evaluate the formulated hypotheses several experiments are conducted. The model used is the TinyYoloV3-Architecture, further described in Chapter 4. The reported metrics are described in Chapter 2. For all experiments mean and standard deviation of 5 runs are reported.

For the random view point generation the following parameters are used:

$$x = \mathscr{U}(-30,30), \quad y = \mathscr{U}(-20,20), \quad z = \mathscr{N}(-4.5,0.5)), \quad \phi = \mathscr{U}(0,0.1\pi), \quad \theta = \mathscr{U}(0,0.1\pi), \quad \psi = \mathscr{N}(-\pi,\pi) \tag{3.9}$$

Where $\mathscr{U}(a,b)$ is a uniform distribution between a,b and $\mathscr{N}(\mu,\sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2.

The parameters are chosen experimentally aiming to resemble common view points of a person standing in the room.

3.3.1 Experiment I

In order to evaluate $\mathscr{H}_1$ a model is trained in S and applied in T. Three objects are investigated namely the *Square-Gate*, *Round-Gate* and *Person*. The source environment is *Basement*, the target domain is *Daylight*. The training set consists of 20 000 samples, where for every batch of 2000 samples the objects are rearranged in the room. The test set consists of 1000 samples and fixed object arrangement. The view points are samples from the distributions described in Equation (3.9)

3.3.2 Experiment II

In order to evaluate $\mathscr{H}_2$ a model is trained in S and applied in T. Three objects are investigated namely the *Square-Gate, Round-Gate* and *Person*. The training set consists of 20 000 samples, where for every batch of 2000 samples the objects are rearranged in the room. The test set consists of 1000 samples and fixed object arrangement. The training set is generated by replacing the background with images samples from the Pascal VOC dataset. The test set is taken in *Daylight* Environment. The view points are samples from the distributions described in Equation (3.9)

3.3.3 Experiment III

In order to evaluate $\mathscr{H}_3$ the individual domain properties and their incorporation in the data generation process are studied. Each property is compared in terms of specialization and generalization. That is the property are varied between three configurations: (1) resembling the target domain, (2) generalizing across domains, (3) disabled (if applicable).

The test set is generated in the environment *IROS*, using the quad-rotor model. In total 1000 images are sampled taken from one trajectory. The post-processing applies:

1. Lense distortion
2. Chromatic Abberration
3. Motion Blur
4. Out-of-Focus Blur
5. Exposure

3.3.4 Experiment IV

In order to evaluate $\mathscr{H}_4$ the individual domain properties are measured on the target domain and incorporated in the training set.

3.4 Results

3.5 Discussion

3.6 Conclusion

Table 3.1: Results of varying training sets on TODO

	General		Specific	General + Specific	Inactive
Scene	Real Backgrounds	*Basement, Daylight*	*IROS2018*	Real Backgrounds, Synth. Environments	-
					-
Camera Placement	Heuristic Selection	Fitted on target set		Heuristic Selection + Target Set	
Lens Distortion	Varying parameters				
Chromatic Aberration					
Motion Noise					
Out-of-Focus Blur					
Exposure					
Sensor Noise					
Brightness					
Grayscale					
Histogram					
Flip					

Chapter 4

Modelling the Detection of EWFO

Object Detection can be described by two individual goals: the description of what kind of object is seen (Classification), as well as where it is seen (Localization). Hence, an Object Detection pipeline transforms the raw image to a set of one or more areas and corresponding class labels. Images are high dimensional signals that can contain redundant and task irrelevant information. As the performance of most machine learning models decreases when the feature space becomes too large (curse of dimensionality), Computer Vision pipelines usually apply a feature extraction stage, before the actual prediction is done. An overview is displayed in Figure 4.1.

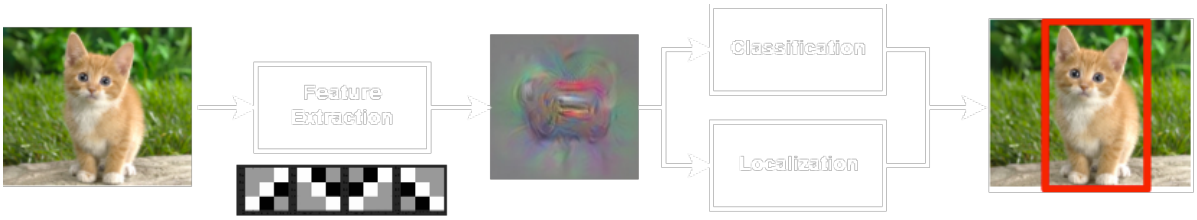


Figure 4.1: Object Detection Pipeline.where B_n describes an area, C_1 a class label, I the image and f the object detection function.

1. The feature extraction stage extracts task relevant information from the image and infers an internal, more abstract representation that is usually of lower dimension.
2. The classification/localization stage produces the final output based on this representation.

An efficient feature extraction pipeline is thereby crucial for the success of an Object Detection pipeline. If the inferred representation is clearly separable, a simple classification stage can distinguish between classes. On the other hand even a flexible classifier struggles with a highly overlapping feature space. Hence, Feature Engineering, the design of feature extractors is a highly investigated field in Computer Vision. Methods range from supervised and unsupervised Machine Learning techniques to conducting domain experts and trying to include their expert knowledge into the pipeline. In practical Computer Vision problems this often results in the cumbersome design of feature extractors for the application. Moreover, it requires domain knowledge and it is questionable whether the features designed for one task are easily transferable to other tasks. Finally, it results in a pipeline where each step needs to be optimized individually.

In recent years Deep Learning based models achieved big advances in the task of Object Detection. In contrast to their traditional counter parts, these models combine Feature Extraction and Classification/Localization stage in one model. The whole pipeline is then optimized given the task and the raw image. This omits the often cumbersome work of designing feature extractors and object models. Furthermore, it has been shown that Deep Learning based features generalize well between different Computer Vision tasks [37]. Finally, the

modular architecture of Deep Learning models allows to trade-off computational costs and model performance.

However, their superior performance comes with several drawbacks. First of all, large amounts of annotated examples are required in order to train the vast amount of parameters present in Deep Learning models. Furthermore, the computational costs during training and inference are high. Only faster and tailored processing units like Graphical Processing Units (GPUs) enable the practical application of Deep Learning models. Finally, the presentation learned by the model is not transparent. Hence, the process that leads to the decision of a Computer Vision system can usually not be understand by a human. Also, there is no guarantee that the learned representation is not highly redundant. This is particularly problematic for devices with time constraints and limited computational resources like MAVs.

This work investigates the detection of EWFOs on MAV. EWFOs consist of simple shapes but are largely occupied by background. Hence, other objects of interest and/or distractors can appear in this area. Furthermore, sensor and lens properties as well as motion noise can have large impact on the appearance of EWFOs in the image. This makes the design of an appropriate feature extractor a non-trivial task. Deep Learning would allow to learn a feature extractor and object model given data and the task.

This work address the design of a Deep Learning model for the task of EWFO-Detection.

The relevant research question of this chapter is stated as follows:
How can a detection model represent EWFOs?

RQ2.1 Can state-of-the art models represent an EWFO?

RQ2.2 What is the representation a state-of-the-art model learns for the Detection of EWFOs?

RQ2.3 Can the insights be used to create a more suitable model for the Detection EWFO?

The first question will be answered by analysing the performance of state of the art model on the detection of EWFOs. RQ2.2 will be answered by conduction a sensitivity analysis on the trained model and visualizing the internal representation. RQ2.3 will be answered by refactoring the model architecture and examining whether the performance can be improved or weights can be removed.

The rest of the chapter is organized as follows: Section 4.1 discusses relevant related work. Section 4.2 describes the methodology of this work. Section 4.3 formulates several hypotheses to be investigated. **??** outlines the experiments conducted to evaluate the formulated hypotheses. **??** describes the obtained results. **??** discusses the results. **??** answers the research question and formulates a conclusion.

4.1 Related Work

The existing methods can broadly be grouped in three groups. Those are more traditional approaches without CNNs-based feature extraction, two-stage detectors and one-stage detectors.

4.1.1 Traditional Methods

One of the first object detection methods was [53] that used simple filters inspired by Haar-basis functions as a feature extractor for human face detection. The image was processed in a cascade of classifiers that assigned the label "face" or "background" to image patches. The output of the first stages were further classified when going deeper in the cascade. The processing of one image patch stopped, when a classifier assigned the label "background". Although being very fast the Haar-based feature extraction is not very robust towards rotation-, scale- or shape-variations .

quote

A more robust feature extractor was proposed by [6] and [31] for pedestrian detection. A local (normalized) histogram of gradients is computed for a fixed window size.

Previously mentioned methods modelled objects as one instance. This prove to be sensitive towards part occlusions or large deformations. Deformable part models detect object parts individually and combine them. Thus a feature extractor can still give a high response when even only object parts are visible or they are arranged in a way untypical to the training set.

Guido proposes a neural network to learn an attention model.

X propose a recurrent neural network architecture to model the attention process.

discussion traditional features

4.1.2 CNNs-based Feature Extraction

In recent years CNNs emerged from Deep Learning research and became a popular feature extractor. CNNs can be seen as small neural networks that are applied locally on image patches in sliding window fashion. The outputs of the initial local operations (first layer) are further processed by higher layers until the desired output size is reached. The model parameters (weights) are trained using a Loss function and the back-propagation algorithm.

The modular structure of CNNs allow to create highly non-linear models that can represent any function. However, this flexibility also introduces the challenge of choosing a suitable architecture. On a fundamental level design parameters can be summarized in depth, width and kernel size.

Section 4.1.2 displays these parameters and introduces additional terminology necessary for the remaining parts of this chapter. The *kernel size* **k** determines the spatial size of a kernel and therefore how big the patch is, the convolution is applied on. A layer usually contains multiple filters that are applied on its input. The amount of filters is also referred to as *width w*. The filters are applied in sliding window fashion which introduces the step size (*strides* **s**) as an additional parameter. The output of each convolution is concatenated and processed by the next layer. The amount of layers is also referred to as *depth*. In the image also the *receptive field* of a filter is visualized. This describes the image patch that is related to a certain feature response. The filter of the first layer (green) has a receptive field corresponding to its kernel size. The filter of the second layer (blue) combines the responses of the filters of the first layer at multiple spatial locations an thus has an increased receptive field.

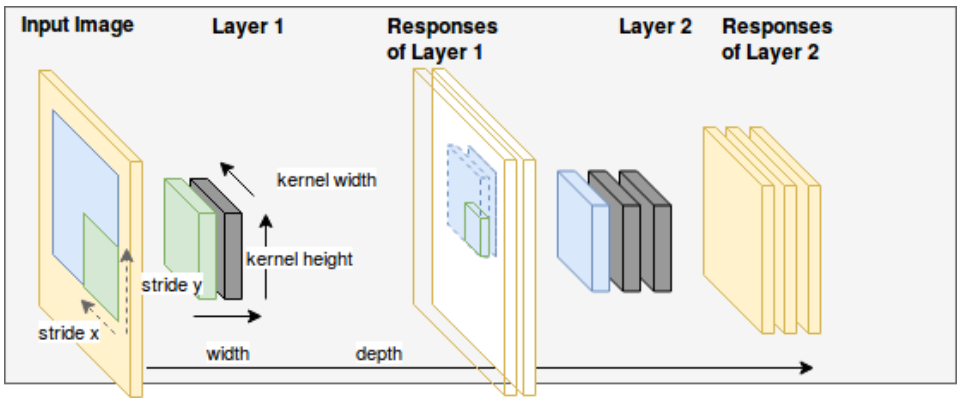


Figure 4.2: Example Architecture of a CNN

Among these parameters depth is considered one of the preliminary parameters to improve performance [13]. **?** [**?**] achieve first places in the 2014 ImageNet Classification challenge using a network that only

contained filters of size 3-by-3 but up to 19 layers. Szegedy et al. [46] achieve similar performance using a network with 22 layers. The proposed network included a *Inception*-module, an architectural element that allows deeper networks at a constant computational budget.

An issue that prevented training even deeper networks is the *vanishing gradient problem*. As the gradient distributes while flowing through the vast amount of nodes its magnitude gets very small. Hence, the training becomes slow and the risk of converging in a local minima increases. This was addressed by He et al. [14] who propose the use of residual connections. Instead of propagating the gradient from the last to the first layer these connections allow the gradient to flow directly into all layers. This circumvents the vanishing gradient problem. The use of residual connections allowed to train a network 101 layers and improved on state of the art at that time.

However, later work by Zagoruyko and Komodakis [57] shows how residual networks do not behave like a single deep model but more like an ensemble of more shallow networks. Moreover, the study shows that similar performance can be achieved by particularly wide networks and residual connections. Being of similar performance the proposed Wide Residual Networks (WRNs) are computationally more efficient to execute.

While wide residual networks can achieve similar performance to deep residual networks with reduced inference time the computational requirements are still large. This work addresses the detection of EWFO with very limited resources. Hence, a network in which the vanishing gradient problem would appear is likely to be already too computationally expensive to be applied on a MAV.

Instead the work focuses on much smaller networks that are fast to execute. Execution time is also the motivation for Fully Convolutional Networks (FCNs). Instead of using a fully connected layer in the last stage, these network only apply local operations. This saves many computations in the last layer and enables the application of models on various input sizes.

However, FCN in combination with a small amount of layers introduce a limited receptive field. A way to increase the receptive field without increasing the number of computations was proposed by Atrous/Dilated convolutions consist of a sparse kernel thereby increasing the receptive field of a filter without increasing complexity or number of computations .

Atrous

Despite a large amount of research conducted in finding suitable architectures there has not yet been a single way that always achieves a goal. It has been shown how models with a large amount of parameters combined with huge training data perform well on various vision tasks and objects. However, there is no guarantee that the found representation is also the most suitable/efficient one. The research resulted in a collection of rules an best practices that need to be considered with the task at hand. This work investigates the design of a CNN for the detection of EWFO.

4.1.3 CNN-based Object Detection

CNNs-based feature extraction is employed in various approaches to Object Detection.

Girshick et al. [12] use the Selective Search algorithm [**?**] to extract object candidates from an image and classify each region with a CNN. However, this requires to run the whole network at various scales and overlapping locations. Hence, the approach contains a lot of redundant operations and is computationally intense.

Ren et al. [39] use a Region Proposal Network (RPN) to propose regions that likely contain an object. In order to define the proposal task as a regression problem, the approach introduces so called *anchor boxes*(also *prior boxes*, *default boxes*). These are boxes of predefined size and location. The model predicts class probabilities and coordinate offsets for each of these boxes. Hence, a certain set of output nodes is responsible for a particular box. If during training a ground truth box has sufficient overlap with a certain box the corresponding

output nodes are assigned "responsible" to predict that object. That means the loss is only propagated via those nodes.

Figure 4.3 illustrates the concept. The anchor boxes are displayed as dashed lines while the ground truth is displayed solid. The ground truth box in blue has sufficient overlap with two anchor boxes. Hence, these two sets of output nodes take part in the loss calculation. In the example each of these sets predicts coordinate offsets $\Delta(cx, cy, w, h)$ and class probabilities $c_1..c_p$.

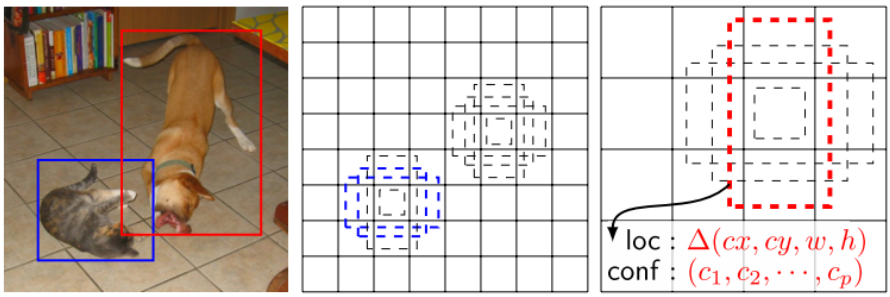


Figure 4.3: Visualization of the anchor box concept [30].

For the classification stage an Support Vector Machine (SVM)-classifier is used. The classifier is trained on the image patches extracted by the first stage. RPNs enabled to propose multiple candidate regions with a single inference of the network. Thus, the expensive feature extraction stage is run only once which results in a significant speed up . A drawback is the fact that individual stages of the method have to be optimized individually. Furthermore, the training requires to store large amounts of extracted patches on the hard drive.

how much?

In the follow up work [39] propose the *ROI*-pooling layer. The layer uses spatial pyramid pooling in order to resize region proposals to a fixed size. This enabled the end-to-end training of the two-stage detection pipeline.

And resulted in?

Another end-to-end pipeline was published by Szegedy et al. [47]. In contrast to aforementioned approaches, the network performs Classification and Localization in a single pass. The task is formulated by dividing the input image in a fixed grid and predicting C class probabilities for each grid cell. Additional $5 * B$ output nodes predict B set of bounding box coordinates and B object probabilities for each cell.

what is good and bad?

Liu et al. propose Single Shot Multibox Detector (SSD) [30], one stage detector using the concept of aforementioned anchor boxes. Instead of only predicting an object score for each anchor box, the model also predicts class probabilities. Another novelty in this approach is the use of multiple predictor layers for various scales. The network does not only use its final layer for prediction but also intermediate representations. Assuming that the lower layers preserve more fine grained features, early output nodes are trained on smaller objects while later output nodes focus on predicting larger scale objects.

Follow up work of Szegedy et al.[38, 47] also included the concept of anchor boxes and prediction layers at multiple scales, making SSD and You only look once (Yolo) converge to a very similar solution. A novelty in [38] is the use of de-convolution layers for small object prediction. In order to achieve a higher accuracy for small objects the final layers are up-sampled and combined with finer grain features from earlier layers. The aim is to enable a combination of deep semantic features at low spatial resolution with fine grain low level features at high resolution.

Within the framework of SSD and Yolo several approaches exist that either change the base network or modify layers in between: [5] propose propose a more efficient non-max-suppression method as well as to include an inception module in the network architecture to reduce computation while keeping/increasing performance. [54] uses *SqueezeNet* as base network and a mixture between the ssd and yolo loss function as training goal. [55] investigates the receptive fields of SSD and tries to incorporate more context, especially on lower feature

maps, to increase detection rate for small objects.[28] applies the framework for vehicle detection. They use *GoogLeNet* as base network (and investigate several others).[50] apply a network very similar to YoloV2 and investigate 8bit quantization of the model to make it runnable on embedded devices.

A common problem of one stage detectors is the imbalance between background and object samples. Most methods upweigh the positive samples and/or use hard negative mining. [29] introduces the *Focal Loss* which focuses on sparse positive samples by design.

CornerNet

Each of the described group of methods has strengths and weaknesses. While shallow methods are typically quite fast they require a lot of manual effort and/or are not so accurate. Two-stage detectors on the other hand are quite accurate but their computational requirements are prohibitive for the hardware to be used in this thesis. One-stage detectors offer a compromise between detection accuracy and inference speed. In addition they can be trained end-to-end which requires only little manual engineering. However, the presented methods are still too slow for the hardware used in this thesis.

Wire detection

4.2 Approach

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum L_{cls}(p_i, p_i*) + \lambda \frac{1}{N_{reg}} p_i * L_{reg}(t_i, t_i*)$$

Where i is an anchor, p_i the predicted probability of the anchor being an object and t_i a vector containing 4 bounding box offsets to the default anchor size. The ground truth label t_i* contains the true coordinates, while p_i* is 1 if the anchor overlaps a ground truth bounding box by some threshold.

Which exact loss functions are used

Comparing state of the art results shows the superiority of CNNs-based methods in basically every vision task . Hence, the first hypothesis formulated is that a state of the art object detector should be able to learn the detection task of wire frame objects.

elaborate

As the single class case is considered the loss functions of state of the art detectors is simplified to the following:

text

Hence, the only difference between X,Y,Z is ... Therefore the second hypothesis to be evaluated is that there is not a very large difference between the mentioned methods.

visualizing cnns

The reason to be assumed responsible for the superiority of CNNs-based methods is the fact that the can learn powerful object representations directly on the task. show how the model combines simple shapes like edges, corners and blobs to more complex shapes like noses and eyes. For the task of wire-frame object detection there is no such intuitive combination of such higher order shapes. Therefore the second hypothesis formulated is that the deeper levels of a CNNs are not necessary for the detector.

4.3 Hypothesis

Several hypothesis are formulated and will be examined experimentally:

$\mathcal{H}_1$ **A CNNs should be able to learn the object detection task.**

$\mathcal{H}_2$ **Considering the single class case state of the art methods will learn the same representation**

$\mathscr{H}_3$ **For wireframe objects the deeper layers don't learn anything as the object consist of relatively simple shapes.**

4.4 Experiments

First we show that many weights in an object detector are superflous when detecting single wire frame objects: 1. We train an object detector on a single but complex object and compare the filters to multiple objects 2. We train an object detector on a single wireframe object and compare the filters to the other feature detectors 3. We use the gained insights to prune the network 4. The pruned network should perform poorly when used on the complex object 4. We analyse the new network in terms of sensitivity towards: occlusion, colour, distance, angle

4.5 Results

4.6 Conclusion

Chapter 5

Investigating the Trade-Off between Detection Performance and Inference Time

A major drawback of CNNs is their huge computational requirements. For example a state-of-the-art Computer Vision model [14] requires 11.3 billion floating point operations [51]. For a device with computational limitations like an MAV this is prohibitive. Furthermore, a perception system on a MAV usually contains of multiple subsystems. Hence, a fast reaction time can be more important than an accurate detection/outbalanced by the filter etc.

This

The research question of this chapter is stated as:

What are the trade-off's between detection performance m and inference time t when a detection model is integrated on a embedded computing platform?

The question is answered on a theoretical level by using the total number of **Multiply-Adds!** (**Multiply-Adds!**) N_O as an indication for the inference time of the model. However, as also stated by N_O is not necessarily directly related to t. On a computing platform t also depends on:

others

1. whether several operations can be executed in parallel,
2. the memory usage of the operations, the kind of operation e.g. floating point or integer
3. the particular low level implementation of the model

Hence, in addition to N_0 also the actual inference time of the model is measured on a particular computing platform.

The chosen hardware is a Jevois Smart Camera . The platform is developed for vision applications and provides a 4 Core CPU, as well as a small GPU . That's why it is perfectly suitable for integrating in lightweight MAVs or other robotic applications.

jevois

more info

The rest of the chapter is organized as follows: **??** discusses relevant related work. Based on the gained insights **??** formulates several hypotheses to be investigated. **??** outlines the experiments conducted to evaluate the formulated hypotheses. **??** describes the obtained results. **??** discusses the results and answers the research question.

5.1 Related Work

In recent years a lot of research has been conducted to reduce the inference time of CNNs. The publications address different levels for optimization:

1. **Conceptual Level**
2. **Architectural Level**
3. **Operational Level**

5.2 Conceptual Level

On a conceptual level authors aimed to incorporate more steps of the object detection pipeline into one model to share computational load and thus reduce inference time.

overfeat

Overfeat, one the first CNN-based object detectors ran a CNN in sliding window manner across the image. As this led to redundant operations for feature extraction quickly two-stage approaches evolved. The consequent publications of R-CNN, Fast-RCNN and Faster-RCNN proposed a region proposal network that extracts features and proposes possible object locations, followed by a classification network that reuses the extracted features for classification. Thereby not only the number of regions that where classified was reduced but also the extracted features could be reused efficiently.

Yolo and SSD proposed to combine the whole pipeline into one model. Although, this led to a bit of loss in performance, the inference time could be reduced significantly. The aforementioned models are further described in section 4.1.

Using Time domain: [3]

put somewhere the overview of performance vs speed gained from object detection paper

5.3 Architectural Level

Reducing the computational cost of CNNs has been addressed in two individual lines of research.

5.3.1 Architectural Blocks

[26] and [57] showed the performance of thin and deep architectures like *ResNet* with more than 100 layers can equally be achieved by wider but shallower networks. At the same time the proposed *Wide Residual Networks* use less parameters and can be executed more efficiently.

DenseNet [18] proposes the use of dense connections in CNNs. Thereby the input of each convolutional layer does not only consist if its direct previous layer but of a concatenation of the activations of all its previous layers. This enables feature reuse and thus the reduction of the total amount of parameters .

is that really true since we need weights for much more filters

MobileNet [17] and *QuickNet* [11] make extensive use of Depthwise Separable Convolutions (DSCs). DSCs replace the original 3D-convolution by several 2D-convolutions followed by a pointwise convolution. **??** illustrates the concept.

MobileNetV2 [44] further includes linear bottlenecks to reduce the total number of operations.

[58] addresses the computational costs of pointwise convolutions. Instead of applying a pointwise convolution on the whole input volume, group convolutions are applied on by dividing the channels in subsets. These channels are shuffled to enable cross-channel information propagation.

5.3.2 Knowledge Distillation

Knowledge Distillation

5.4 Operational Level

Operational Level - Quantization: [50],

5.5 Experiments

We choose one/two of the above because trying everything is a bit too much. So which one and why?

5.6 Conclusion

Evaluate effects on performance and accuracy

Chapter 6

Method

Can the gained insights be used to build a lightweight and robust detection model for wire frame objects to be applied in autonomous drone racing?

Method from last year, Other papers on drone racing

Describe final method and compare to state of the art in terms of accuracy and speed

Chapter 7

Discussion

answer research questions

Appendix A

Appendix

A.1 Data Generation

This section describes how the ground truth labels are obtained when generating data.

A.1.1 Camera Model

The camera itself is modelled with the pinhole camera model that contains six parameters:

1. Focal length f_x, f_y
2. Central point c_x, c_y
3. Sensor skew s_x, s_y

The model can be summarized in the intrinsic camera matrix C:

$$C = \begin{bmatrix} \frac{fx}{s_x} & 0 & cx \\ 0 & \frac{f_y}{s_y} & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

The model projects 3D coordinates X to the image plane following:

$$X' = CX \tag{A.2}$$

Where X are points described in homogeneous coordinates originating from the cameras position.

For data generation several tools are used. 3D Models for the Target Object (TO) are taken from ... OpenGl is used to render these objects and replace the background with a particular image. The Unreal Engine and AirSim are used to render a full scene.

Within the graphic engines, the objects can be placed in 3D space. From the known object shape the surrounding bounding box can be defined in 3D coordinates. Using the pinhole camera model described in Equation (A.1) the corresponding 2D coordinates on the image plane can be obtained with the following:

The camera position is described by its rotation matrix R and its translation vector t. Where R is obtained from the Euler angles with:

$$R =$$

The 3D coordinates of the objects relative to the camera can be obtained by applying the inverse transformation T of R and t with:

$$t' = R \times t$$

$$T = R^{-1} | -t'$$

$$X_{Cam} = T \times X$$

The full projection can then be expressed by the matrix multiplication:

$$X' = C \times T \times X$$

Where C is the intrinsic camera matrix defined in Equation (A.1).

Bibliography

[1] A. Andreopoulos and J. K. Tsotsos. On Sensor Bias in Experimental Methods for Comparing Interest-Point, Saliency, and Recognition Algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):110–126, jan 2012. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.91. URL http://ieeexplore.ieee.org/document/5765998/.

[2] Alexandra Carlson, Katherine A. Skinner, Ram Vasudevan, and Matthew Johnson-Roberson. Modeling Camera Effects to Improve Deep Vision for Real and Synthetic Data. mar 2018. URL http://arxiv.org/abs/1803.07721.

[3] Kai Chen, Jiaqi Wang, Shuo Yang, Xingcheng Zhang, Yuanjun Xiong, Chen Change Loy, and Dahua Lin. Optimizing Video Object Detection via a Scale-Time Lattice. apr 2018. URL http://arxiv.org/abs/1804.05472.

[4] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain Adaptive Faster R-CNN for Object Detection in the Wild. mar 2018. URL http://arxiv.org/abs/1803.03243.

[5] Chengcheng Ning, Huajun Zhou, Yan Song, and Jinhui Tang. Inception Single Shot MultiBox Detector for object detection. In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 549–554. IEEE, jul 2017. ISBN 978-1-5386-0560-8. doi: 10.1109/ICMEW.2017.8026312. URL http://ieeexplore.ieee.org/document/8026312/.

[6] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.177. URL http://ieeexplore.ieee.org/document/1467360/.

[7] Samuel Dodge and Lina Karam. Understanding How Image Quality Affects Deep Neural Networks. apr 2016. URL http://arxiv.org/abs/1604.04004.

[8] M. Elbanhawi, A. Mohamed, R. Clothier, J.L. Palmer, M. Simic, and S. Watkins. Enabling technologies for autonomous MAV operations. *Progress in Aerospace Sciences*, 91:27–52, may 2017. ISSN 0376-0421. doi: 10.1016/J.PAEROSCI.2017.03.002. URL https://www.sciencedirect.com/science/article/pii/S0376042116300367.

[9] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing using Active Vision. URL http://rpg.ifi.uzh.ch/aggressive{_}flight.html.

[10] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. may 2016. URL http://arxiv.org/abs/1605.06457.

[11] Tapabrata Ghosh. QuickNet: Maximizing Efficiency and Efficacy in Deep Architectures. jan 2017. URL http://arxiv.org/abs/1701.02291.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. nov 2013. URL http://arxiv.org/abs/1311.2524.

[13] Kaiming He and Jian Sun. Convolutional Neural Networks at Constrained Time Cost. URL https://arxiv.org/pdf/1412.1710.pdf.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. dec 2015. URL http://arxiv.org/abs/1512.03385.

[15] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On Pre-Trained Image Features and Synthetic Images for Deep Learning. oct 2017. URL http://arxiv.org/abs/1710.10710.

[16] Andrew G. Howard. Some Improvements on Deep Convolutional Neural Network Based Image Classification. dec 2013. URL http://arxiv.org/abs/1312.5402.

[17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. apr 2017. URL http://arxiv.org/abs/1704.04861.

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. aug 2016. URL http://arxiv.org/abs/1608.06993.

[19] Tadanobu Inoue, Subhajit Chaudhury, Giovanni De Magistris, and Sakyasingha Dasgupta. Transfer learning from synthetic to real images using variational autoencoders for robotic applications. URL https://arxiv.org/pdf/1709.06762.pdf.

[20] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks? oct 2016. URL http://arxiv.org/abs/1610.01983.

[21] Joshua Bateman. China Uses Drones for Earthquake Search and Rescue Missions | WIRED, 2017. URL https://www.wired.com/2017/01/chinas-launching-drones-fight-back-earthquakes/.

[22] Sunggoo Jung, Hanseob Lee, and David Hyunchul Shim. Real Time Embedded System Framework for Autonomous Drone Racing using Deep Learning Techniques. doi: 10.2514/6.2018-2138.

[23] Sunggoo Jung, Sungwook Cho, Dasol Lee, Hanseob Lee, and David Hyunchul Shim. A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge. *Journal of Field Robotics*, 35(1):146–166, jan 2018. ISSN 15564959. doi: 10.1002/rob.21743. URL http://doi.wiley.com/10.1002/rob.21743.

[24] Kate Baggaley. Drones are fighting wildfires in some very surprising ways, 2017. URL https://www.nbcnews.com/mach/science/drones-are-fighting-wildfires-some-very-surprising-ways-ncna820966.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks, 2012. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.

[26] Youngwan Lee, Huien Kim, Eunsoo Park, Xuenan Cui, Hakil Kim, and Communication Engineering. Wide-Residual-Inception Networks for Real-time Object Detection Youngwan. pages 2–8, feb 2016. URL https://arxiv.org/pdf/1702.01243.pdfhttp://arxiv.org/abs/1702.01243.

[27] Guohao Li, Matthias Mueller, Vincent Casser, Neil Smith, Dominik L Michels, and Bernard Ghanem. Teaching UAVs to Race With Observational Imitation Learning. mar 2018. URL https://arxiv.org/pdf/1803.01129.pdfhttp://arxiv.org/abs/1803.01129.

[28] Che-tsung Lin, Patrisia Sherryl Santoso, Shu-ping Chen, Hung-jin Lin, and Shang-hong Lai. Fast Vehicle Detector for Autonomous Driving. . URL http://openaccess.thecvf.com/content{_}ICCV{_}2017{_}workshops/papers/w3/Lin{_}Fast{_}Vehicle{_}Detector{_}ICCV{_}2017{_}paper.pdf.

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. . URL https://github.com/facebookresearch/Detectron.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot MultiBox Detector. URL https://www.cs.unc.edu/{~}wliu/papers/ssd.pdf.

[31] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, nov 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://link.springer.com/10.1023/B:VISI.0000029664.99615.94.

[32] Ratnesh Madaan, Daniel Maturana, and Sebastian Scherer. Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3487–3494. IEEE, sep 2017. ISBN 978-1-5386-2682-5. doi: 10.1109/IROS.2017.8206190. URL http://ieeexplore.ieee.org/document/8206190/.

[33] Abdulghani Mohamed, Kevin Massey, Simon Watkins, and Reece Clothier. The attitude control of fixed-wing MAVS in turbulent environments. *Progress in Aerospace Sciences*, 66:37–48, apr 2014. ISSN 0376-0421. doi: 10.1016/J.PAEROSCI.2013.12.003. URL https://www.sciencedirect.com/science/article/pii/S0376042113000912.

[34] Robin R. Murphy, Satoshi Tadokoro, and Alexander Kleiner. Disaster Robotics. In *Springer Handbook of Robotics*, pages 1577–1604. Springer International Publishing, Cham, 2016. doi: 10.1007/978-3-319-32552-1_60. URL http://link.springer.com/10.1007/978-3-319-32552-1{_}60.

[35] Kuan-Chuan Peng, Ziyan Wu, and Jan Ernst. Zero-Shot Deep Domain Adaptation. jul 2017. URL http://arxiv.org/abs/1707.01922.

[36] Xingchao Peng, Baochen Sun, Karim Ali, and Kate Saenko. Learning Deep Object Detectors from 3D Models. URL http://www.karimali.org/publications/PSAS{_}ICCV15.pdf.

[37] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson Cvap. CNN Features off-the-shelf: an Astounding Baseline for Recognition. URL https://arxiv.org/pdf/1403.6382.pdf.

[38] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. URL https://pjreddie.com/media/files/papers/YOLOv3.pdf.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, Xiangyu Zhang, and Jian Sun. Object Detection Networks on Convolutional Feature Maps. URL https://arxiv.org/pdf/1504.06066.pdf.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. jun 2015. URL http://arxiv.org/abs/1506.01497.

[41] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243. IEEE, jun 2016. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.352. URL http://ieeexplore.ieee.org/document/7780721/.

[42] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. On Rendering Synthetic Images for Training an Object Detector. URL https://arxiv.org/pdf/1411.7911.pdf.

[43] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real Single-Image Flight without a Single Real Image. nov 2016. URL http://arxiv.org/abs/1611.04201.

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. jan 2018. URL http://arxiv.org/abs/1801.04381.

[45] Stephen Shankland. Watch out, Amazon. Zipline's new medical delivery drones go farther, faster - CNET, 2018. URL https://www.cnet.com/news/zipline-new-delivery-drones-fly-medical-supplies-faster-farther/.

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. sep 2014. URL http://arxiv.org/abs/1409.4842.

[47] Christian Szegedy, Scott Reed, Pierre Sermanet, Vincent Vanhoucke, Andrew Rabinovich, Marcel Simon, Erik Rodner, Joachim Denzler, Joseph Redmon, Ali Farhadi, Sergey Ioffe, Christian Szegedy, Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-yang Fu, Alexander C Berg, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, Scott Reed, Pierre Sermanet, Vincent Vanhoucke, Andrew Rabinovich, Jonathon Shlens, Zbigniew Wojna, Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, Kurt Keutzer, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Tianqi Chen, and Carlos Guestrin. YOLO9000: Better, Faster, Stronger. *Data Mining with Decision Trees*, 7(3): 352350, 2016. ISSN 0146-4833. doi: 10.1142/9789812771728_0012. URL https://arxiv.org/abs/1612.08242.

[48] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. mar 2017. URL http://arxiv.org/abs/1703.06907.

[49] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. apr 2018. URL https://arxiv.org/abs/1804.06516.

[50] Subarna Tripathi, Gokce Dane, Byeongkeun Kang, Vasudev Bhaskaran, and Truong Nguyen. LCDet: Low-Complexity Fully-Convolutional Neural Networks for Object Detection in Embedded Systems. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2017-July, pages 411–420, 2017. ISBN 9781538607336. doi: 10.1109/CVPRW.2017.56. URL https://vision.cornell.edu/se3/wp-content/uploads/2017/07/LCDet{_}CVPRW.pdf.

[51] Michael Tschannen, Aran Khanna, and Anima Anandkumar. StrassenNets: Deep Learning with a Multiplication Budget. dec 2017. URL http://arxiv.org/abs/1712.03942.

[52] Gergely Vass and Tamás Perlaki. Applying and removing lens distortion in post production. URL http://www.vassg.hu/pdf/vass{_}gg{_}2003{_}lo.pdf.

[53] P ; Viola and Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. 2004. URL http://www.merl.com.

[54] Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving. URL https://arxiv.org/pdf/1612.01051.pdf.

[55] Wei Xiang, Dong-Qing Zhang, Vassilis Athitsos, and Heather Yu. Context-aware Single-Shot Detector. URL https://pdfs.semanticscholar.org/a299/fd58b86c7d92ac617395b2ada496bc097236.pdf.

[56] Gao Xu, Yongming Zhang, Qixing Zhang, Gaohua Lin, and Jinjun Wang. Domain Adaptation from Synthesis to Reality in Single-model Detector for Video Smoke Detection. sep 2017. URL http://arxiv.org/abs/1709.08142.

[57] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. may 2016. URL http://arxiv.org/abs/1605.07146.

[58] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. jul 2017. URL http://arxiv.org/abs/1707.01083.