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

AP Average Precision

CAD Computer Aided Design

CNN Convolutional Neural Network

DPM Deformable Part Model

DR Domain Randomization

DSC Depthwise Separable Convolution

EWFO Empty Wire Frame Objects

FoV Field of View

FPV First Person View

GPS Global Positioning System

GPU Graphical Processing Unit

HOG Histogram of Oriented Gradients

IR Infrared

IMU Inertial Measurement Unit

IROS International Conference of Intelligent Robots

LIDAR Light Detection And Ranging

MAV Micro-Air Vehicle

mAP Mean Average Precision

NED North-East-Down

IoU Intersection over Union

FCN Fully Convolutional Network

PCA Principal Component Analysis

RPN Region Proposal Network

SIFT Scale Invariant Feature Transform

SSD Single Shot Multibox Detector

SVM Support Vector Machine

TO Target Object

WRN Wide Residual Network

Yolo You only look once

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 [42], 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 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. This can be done by designing an algorithm manually or learning the image processing from 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 [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 network transforms an image of size 224x224 from its input (left) to a task dependent output (right). In this case a classification network predicting 1000 class probabilities is displayed. Each layer applies a non-linear transformation for which the parameters are learned during training. By stacking more layers on top of each other (deepening) and increasing the number of nodes D per layer (widening), highly non-linear functions can be modelled.

Experiments have shown the superior performance of particularly deep/wide models [13, 14, 43, 54]. 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 [48].

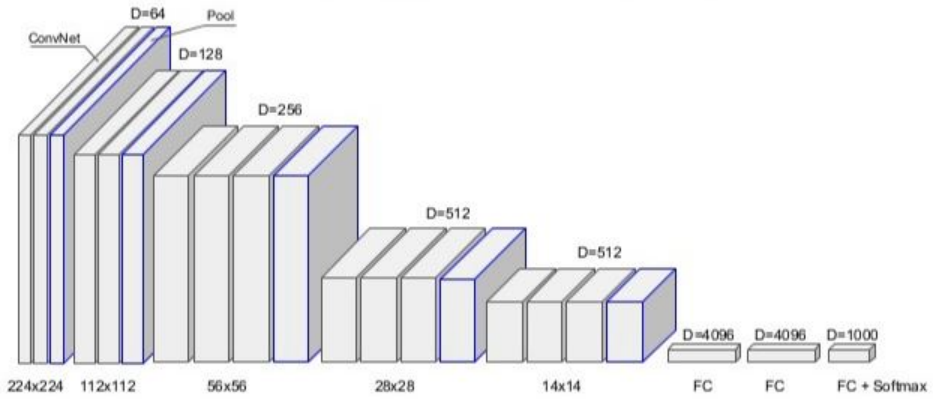

Figure 1.2: Example Architecture of a CNN.

Robotic platforms like MAVs have limited resources in terms of processing power and battery life. Hence, the use of CNNs on such devices is still an open challenge. Research has addressed to reduce the number of computations in Deep Learning models on multiple levels[11, 17, 26, 41, 54, 55]. However, 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.

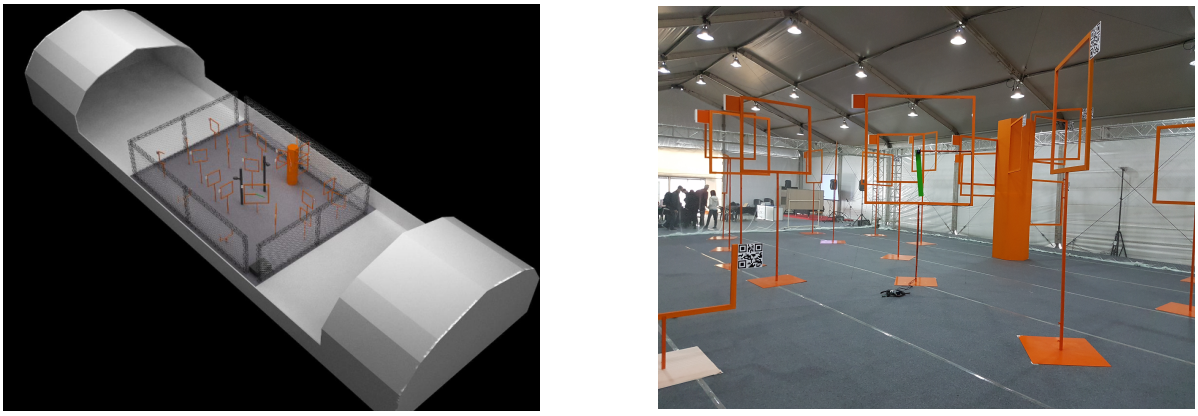

Figure 1.3: Example Images of the IROS 2016 Autonomous Drone Race

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 Wireframe Objects

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

The detection of EWFO is studied in the examples of the IROS drone race gates. These can be seen can be seen in Figure 1.4. The image shows the *Closed Gate* as well as the *Jungle Gate*. Thereby the orange part is considered to be the object of interest. 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 do not study the particular effects of the object shape. This work particularly addresses the implications of the object shape in using a Deep Learning based detection system for EWFO.

A drawback of Deep Learning based vision systems is their need for vast amounts of annotated examples, which is not always available. Racing gates for example are not an object that appears often in everyday life

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

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

and therefore not many example images exist. To this end no publicly available dataset can be used to train a Computer Vision system for EWFO. Since a large part of the object consists of background, 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 (Overfitting).

Figure 1.5: Example of the Cyberzoo dataset. On the left an image while the MAV is hovering, on the right an image during a turn manoevre.

In Chapter 1 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 appear as circular shape. Furthermore, 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 object appearance.

While it is possible to remove lens and sensor effects in post-processing, this can lead to information loss and requires on-board resources. Instead it is computationally more efficient to perform the detection on the raw image data. However, sensor effects have been shown to significantly influence the performance of neural networks [1, 7]. Furthermore, they can lead to varying object appearance on different MAVs. This further complicates the collection of annotated examples.

Another option is the artificial generation of data. By synthetically generating samples with corresponding labels, the theoretical amount of training data is infinite. Moreover, the generation allows to incorporate domain specific properties such as motion blur or image distortion. Hence, data generation is particularly

useful for the detection of MAVs on EWFOs where a large variety of backgrounds is required while samples are difficult to obtain. 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.

Yet introduces 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 is computationally intense. Despite advances in Computer Graphics can virtual environments not yet fully capture the real world. Hence, this work investigates the use of data generation in order to detect EWFOs on MAVs.

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. Moreover, the vision system is part of a greater state estimation and control system which also includes further sensor measurements. Depending on the remaining part of the system, faster and less accurate detections can be more useful than slow but accurate detections. Hence, the trade-off between accuracy and inference speed is of particular interest for this application and is addressed in this work.

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 we learn a CNNs to detect EWFO on MAVs using synthetic data?

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

RQ2 What kind of architecture is suitable to detect 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

The thesis is structured as displayed in Figure 1.6. Chapter 2 describes the metrics and systems used for evaluation. **??**, Chapter 4, **??** and **??** 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. **??** 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

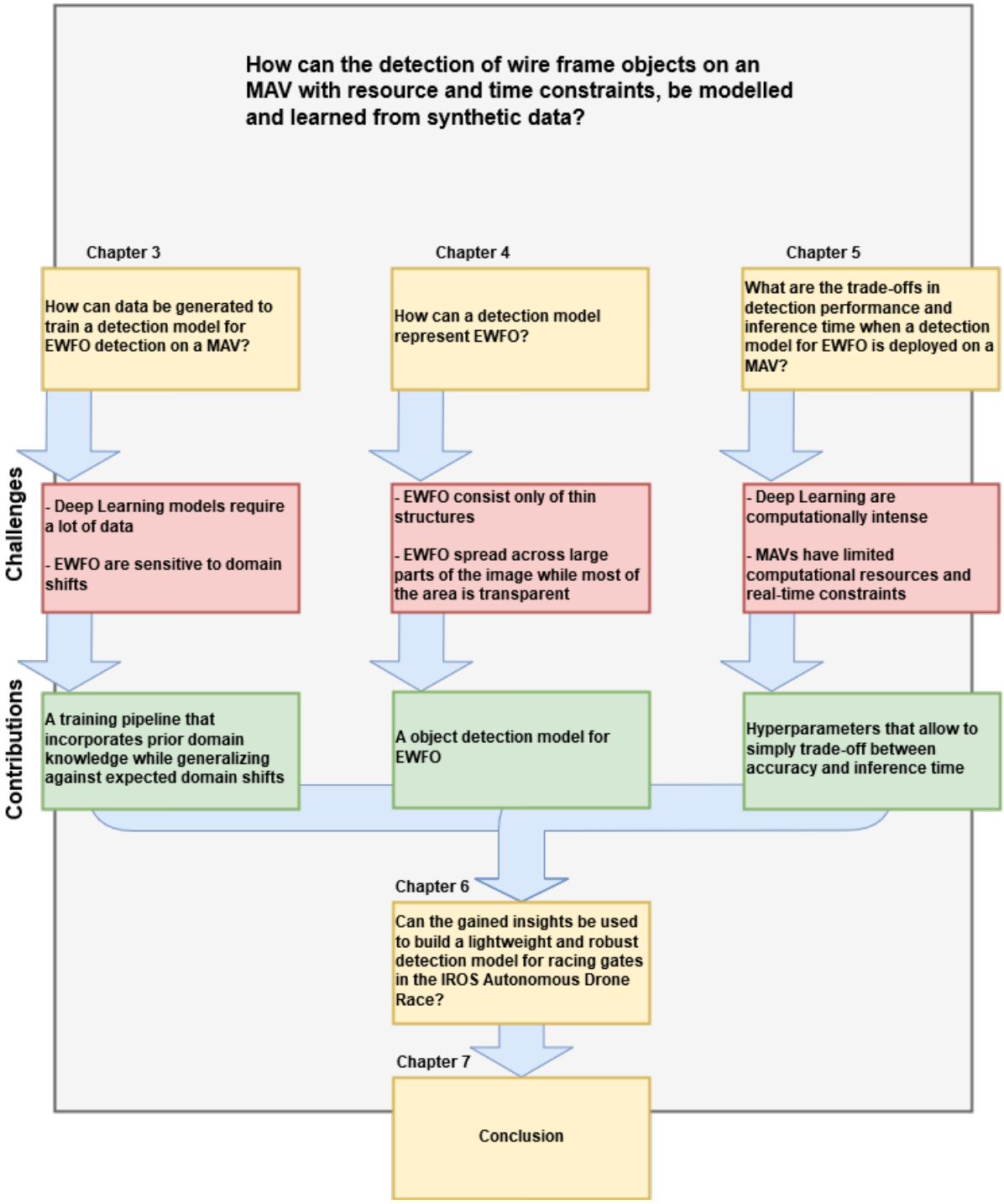

Figure 1.6: Thesis Outline

application for EWFOs. **??** 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. **??** 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. **??** discusses the overall results and formulates a conclusion.

Chapter 2

Background

This chapter describes background knowledge required to understand the remaining parts of the thesis. It introduces the target system for this work as well as datasets and metrics used for evaluation. Furthermore, it discusses related work in Object Detection and Data Generation.

2.1 System Overview

A MAV consist of multiple components of Software that are responsible for higher and lower level tasks. Figure 2.1 illustrates these components in the example of the target platform of this thesis. On the lowest level drivers read out sensors such as the camera and an IMU or communicate with a ground station. A low level control loop is responsible for controlling the local state of the MAV such as altitude and attitude. A higher level control loop controls the global state of the MAV which is the position and the flying trajectory.

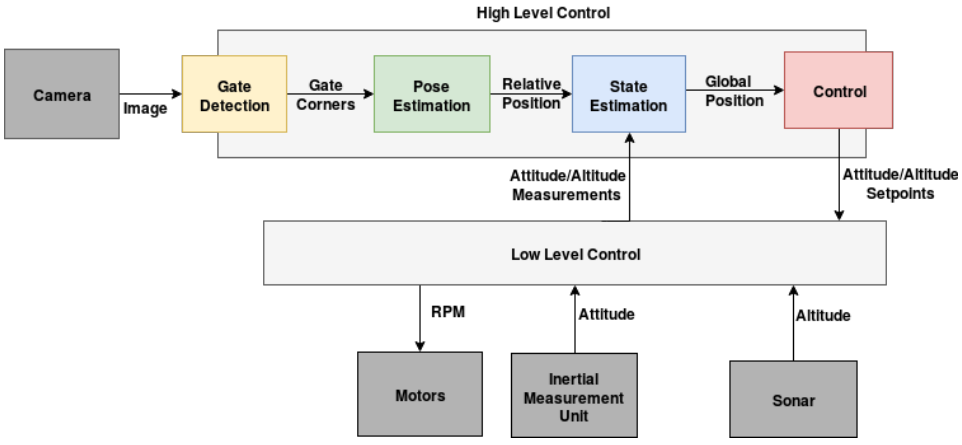

Figure 2.1: Control Loop

The high level control loop of this work is described in further detail. A first step detects the racing gate and yields the corner coordinates. These are used to estimate the relative position of the MAV towards the gate. In the second step the visual measurements are fused with measurements of other sensors. In this case IMU and a sonar deliver altitude and attitude data. This step yields a global position estimate of the MAV. Combined with prior knowledge about the race court, the desired attitude and altitude required to fly the trajectory is calculated. The results are send as set points to the low level controller.

The hardware platform used to run the high level control loop is the *JeVois* smart camera. It contains a 1.3 MP camera with 65 degree field of view. The processing units are a quad core ARM Cortex A7 processor with 1.35 GHz and a dual core MALI-400 GPU with 233 Mhz. In order to extent the field of view a 120 degree wide angle lens is mounted. In Figure 2.2 the camera is shown.

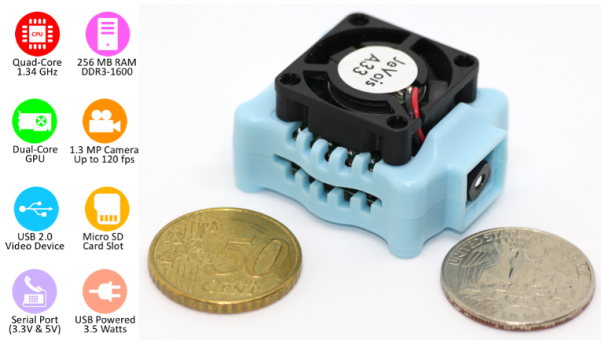

Figure 2.2: JeVois Camera

2.1.1 Baseline Algorithm

Within this work the first step of the aforementioned pipeline is addressed. It aims to replace the current baseline algorithm *SnakeGate*, a low-level image processing algorithm. Its scheme is summarized in and described in the following.

1. Filter image by colour threshold
2. Sample stochastically
3. Follow the pixels horizontally as long as they are within the colour threshold otherwise return to 2.
4. If a bar of sufficient length has been found repeat 3. vertically along one end of the line found in 3.
5. If a vertical bar is found the square is considered as gate candidate
6. Create local histogram around the corners of the gate candidate and choose the highest peak as gate corner.
7. Count the fraction of pixels within the color threshold in relation to the total number of pixels a long all edges of the gate candidate to determine the *color fitness*.
8. Gate candidates that exceed a chosen threshold are considered valid detections.

this can be described more formally

SnakeGate is fast to execute but lacks from several drawbacks. (1) it is colour dependent and thus very subtle to light changes. Hence it must be fine tuned according to light conditions in a certain room. Strong colour variations across the object cannot be handled by the method; (2) the method requires the full object to be visible; (3) the method cannot exploit context such as the pole of an object. These drawbacks motivate a learning based approach to be investigated in this thesis.

2.2 Datasets

A dataset has been recorded to serve as a benchmark for the developed methods. The dataset consists of 300 images recorded with the JeVois camera during flight and while remaining on ground. The samples stem

Figure 2.3: Examples of the three test domains. From left to right: *Basement*, *Cyberzoo* and *Hallway*

from three different rooms with varying light conditions. The rooms are referred to as *Basement*, *Cyberzoo* and *Hallway*. Example images for each room can be seen in Figure 2.3.

All environments are indoor scenes which are a typical examples for GPS-denied areas, where vision based state estimation is required. The scenes contain two gates that are arranged in varying order. Hence up to two objects are visible and can overlap which means the gate farer away can be seen through the closer gate. Each of the rooms has different environmental conditions:

1. *Basement* is a bright environment illuminated by artificial light sources. The corridor in which the objects of interest are placed are narrow while also objects and persons are visible on the samples. The dataset contains 163 samples with 312 objects in total.
2. *Cyberzoo* is taken from a test environment for MAV flights. External light sources are covered such that an even illumination and dark background is created. Only in a small subset of images distractors like other objects or persons are visible. In total 88 samples stem from this room while 71 objects are present.
3. *Hallway* is a bright environment illuminated by a combination of artificial light sources as well as daylight that shines through the windows. The samples are taken with the windows as background. This leads to a very bright background such that the thin structure of the objects are hardly visible. The dataset contains 49 samples with a total of 86 objects.

2.3 Evaluation Metrics

The detection performance is evaluated in terms of precision and recall. These metrics are defined as:

Precision

$$p = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall

$$r = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Where true positives are objects that are detected, false positives are detections although there is no object and false negatives are objects which have not been detected.

Hence, recall expresses how many of all objects are detected and therefore how complete the result is. Precision measures how many of the predicted objects are actually correct detections.

A correct detection is determined based on its overlap with a ground truth box. This is measured by the relation of Intersection over Union (IoU). In experiments we determine 0.6 as sufficient overlap for a detection.

The model used within this thesis associates a "confidence" value with each prediction that can trade off precision and recall. This is further explained in Chapter 4. By accepting more detections with a lower confidence threshold, the probability increases that one of the predictions is a true positive. Hence, it increases recall. However, it also increases the probability of false positives and thus lowers precision. In order to evaluate this trade-off, precision is plotted over recall at increasing confidence values.

As the learning of CNNs is stochastic, the mean across several trainings is reported. In order to determine the average precision recall trade-off the precision is interpolated across evenly distributed recall levels between 0 and 1 using:

$$p_{\text{interp}}(r) = \max_{r' \geq r} p(r')$$

Subsequently the mean at all recall levels can be calculated. A metric that combines the precision-recall trade-off is Average Precision (AP):

$$ap = \int_r p_{interp}(r)dr$$

There should be a source for this

We denote precision, recall and average precision at a certain IoU threshold such as 60% as $p_{60}, r_{60} and ap_{60}$.

2.4 Related Work

This section describes previous work related to this thesis. Covered are methods to detect objects, reduce the inference time of CNNs and methods to train detectors in a simulated environment.

2.4.1 Detecting Objects

Object Detection is one of the central domains in Computer Vision. Giving a full review of all approaches is beyond the scope of this work. We mention some of the most important milestones relevant for this work.

On a high level 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. Performing detection in this space is difficult, also because the performance of 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 2.4.

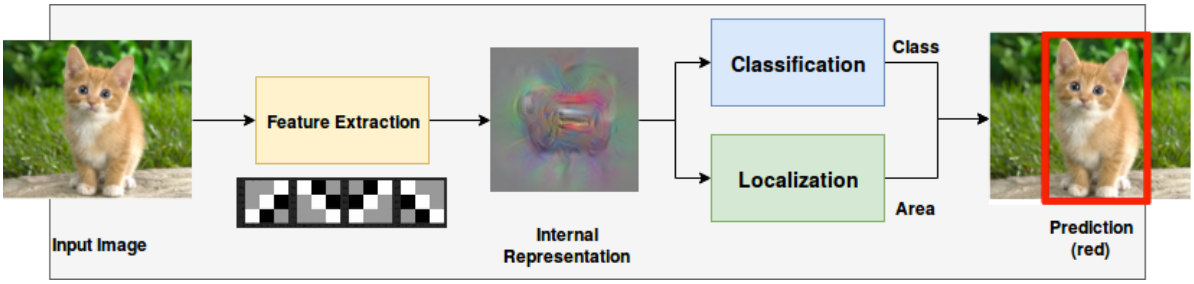

Figure 2.4: Object Detection Pipeline.

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

An efficient feature extraction stage is thereby crucial for the success of an Object Detection pipeline. If the inferred representation is clearly separable, a simple classification stage can distinguish an object from the background. In contrast, even a flexible classifier cannot separate a highly overlapping feature space.

Traditional Methods

The early attempts to Object Detection define objects in terms of basic volumetric shapes such as cubes and cylinders. During inference these features are extracted and compared to a database. However, in practice even recognizing these basic shapes proves to be difficult [**?**].

Later approaches focus more on appearance based features such as wavelets [**?**] which also applied in [50] for human face detection. Thereby the image is processed by a cascade of classifiers using a sliding window in multiple scales. The processing of an image patch is stopped when a classifier assigns background to that patch. The features can be computed with simple operations and thus the detector can be executed extremely fast. However, the used Haar-wavelets cannot efficiently encode large textures making the approach less suitable for more complex objects [**?**].

In contrast Histogram of Oriented Gradients (HOG) [6] and Scale Invariant Feature Transform (SIFT) [31] use the image gradient to cover shape information. In a sliding window local histograms based on the gradient orientation are calculated. Dalal and Triggs [6] use the feature for pedestrian detection.

A general challenge in Computer Vision is the combination of local image features such as corners and edges to a more global detection of an object. Especially, when parts of the object can be occluded or deformed and thus undergo large variations in appearance. In order to cope with these issues Felzenszwalb et al. [10] model pedestrians in individual parts and combine them in their proposed Deformable Part Model (DPM).

Traditional methods have in common that the individual steps of the detection pipeline are optimized separately. Furthermore, often the feature extractors and object models are designed manually. Hence, designing such a detector can result in cumbersome application dependent work. EWFO have sparse features and cameras on MAV can have a strong influence on the object appearance. Modelling this appearances manually is difficult. Also, the methods seem to have reached a limit in performance in the years of 2005-2012 where almost no improvement in performance was achieved.

CNNs-based Feature Extraction

A breakthrough in detection performance came with CNNs which emerged from Deep Learning research and subsequently 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 allows 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 2.4.1 displays these parameters and introduces additional terminology necessary for the remaining parts of this chapter. The *kernel size* $\mathbf{k}$ 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.

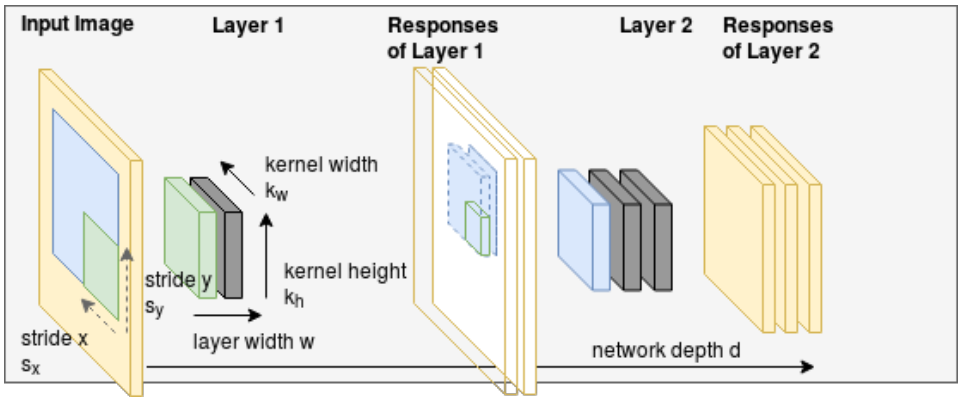

Figure 2.5: Example Architecture of a CNN

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. [43] achieve similar performance using a network with 22 layers. The proposed network included an *Inception*-module, an architectural element that allows deeper networks at a constant computational budget.

Residual Connections. An issue that prevented training even deeper networks is the *vanishing gradient problem*. As the gradient distributes across nodes its magnitude gets smaller with increasing amount of nodes. 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.

Wide-Residual Networks. However, later work by Zagoruyko and Komodakis [54] 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.

Fully Convolution Networks. 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 con-

nected layer in the last stage, these networks only apply local operations. This saves many computations in the last layer and enables the application of models on various input sizes.

Dilated Convolutions. However, FCN in combination with a small amount of layers introduce a limited receptive field. If the network is too shallow the last layer cannot take into account the whole input image. A way to increase the receptive field without increasing the number of computations is the use of sparse kernels, also called Atrous/Dilated convolutions.

Depthwise Separable Convolutions. Another line of research to reduce the number of computations in CNNs address the convolution operation. *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. This reduces the total number of operations from $N = k_w \cdot k_h \cdot w_n \cdot w_{n+1}$ to $N = (k_w \cdot k_h + w_n) + w_n \cdot w_{n+1}$. *MobileNetV2* [41] further includes linear bottlenecks to reduce the total number of operations. These are convolutions with a 1by1 kernel and linear activation.

Channel Shuffling. [55] 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.

Dense Connections. *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. By enabling feature reuse the total amount of parameters shall be reduced.

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 vast amounts of 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.

CNN-based Object Detection

After showing promising results for Classification, CNNs where quickly also applied for Object Detection.

Two-Stage Detectors. Girshick et al. [12] use Selective Search [**?**] 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. Thereby, 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 2.6 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 $c_1..c_p$.

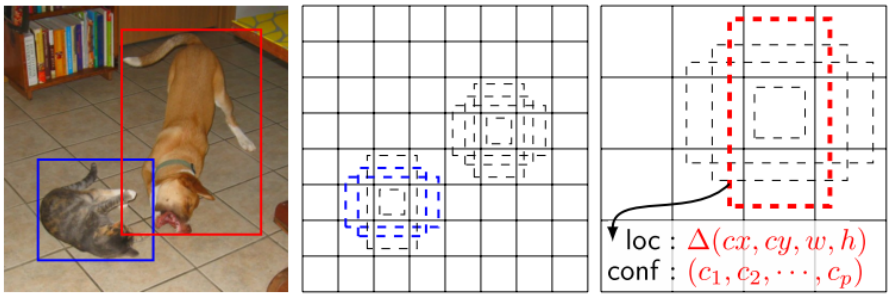

Figure 2.6: Visualization of the anchor box concept [30].

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 213x 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.

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.

Being currently the method with the best performance in terms of Mean Average Precision (mAP) two stage approaches would be a valid choice for the detection of EWFO. However, their two stage character makes the inference time relatively slow, which is not suitable for the application on a MAV. Also, this work investigates the detection of a single object. For this application the RPN can be used directly.

One-Stage Detectors. An alternative approach to CNN-based Object detection are one-stage detectors. Here, the network performs Classification and Localization in a single pass.

The first method was published by Szegedy et al. in [44]. 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. Being a breakthrough as the first one stage detector this approach was still limited as for each grid cell at most one object could be predicted.

Liu et al. propose Single Shot Multibox Detector (SSD) [30], a 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, 44] 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.

CornerNet

One stage detectors achieve generally a lower performance in terms of mAP than two stage detectors. However, they are significantly faster to evaluate and generally more accurate than traditional methods. The TinyYoloV3 architecture is 9-layer network with a suitable size to be applied on a MAV. This work uses this approach as the baseline model.

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

Attention Models

The methods described so far share the sliding window paradigm. The whole input image is processed and subsequent pooling operations reduce the spatial dimension. Hence, the computational complexity scales linearly with the number of pixels. In contrast attention based models only process subregions of the image. For example XX use a recurrent neural network to process a certain amount of patches of the input image. Which patches are evaluated is decided by the model based on the patches seen until then. This enables to deploy a method where computational complexity can be controlled independently of the image size.

google deep mind

Guido proposes a neural network to learn an attention model.

Other attention mechanisms aim to increase performance by ..

Attention mechanism can not yet compete with state-of-the art CNN based object detectors.

Using Time domain: [3]

Knowledge Distillation

write

Weight Quantization

The hardware of most embedded **CPU!** (**CPU!**) support only integer operations and thus rely on software to perform floating point operations. Hence, weight quantization is another line of research to reduce the inference time of deep networks on mobile devices. By mimicking the quantization effects during training, the network learns to deal with these kind of artefacts. However, the target platform of this work supports hardware accelerated floating point multiplications. While weight quantization could still be used to accelerate the network it would require serious low level operation optimization. This work addresses the optimization on a more high level architectural basis.

put some refs where this is applied [47]

2.4.2 Generating Data

Related 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.

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. [44] 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.

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% mAP70) 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.

Augmenting Existing Images with CAD - Models

In order to create new view points Computer Aided Design (CAD)-models can be used. These models describe 3D-shape of an object and can be placed on existing images to augment or increase 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 image cues like texture, colour and background affects the detection 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.

Fully Synthesizing Environments

A more realistic placement of objects can be achieved when fully synthesizing environments. 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 details of real world objects, the generated data might look too artificial.

Johnson-Roberson et al. [20] use a powerful graphical engine and a highly detailed environment to 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.

In order to create realistic environments intense manual work is required for the design. In contrast [40, 45, 46] 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.

This work addresses the generation of data for the detection of EWFO on MAVs in GPS denied scenarios. Such scenarios cover a wide range of possible environmental conditions and the images taken from MAV cameras are peculiar. Hence the creation of a full environment is investigated in this work.

2.4.3 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] [53].

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.

Generative Adversarial Networks

write [19]

Chapter 3

Synthesizing Data for the Detection of EWFO on MAVs

This chapter addresses the generation of data for the detection of EWFO. In particular the following research question is addressed:
How can data be generated to train a detection model for EWFO detection on a MAVs?

The question is subdivided in three research questions to be addressed in this chapter:

RQ1.1 What role does context play for the detection EWFOs on MAVs?

RQ1.2 What role does the view perspective play for the detection EWFOs on MAVs?

RQ1.3 Can the incorporation of sensor effects or image augmentation benefit the detection performance?

The chapter starts by introducing terminology and the data generation pipeline. In the later sections each research question is addressed in several experiments. The chapter finishes with an overall conclusion.

3.1 Methodology

In order to to investigate the research questions a data generation pipeline is implemented using OpenGL, UnrealEngine and AirSim. An overview can be seen in Figure 3.1. In the first step a scene is created in which the objects of interest as well as the camera are placed. In 3D space position and orientation (pose) of each object are determined by translation $\mathbf{t}$ and rotation $\mathbf{r}$. The coordinate system is North-East-Down (NED).

A view projection yields an image through the lens of the camera. The coordinates of each point in 3D space are projected on the 2D image plane. A final post processing step can simulate further effects like lens distortion and sensor noise. This step is implemented using OpenCV and Python.

All source code is made publicly available at `https://github.com/phildue/datagen.git`.

Within the pipeline environments for training and testing are created. A black environment serves as base to replace the background with existing images. Furthermore, three indoor base environments are created that fully simulate illumination and background. An overview can be seen in Figure 3.2. Within the environment light conditions, background textures, object locations can be changed manually. The environments are described in the following:

1. *Dark:* The environment is a room without windows, only containing artificial light sources.

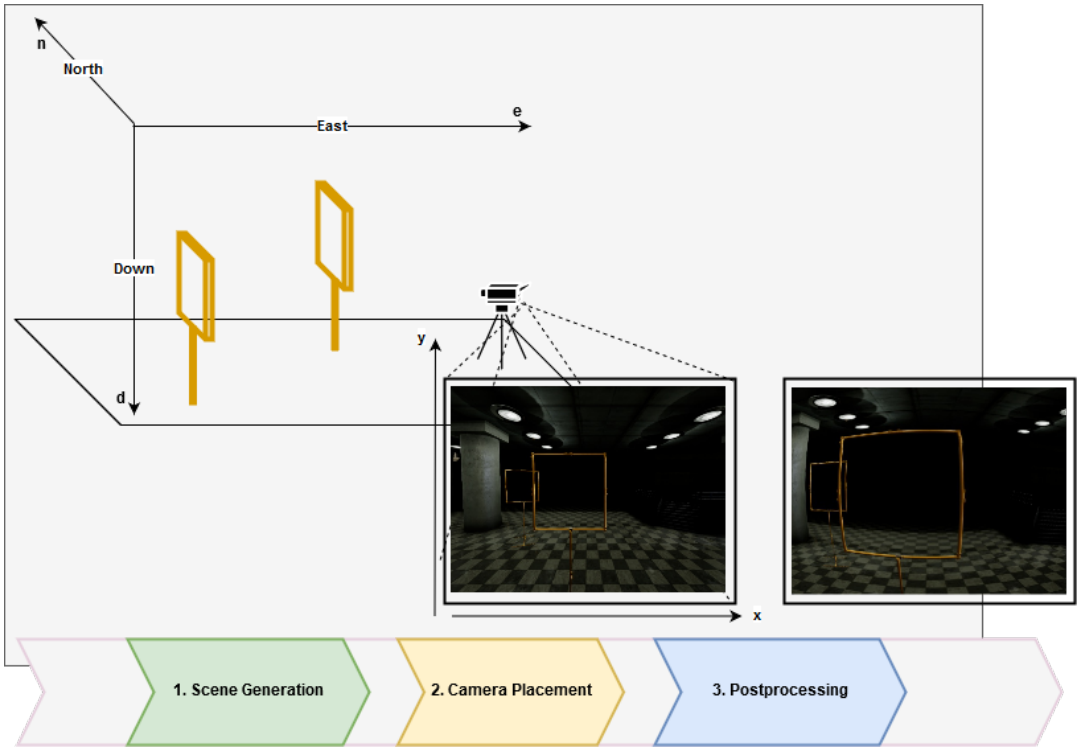

Figure 3.1: Overview of the data generation process.

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. *IROS:* 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 3.2: The environments from left to right *Dark*, *Daylight*, *IROS2018*

The data generation pipeline is used to generate a test set in simulation. The test set is set up in the *IROS* environment and inspired by the race court of the IROS Autonomous Drone Race. The AirSim flight controller is used to simulate a flight of an MAV through the race court. The test set contains 550 images with a total of 1361 objects.

It should be noted that when generating data with the described methods it is likely that samples appear in which only a corner of the object is visible or the view angle is in such a way that the object appears only as a thin line. In initial experiments these corner cases led to unstable training. Hence, some minimum requirements for the labels are set and samples/labels are removed if these are not met. Objects have to have a minimum size of 1% of the image area as well as an aspect ratio between 1/3 and 3/1. Furthermore, at least three corners have to be visible.

The models and training are implemented using the *Keras* framework with *tensorflow*-backend. For all train-

ings the Adam optimizer is applied using a learning rate of 0.001 for the first 60 epochs and a learning rate 0.0001 afterwards. A validation set containing 1% randomly sampled images from the training set is used. The training is stopped if the validation error does not decrease for more than 5 epochs or 100 epochs are reached.

Throughout the experiments the baseline TinyYoloV3 architecture is used. Thereby we simplify the loss function to a single class prediction. The exact model is described in Chapter 4. The input image resolution is 416x416.

3.2 Context

The context in which an object is placed consists of background as well as light conditions or other objects. It is determined in the first step of the data generation process. Theoretically CNNs can learn to exploit context for detection. Yet, the actual influence is not clear. For example the Object Detectors trained in [36] do not seem to make use of this image cue. Nevertheless, we investigate the importance of context for the detection of EWFOs. We hypothesize that due to the sparsity of their features and their emptiness context and background plays a more important role than for solid objects.

For the investigation we compare three methods to generate data: The placement of CAD-models on uniformly coloured backgrounds, on backgrounds taken from a dataset as well as from fully synthesized environments. In terms of geometric and physical properties the fully synthesized environments provide the most realism and hence should give the best results. However, as the graphical engine only contains a limited set of textures, the variance in background is way smaller than in the backgrounds taken from a dataset. Furthermore, the objects in the background are also rendered. Hence, when tested on real data this method could lead to better performance.

3.2.1 Experiments

Three datasets are generated with 20 000 samples each. Dataset I contains samples generated by placing CAD-models on uniformly coloured backgrounds; Dataset II contains samples generated by placing CAD-models on backgrounds taken from the Pascal VOC dataset, Dataset III contains samples generated by simulating a full environment. A fourth model is trained on a combination of Dataset II and III (equal proportion).

Within the environment the camera is placed at random locations following these distributions:

$$x = \mathcal{U}(-30,30), \quad y = \mathcal{U}(-20,20), \quad z = \mathcal{N}(-4.5,0.5)), \quad \phi = \mathcal{U}(0,0.1\pi), \quad \theta = \mathcal{U}(0,0.1\pi), \quad \psi = \mathcal{N}(-\pi,\pi) \tag{3.1}$$

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

The models are tested on the real datasets described in Section 2.2 as well as the simulated flight described in Section 3.1.

3.2.2 Results

Figure 3.3 display the results. It can be seen how using uniformly coloured backgrounds achieve almost no detection. Only on the simulated data a few objects could be detected.

In contrast using fully synthesized data achieves the best performance on the simulated data for low and high quality detections. It can be seen that almost all detections have an IoU of more than 60% and are thus good quality. On the real data the performance in low quality detections is comparable to the performance on the simulated data. However, in hiqh quality detections the performance drops significantly to an equal level of using real backgrounds.

Using real backgrounds performs poorer than using a fully synthesized environment in almost every case. However, for higher quality detections the performance is competitive to using a fully synthesized environment. Combining both methods does achieve at most the performance of using fully synthesized data. However, in most cases the performance is worse.

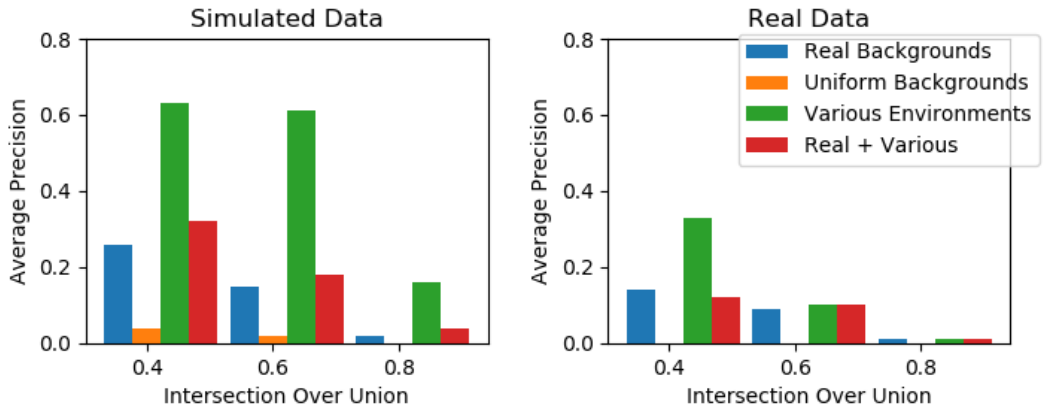

Figure 3.3: Results of different ways to generate context. Synthesizing more parts of the environment improves performance on both test sets. However, this only holds for weakly localized predictions.

3.2.3 Discussion

The results meet our hypothesis that using only uniformly coloured backgrounds is not enough to train an Object Detector for EWFO. Providing more variance in the background is crucial to improve performance.

Despite having not seen a single real background, the model trained on fully synthesized data achieves the best performance on the real data. Hence, it seems that synthesizing correct geometric/physical properties is more important than providing a large variance in background.

However, the margin gets small for higher quality detections. Thus, the simulation of geometric/physical properties does not help for good localization.

3.2.4 Conclusion

We investigated the role context plays for the detection of EWFOs on MAVs. We can conclude that providing some background is crucial for the detection. Otherwise the detector performs poorly in simulation as well as the real world. Furthermore, synthesizing the whole environment benefits the detection on real images. However, most of the additional detections have not very accurate bounding boxes.

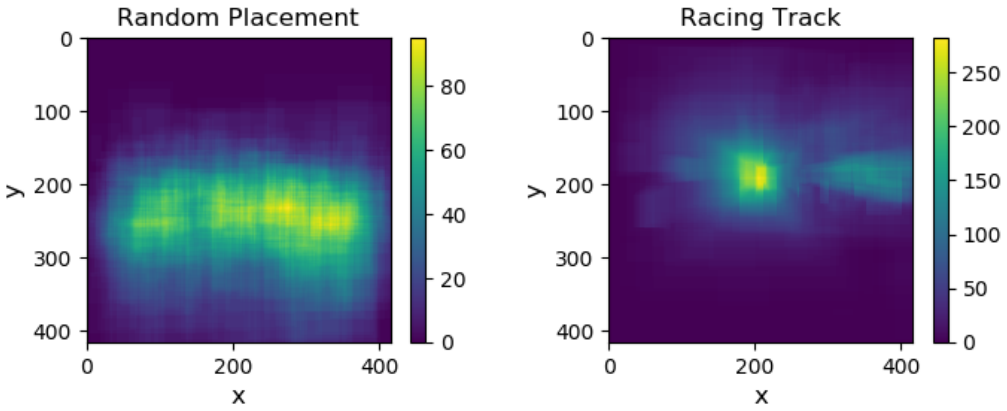

Figure 3.4: Object appearances when generating samples with random poses (left) and during a MAV flight. During the flight the object appears mostly centered on the horizontal line.

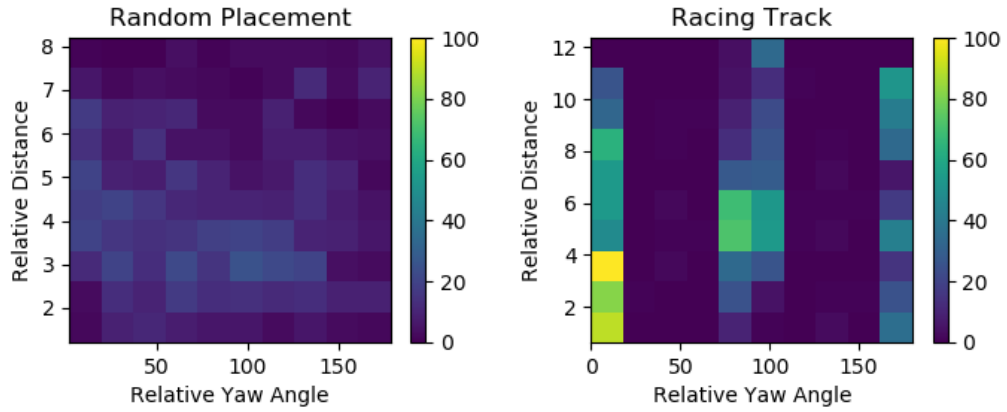

Figure 3.5: Histogram of object occurrences in yaw angle and distance relative to the camera. The random placement does rarely cover facing the object closely and frontally.

3.3 View

Another important property that influences the generated sample is the camera pose. It determines the view on the scene and therefore at which distance, angle and location the objects appear on the image.

A straightforward way is placing the camera randomly (within some margin) in order to cover a large variation of views on the object. That way the network can learn a general object representation and hopefully detect unseen objects from different view points. However, random placement might not resemble the real world sufficiently. An MAV does not appear at random places within a scene, especially not when it follows a racing track. We examine this by simulating a flight through a race court using AirSim's MAV model and analyzing the relative object poses. We compare these to the relative poses obtained when placing the camera randomly using the distributions from Equation (3.1).

Figure 3.4 shows the distribution of bounding boxes when created with random camera placement and when following a racing track. It can be seen how, when following the race track most of the objects are centered and distributed across the horizon, as camera focuses the next object frontally most of the time. In contrast, random placement leads to more evenly distributed object locations. This can also be seen in Figure 3.5 where a 2D histogram of the yaw angle and distance with respect to the camera is displayed. Thereby 0 corresponds to facing the object frontally, 180 degrees facing the object from the back. It is apparent how the

random placement covers a much larger range of relative angles, while in the racing track certain angles do not appear at all. Even more importantly the largest bins of the racing track is an angle of 0 and a distance between 0m and 4m. These bins are almost not present when placing the camera randomly. This is because close to the camera the field of view is small, while the area of the object faced frontally is big. Hence, the probability of an object ending up at this specific location is relatively low. Furthermore, when placing the camera randomly there are no samples further away than 8m. This is because in the race track the camera traverses the room from one end - where it can see almost all gates - to another. The probability that the randomly placed camera ends up in a similar position is relatively low.

The filters of CNNs are translation invariant by design but cannot inherently handle variation in rotation and scale. We hypothesize that the generation of samples with only one of the two methods can miss important object appearances. Random placement does not cover appearances that are typical for a autonomous drone race. On the other hand, only training on racing tracks might lead to a bias towards the created courses.

3.3.1 Experiments

In order to to evaluate this hypothesis three models are trained on 20 000 samples each. Model I is trained when placing the camera randomly, following the distribution in Equation (3.1). Model II is trained on varying racing courts. Model III is trained on a combination of both datasets (equal proportion).

The models are tested on the real datasets described in Section 2.2 as well as the simulated flight described in Section 3.1.

3.3.2 Results

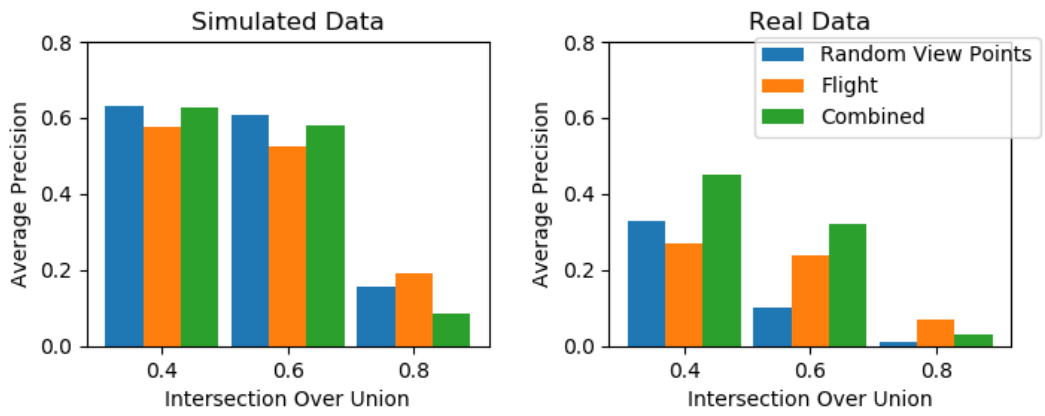

Figure 3.6: Average Precision for the three different models.

Figure 3.6 shows the average precision of the different models. It should be noted that *Random View Points* is the same model as *Various Environments* of Figure 3.3. It can be seen how Model I and Model II perform comparable on the simulated dataset. In contrast on the real dataset the performance in high quality detections is significantly better when training the network on flight images.

Combining both datasets improves the performance mainly on lower quality detections. This holds particularly for low detections with lower bounding box accuracy.

combined

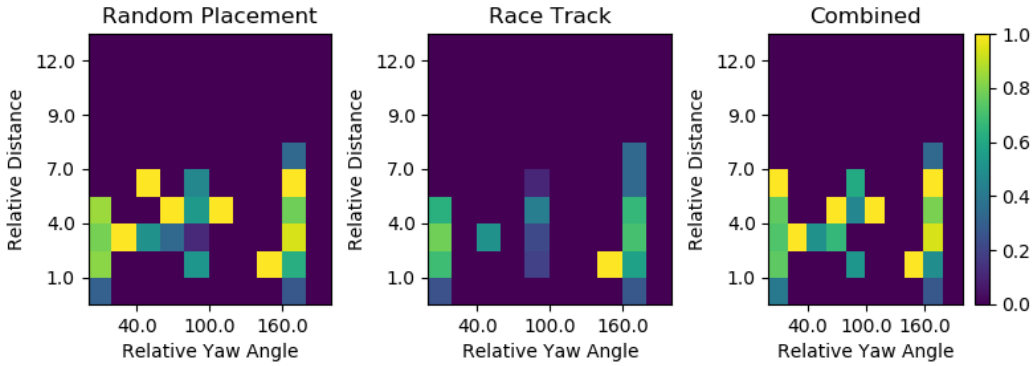

Figure 3.7: Recall on clusters depending on yaw angle and distance.

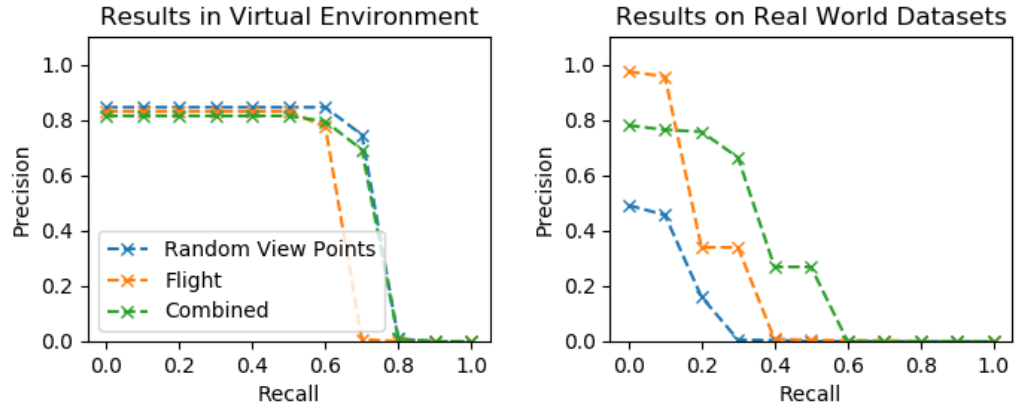

Figure 3.8: Precision - Recall at IoU 0.6.

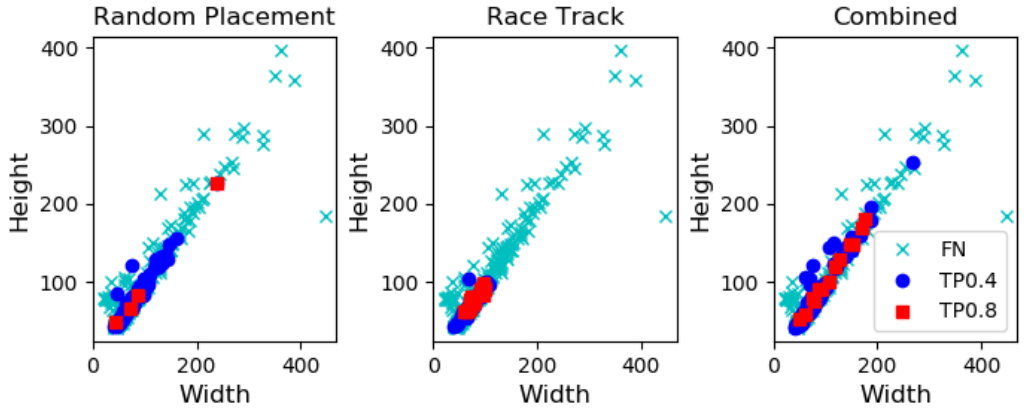

Figure 3.9: Detections on the real data shown with respect to width and height of the bounding box. The more top left a sample is located the closer it is to the camera. When Height > Width the object mostly has a yaw angle unequal to zero.

3.3.3 Discussion

3.3.4 Conclusion

3.4 Image Augmentation

The final step applies low-level image transformations. It allows to further simulate sensor effects and increase the variance in the generated data.

In literature [16, 25, 30, 44] the application of image augmentation is a common tool to improve the detection performance. The experiments in [2] show how the incorporation of sensor effects particularly improves the performance of models learned on fully synthesized data. In the MAV domain sensor and lens effects have a significant influence on the obtained sample. Hence, we hypothesize that the incorporation of these effects will improve the performance of the trained models. A total of five image augmentation techniques are investigated.

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.

For this work a 120 wide angle lens is used. Especially, close objects undergo a strong variation in appearance. Hence, we hypothesize including this effect in the training process will improve the performance on the real world dataset.

The effect is applied using the model for wide-angle lenses from [49]. It models the removal of lens distortion as combination of radial and non-radial part, that is approximated with a second order Taylor expansion:

double check formula is (y in first line?)

$$\begin{pmatrix} x_u \\ y_u \end{pmatrix} = f(x,y) = \begin{pmatrix} x(1+\kappa_1 x^2 + \kappa_1(1+\lambda_x)y^2 + \kappa_2(x^2+y^2)^2) \\ y(1+\kappa_1 x^2 + \kappa_1(1+\lambda_y)y^2 + \kappa_2(x^2+y^2)^2) \end{pmatrix} \tag{3.2}$$

Where:

- x_u and y_u are the undistorted coordinates.
- κ_1 κ_2 control the radial distortion
- λ_x and λ_y control the tangential distortion

Applying the lens distortion to an image is done using the inverse of Equation (3.2). However, as there is no closed form solution, so the Newton-approximation.

An example with $\kappa_1 = 0.5, \kappa_2 = 0.5$ is displayed in Figure 3.12. 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.

In [2] including chromatic aberration significantly improves the performance of models that are trained on fully synthesized data. Hence, we hypothesizes this will also help for our work.

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:

$$f(x_C, y_C) = \begin{pmatrix} S & 0 & t_x \\ 0 & S & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \\ 1 \end{pmatrix} \tag{3.3}$$

Where C is one colour channel of the image.

An example is displayed in Figure 3.11. It can be seen how the red and green channel are shifted relative to each other. Thus two bars appear in the image.

Blur Fast movement and sensor noise can lead to blurry images. This is particularly present in the domain of MAV/Autonomous Drone Racing. Hence, we hypothesize including this effect will improve the detection in the real world.

The effect is modelled using a Gaussian-filter with:

$$k = \frac{1}{2\sigma_x \sigma_y \pi} e^{-\frac{x^2+y^2}{2\sigma_x \sigma_y}} \tag{3.4}$$

double check notation

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]:

$$f(S) = \frac{255}{1 + e^{-AS}} \tag{3.5}$$

where A is a constant term for contrast and S the exposure. The image can be re-exposed with: whats a

$$I' = f(S + \Delta S) \tag{3.6}$$

where S is obtained from :

$$S = f^{-1}(I) \tag{3.7}$$

An example for overexposure is displayed in Figure 3.15. It can be seen how lighter areas appear particularly light, while dark areas remain dark.

Color Variations describe hsv scaling

3.4.1 Experiments

Distortion

3.4.2 Results

Figure 3.16 shows the results in terms of average precision in the training domain. On the simulated data there is only a minor influence on the data augmentation. On the real data blurring improves the detection significantly. In contrast image augmentation with chromatic aberration or variations in exposure/hsv leads to worse results than using no image augmentation at all.

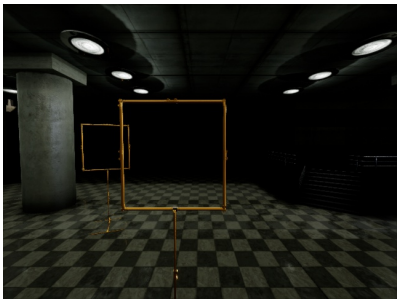

Figure 3.10: Original Image.

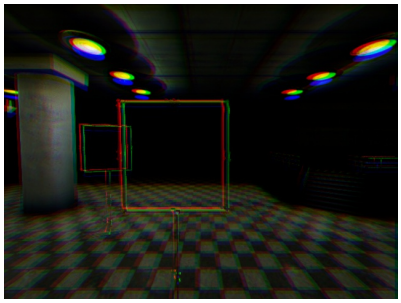

Figure 3.11: Chromatic Aberration.

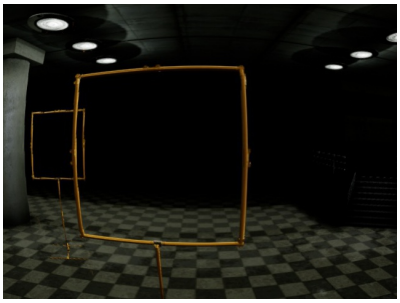

Figure 3.12: Lens Distortion.

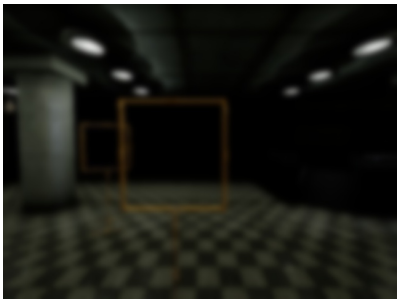

Figure 3.13: Out-of-Focus blur.

Figure 3.14: Vertical Motion Blur.

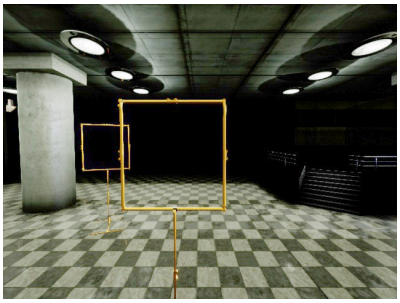

Figure 3.15: Exposure.

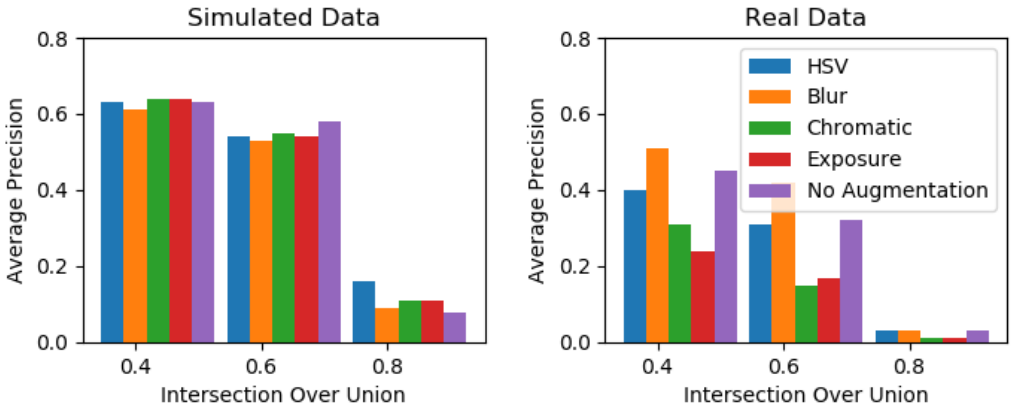

Figure 3.16: Performance in terms of Average Precision for different methods of image augmentation.

3.4.3 Discussion

In simulation where the colours are similar to the training data and no blurring is present, the image augmentation has only little influence. Hence, it seems instead of getting confused by the added noise, the models actually learn to generalize and achieve similar performance.

On the real data there is a stronger effect measurable. In contrast to our hypothesis including more sensor noise in the training does only improve the detection in one case. For the variations in HSV the result is especially surprising as the object colour in training and test data are slightly different. Yet, including more variation in colour and illumination does not help the prediction. Chromatic Aberration led to a significant improvement in [2] however, we cannot confirm these results. It is possible that our camera suffers only little from chromatic aberration and thus including the effect in the training does not further help the prediction. The same holds for variations in exposure. Motion and sensor blur are effects that are clearly visible in the dataset. Here we can confirm that including the effect in the training process improves the performance in the real world.

3.4.4 Conclusion

We investigated whether including sensor effects present in the target domain in the data generation process can improve the detection. Therefore we modelled several effects that were observed when working with the camera or that improved the detection in experiments conducted in literature. Finally, only image blurring improved the detection. Other effects led to a deterioration in performance.

We also investigated image augmentation by adding variations in HSV-space to evaluated whether this improves the robustness of the model, improving the performance on the real world dataset. However, the image augmentation did not lead to an improvement.

3.5 Discussion

3.6 Conclusion

Chapter 4

Detecting EWFO on MAV

4.0.1 Reducing 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 [48]. 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.

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] Pedro Felzenszwalb, David Mcallester, and Deva Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. URL http://people.cs.uchicago.edu/{~}pff/papers/latent.pdf.

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

[41] 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.

[42] 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/.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

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

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

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

[51] 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.

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

[53] 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.

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

[55] 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.