# Semantic Segmentation using Deep Neural Networks for MAVs

## Gate Detection for MAVs in Autonomous Drone Racing

T.V. Tran

Faculty of Aerospace Engineering

TU Delft

# Semantic Segmentation using Deep Neural Networks for MAVs

## Gate Detection for MAVs in Autonomous Drone Racing

by


## T.V. Tran


to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on January 19, 2022 at 09:30 AM.

| | | |
|---|---|---|
| Student number: | 4449142 | |
| Project duration: | February, 2021 – January, 2022 | |
| Thesis committee: | Prof. dr. G.C.H.E. de Croon | TU Delft, Chair |
| | Dr. J.C. van Gemert | TU Delft, External Member |
| | Ir. C. de Wagter | TU Delft, Independent Member |
| | Y. Xu | TU Delft, Additional Member |


An electronic version of this thesis is available at `http://repository.tudelft.nl/`.


TU Delft

MAVLab

# Acknowledgments

This marks the end of a challenging but enjoyable academic journey at the faculty of Aerospace Engineering. Throughout my time in Delft, I have gained invaluable educational experience, personal growth that I could not have imagined when starting this journey, and most importantly met incredible people that are highly passionate and motivated that always kept me going even in extremely challenging times.

First of all, I would like to thank both my supervisors Prof. dr. Guido de Croon and Yingfu Xu for their invaluable guidance and positivity throughout this research. Guido, thank you for always being positive-minded, providing valuable feedback, inspiration, and motivation even in times when I was completely lost in the research topic. Yingfu, thank you for always being kind, taking the time to answer the endless questions that I had, and helping me solve the problems that I encountered.

Secondly, special thanks to Nick for being a great friend, the infinite amount of time spent in the fellowship, the endless talks and discussions about the challenges that we faced during the thesis, and last but not least, starting this long journey together in Delft.

Finally, I would like to thank my friends and family for being supportive during this period and taking the time to listen to my unsolicited conversations about me trying to explain my research, knowing in the back of my mind that most are not able to follow me but still be very patient. A great appreciation to all that have been with me during this period.

*T.V. Tran*
*Delft, January 2022*

# Abstract

Semantic segmentation methods have been developed and applied to single images for object segmentation. However, for robotic applications such as high-speed agile Micro Air Vehicles (MAVs) in Autonomous Drone Racing (ADR), it is more interesting to consider temporal information as video sequences are correlated over time. In this work, we evaluate the performance of state-of-the-art methods such as Recurrent Neural Networks (RNNs), 3D Convolutional Neural Networks (CNNs), and optical flow for video semantic segmentation in terms of accuracy and inference speed on three datasets with different camera motion configurations. The results show that using an RNN with convolutional operators outperforms all methods and achieves a performance boost of 10.8% on the KITTI (MOTS) dataset with 3 degrees of freedom (DoF) motion and a small 0.6% improvement on the CyberZoo dataset with 6 DoF motion over the single-frame-based semantic segmentation method. The inference speed was measured on the CyberZoo dataset, achieving 321 fps on an NVIDIA GeForce RTX 2060 GPU and 30 fps on an NVIDIA Jetson TX2 mobile computer.

# Contents

# List of Figures

# List of Tables

# Nomenclature

**List of Abbreviations**

| | |
|---|---|
| ADR | Autonomous Drone Racing |
| AHRS | Attitude Heading Reference System |
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| AP | Average Precision |
| BCE | Binary Cross Entropy |
| BPTT | Back-Propagation Through Time |
| CE | Cross Entropy |
| CNN | Convolutional Neural Network |
| DFF | Deep Feature Flow |
| DNN | Deep Neural Network |
| DoF | Degrees of Freedom |
| EKF | Extended Kalman Filter |
| EPE | End-Point Error |
| FCN | Fully Convolutional Network |
| FGFA | Flow-Guided Feature Aggregation |
| FN | False Negative |
| FoE | Focus of Expansion |
| FP | False Positive |
| FPS | Frames per Second |
| GPU | Graphics Processing Unit |
| GRU | Gated Recurrent Unit |
| HOG | Histogram of Oriented Gradients |
| HSV | Hue Saturation Value |
| ILSVRC | ImageNet Large Scale Visual Recognition Challenge |
| IoU | Intersection over Union |
| LSTM | Long Short-Term Memory |
| mAP | Mean Average Precision |
| MAV | Micro Air Vehicle |
| MAVLab | Micro Air Vehicle Laboratory |
| ML | Machine Learning |
| MLP | Multi-Layer Perceptron |
| MSE | Mean Squared Error |
| NMS | Non Maximum Suppression |
| PAF | Part Affinity Fields |
| PCA | Principal Component Analysis |

| | |
|---|---|
| R-CNN | Region-based Convolutional Neural Network |
| RCNN | Recurrent Convolutional Neural Network |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |
| RoI | Region of Interest |
| RPN | Region Proposal Network |
| SGD | Stochastic Gradient Descent |
| SIFT | Scale Invariant Feature Transform |
| SLAM | Simultaneous Localization and Mapping |
| SURF | Speeded Up Robust Features |
| TP | True Positive |
| TTC | Time-to-Contact |
| VIO | Visual Inertial Odometry |
| VMT | Vehicle Motion Tensor |

# 1 Introduction

The first Autonomous Drone Racing (ADR) competition was held at the IROS 2016 robotics conference in Daejeon, South Korea [1]. This ADR competition requires the drones to navigate through a predefined sequence of gates using available onboard resources. ADR competitions have pushed the boundaries of the technology advancement in autonomous navigation of Micro Air Vehicles (MAVs) and have challenged to close the gap between Artificial Intelligence (AI) and humans. In 2019, team MAVLab from the Netherlands participated in the AlphaPilot AI Drone Innovation Challenge hosted by Lockheed Martin and was able to secure first place in the competition. As the winning team, they also participated in a race against one of the best drone pilots in the world on a race track. While the drone pilot won the race, it was only five seconds faster than team MAVLab, showing the great potential of closing the gap in performance between AI and humans.

The ADR competitions have led to sophisticated methods to improve the autonomous navigation of MAVs, in particular in the domain of perception based on computer vision. In the work of Rojas-Perez et al. [2], a solution is proposed using monocular metric visual Simultaneous Localization and Mapping (SLAM). Li et al. [3] developed a computationally efficient vision-based navigation and control strategy using the snake gate algorithm based on color segmentation. Jung et al. [4] noticed that traditional color-based detection was sensitive to changes in illumination and introduced a deep learning detection method referred as ADR-Net based on single-image training input. Recently, researchers have noticed the potential of deep learning methods contributing towards the advancement of autonomous navigation of MAVs. Current deep learning methods applied in the perception of high-speed autonomous MAVs heavily rely on detection based on single frames obtained from onboard cameras. However, these methods only consider spatial dependencies and neglect the temporal coherence in perception. These deep learning methods are trained on still images, whereas the frames obtained from the onboard cameras contain salient temporal information that is not used to their fullest potential. Hence, the accuracy of the detection methods suffers from degenerated object appearance such as motion blur, occlusion, and pose variations during high-speed ADR [5].

In this research, different deep learning detection methods will be investigated using image sequences to incorporate temporal information. Training deep learning detection methods with image sequences provide both salient spatial and temporal information compared to a single image input. The goal of this research is to gain a better understanding of the effect of incorporating temporal information on the detection accuracy in deep-learning-based object detection and apply it to a better gate detection approach in ADR. The detection accuracy and performance of the deep learning algorithms will be evaluated.

## 1.1. Motivation and Research Objective

The Micro Air Vehicle Laboratory (MAVLab), at which the research will take place was the winning team in the AlphaPilot AI Drone Innovation Challenge hosted by Lockheed Martin in 2019. They participated in a race against one of the best drone pilots in the world in a short race track in which the drone pilot outperforms team MAVLab by only 5 seconds. The performance of team MAVLab was close to the human pilot, however, the current high-speed autonomous Micro Air Vehicle (MAV) system is still sub-optimal and has to be improved.

The high-speed autonomous MAV system can be divided into four main areas; perception, state estimation, planning, and control module. During Autonomous Drone Racing (ADR), the MAV has to perceive the environment and be able to successfully perform gate detection to fly through gates to finish the race track as fast as possible. Without a robust detection system, the MAV is incapable of navigating autonomously through a race track. Hence, the perception module is fundamental and acts as the basis for the state estimation, planning, and control module of the MAV.

In this research, the main goal is to gain a better understanding of the effect of incorporating temporal information on the detection accuracy in deep-learning-based object detection and apply it to a better gate detection approach in ADR. Different deep learning methods will be explored and investigated in the domain of perception based on computer vision. The most suitable deep learning methods will be selected, tested, and evaluated for real-time gate detection. The main objective of the thesis can be formulated as:

> *To provide a solution for high-speed autonomous MAV racing in gate detection by designing a deep learning algorithm using image sequences that can be deployed on a MAV capable of detection in real-time.*

## 1.2. Research Questions

The research objective is to provide a solution for high-speed autonomous MAV racing in gate detection by designing a deep learning algorithm using image sequences that can be deployed on a MAV capable of detection in real-time. The main research question is defined as:

> *How can a high-speed autonomous MAV perform gate detection with deep learning using image sequences in real-time?*

The research question can then be divided into sub-questions:

1. What are the fundamental differences between training a deep learning algorithm with a single input image compared to image sequences?
2. Which deep learning algorithms are best suited for detection using image sequences?
3. What deep learning algorithms perform the best in terms of detection accuracy and inference speed?
4. How does the algorithm perform in real-time when deployed on a high-speed autonomous MAV?

## 1.3. Thesis Structure

In Part I, the main contribution of this thesis will be presented in the form of a scientific article that can be read as a standalone document. The article consists of an introduction about the research, an overview of state-of-the-art methods in the research area, a detailed description of the methodology, the results obtained from the experiments, and concluding remarks on the results obtained from this research. The content of the remainder of this thesis acts as supporting material in addition to the article. Therefore, it is advised that readers that are not familiar with the scientific concepts of this research, should first read the supporting documents.

Part II provides a detailed overview of the concepts and state-the-art methods applied in the field of object detection. First, in chapter 2, the theory behind the concept of deep learning applied for computer vision tasks is discussed. Then, in chapter 3, an overview of object detection methods that have been applied in the past and current state-of-the-art methods is presented. In chapter 4, existing methods that have been applied for gate detection in ADR are reviewed and in chapter 5 the most important findings from the literature are summarized. Finally, in chapter 6, the final remarks of the content discussed in Part II are presented.

Finally, in Part III, the main conclusions of the research are presented. In chapter 7, the research questions that were initially proposed in chapter 1 are answered in detail. An overview of the project planning timeline for this research can be found in Appendix A.

# I

# Scientific Article

# Semantic Segmentation using Deep Neural Networks for MAVs

Tommy Tran Yingfu Xu Guido C.H.E. de Croon

Micro Air Vehicle Laboratory, Delft University of Technology, The Netherlands

***Abstract*— Semantic segmentation methods have been developed and applied to single images for object segmentation. However, for robotic applications such as high-speed agile Micro Air Vehicles (MAVs) in Autonomous Drone Racing (ADR), it is more interesting to consider temporal information as video sequences are correlated over time. In this paper, we evaluate the performance of state-of-the-art methods such as Recurrent Neural Networks (RNNs), 3D Convolutional Neural Networks (CNNs), and optical flow for video semantic segmentation in terms of accuracy and inference speed on three datasets with different camera motion configurations. The results show that using an RNN with convolutional operators outperforms all methods and achieves a performance boost of 10.8% on the KITTI (MOTS) dataset with 3 degrees of freedom (DoF) motion and a small 0.6% improvement on the CyberZoo dataset with 6 DoF motion over the single-frame-based semantic segmentation method. The inference speed was measured on the CyberZoo dataset, achieving 321 fps on an NVIDIA GeForce RTX 2060 GPU and 30 fps on an NVIDIA Jetson TX2 mobile computer.**

## I. INTRODUCTION

Semantic segmentation is a fundamental task in computer vision, with the goal to separate objects pixel-wise from the background that belong to a certain class or category. In the past, semantic segmentation has been applied to single images using Convolutional Neural Networks (CNNs) to capture spatial features and achieve decent performance [1–4].

In semantic segmentation, CNNs are state-of-the-art and achieve promising results. However, CNNs process images independently, this results in one main limitation, which is the inability to capture temporal information. This aspect is important because computer vision tasks in real-world flying robotic applications contain a stream of continuous images, which are correlated over time. Hence, conventional semantic segmentation methods in robotic applications may suffer from performance degradation due to motion blur in images as a result of fast agile maneuvers [5], incomplete segmentation masks for distant objects, and objects that are occluded.

Early methods adopted in gate detection for Micro Air Vehicles (MAVs) rely on characteristic features such as the structure, shape, or color of a certain object [7, 8]. However, these methods lack the ability to adapt to different illumination and environmental conditions. A more robust approach is to apply Deep Learning (DL) for gate detection, which has shown satisfactory performance and proven its capability to run in real-time onboard of MAVs [5, 9, 10]. However, none of these methods leverage information over time for gate detection, which remains an unexplored area of research.

The authors are with the Micro Air Vehicle Laboratory, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands. (emails: tommyvtran97@gmail.com; Y.Xu-6@tudelft.nl; G.C.H.E.deCroon@tudelft.nl).

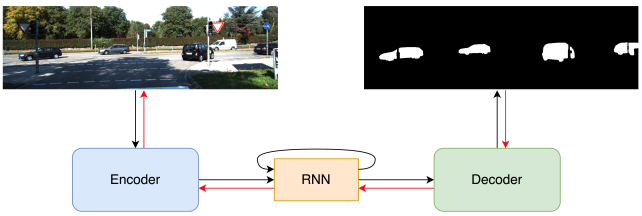


Fig. 1. An overview of the framework that demonstrates a good trade-off between performance and inference speed for flying robotic applications. The model is trained using an encoder-decoder architecture as proposed in U-Net [1]. Moreover, a recurrent neural network between the encoder and decoder is added to propagate information over time. The red arrows indicate back propagation through time [6].

Recent works, outside the domain of flying robotic applications, have explored possibilities to incorporate information over time to improve the performance in video semantic segmentation. Recurrent Neural Networks (RNNs) capture temporal information as past information is used as input to the network, allowing the network to retain relevant information over time. Methods relying on RNNs in video semantic segmentation [11–13] extend an encoder-decoder network architecture using Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) with convolutional operators to preserve spatial information and capture temporal dependencies. Alternatively, 3D CNNs allow capturing temporal information using convolution in both the spatial and temporal direction in contrast to conventional CNNs [14, 15]. Both the RNN and 3D CNN methods are learning motion cues over time without motion information provided. Therefore, explicitly providing the network with motion information such as optical flow maps from existing state-of-the-art optical flow estimation algorithms [16–19] is another approach to incorporate information over time. There are several ways to provide the network with motion information from the optical flow estimations. For example, the optical flow estimations can be used to warp the features [20, 21] from the previous to the current frame. Furthermore, an appearance stream and optical flow stream [22–24] can be used to process both modalities separately, which then can be fused to enhance the segmentation predictions. Alternatively,

solely optical flow can also be leveraged without the use of an appearance stream for semantic segmentation [25].

These methods achieve state-of-the-art performance, however, the question remains whether they can be deployed in real-time on MAVs with limited computational resources while still achieving good performance in terms of accuracy. In this paper, we study several methods that utilize temporal information to improve video semantic segmentation for robotic applications such as MAVs. Our main contributions in this paper are as follows:

1) To provide a quantitative comparison between the single-frame-based semantic segmentation baseline against methods that utilize temporal information in terms of accuracy and inference speed, evaluated on three datasets with different camera motion configurations.
2) To show that a simple RNN with convolutional operators outperforms all methods and achieves a performance increase of 10.8% and 0.6% relative to the single-frame-based semantic segmentation baseline on the KITTI (MOTS) dataset with 3 degrees of freedom (DoF) and CyberZoo dataset with 6 DoF.
3) To show that a network using a simple RNN can be deployed on an MAV for gate detection in real-time equipped with an NVIDIA Jetson TX2 running at 30 fps.

## II. Related Work

### A. Image Semantic Segmentation

Early methods have shown the great success of applying deep neural networks [26–29] for image classification on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [30]. Due to this success, image classification tasks have been extended towards semantic segmentation tasks in which the objects are segmented from the background pixel-wise. Shelhamer et al. [3] introduced a fully convolutional neural network (FCN) that can learn to predict dense pixel-wise predictions for semantic segmentation tasks. Ronneberger et al. [1] build upon this concept and extended the architecture [3] to allow faster training time with more accurate segmentation predictions called U-Net, consisting of a contracting and expanding path with skip-connections. The main limitation with these methods is that temporal dependencies are not considered, which is an important factor considering the sequential data processing in real-world flying robotic applications as with MAVs. Therefore, when applied to video sequences their performance may drop due to motion blur from fast agile maneuvers, incomplete segmentation masks for distant objects, and objects that are occluded.

### B. Video Semantic Segmentation

*1) Recurrent Neural Network:* Recent works in video semantic segmentation have used RNNs due to their ability to process images sequentially and retain relevant information over time that is paramount in making segmentation predictions. Recurrent networks that are able to process long sequences and partly solve the vanishing and exploding gradient problem are LSTMs and GRUs. Since video semantic segmentation deals with a stream of consecutive images, LSTMs and GRUs have been extended towards their convolutional counterparts by replacing the operators with convolutional operators, convLSTM [31] and convGRU. In the work of Valipour et al. [12], the FCN is extended with a recurrent module using a convLSTM and convGRU that preserve and capture spatial and temporal information. The results show that by extending the network with a recurrent module, the performance of semantic segmentation is improved. Yurdakul et al. [13] investigated the contribution of depth information for semantic segmentation and showed that depth information contributes towards a higher quality of segmentation. Pfeuffer et al. [11] studied different locations of placing the convLSTM layer in the network and showed that a convLSTM layer between the encoder and decoder does not yield the most optimal results. Furthermore, they showed that the convLSTM based approach is able to achieve real-time capabilities running less than 100 ms.

*2) 3D Convolutional Neural Network:* 3D CNN is an extension of the conventional CNN which allows the network to extract both spatial and temporal features. In the work of Dong et al. [14], a combination of both 2D and 3D convolution is proposed. First, the spatial features are extracted with the 2D convolutional pipeline. Then, the spatial features are combined across time such that the 3D convolutional pipeline is able to extract temporal features. Akilan et al. [15] proposed a network that combines both 3D CNN and LSTM. The 3D CNN can directly capture short temporal motion, whereas the LSTM is responsible for capturing long-term dependencies.

*3) Optical Flow:* Another promising way to incorporate information over time is to provide the network explicitly with motion information using existing state-of-the-art optical flow estimation algorithms [16–19]. Zhu et al. [20] and Gadde et al. [21] both leveraged the optical flow estimations to warp features from the previous to the current frame to improve the performance of semantic segmentation. Siam et al. [23] proposed a two-stream network with an appearance and optical flow stream, where the appearance stream processes the RGB image and the optical flow stream the optical flow map. In the work of Rashed et al. [24], the contribution of different modalities is investigated such as RGB, depth, and optical flow. The results from this study showed that fusing depth and optical flow separately achieved the highest performance. The decent performance of providing the network with depth and motion information, however, requires a high computational overhead which poses a challenge to run in real-time for flying robotic applications with hardware constraints.

### C. Gate Detection in Autonomous Drone Racing

Early methods adopted in gate detection for Autonomous Drone Racing (ADR) heavily rely on feature extraction based on the structure, shape, and color of the object. In the work

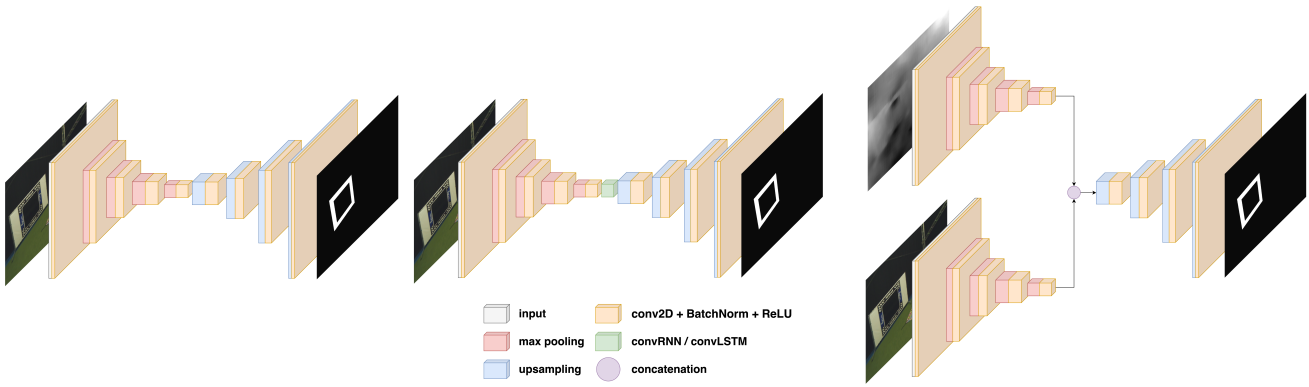


Fig. 2. The different network architectures that are considered in this paper from left to right, U-Net, U-Net with a recurrent module between the encoder and decoder, and U-Net with an appearance and optical flow network stream. Note that in all network architectures element-wise summation skip connections are used, but not shown in the figure. Furthermore, the 3D CNN architecture is not shown as it is similar to U-Net with 2D convolutional operators replaced by 3D convolutional operators and with multiple images as input to the network.

of Jung et al. [7] and Li et al. [8], a color detection algorithm is used to successfully perform gate detection in ADR. The main limitation with this method is that it is not robust because it does not account for different illumination and environmental conditions. More recent methods have applied DL to perform gate detection. In the work of Jung et al. [9], a visual detection method on a high-speed MAV using CNNs is presented which showed decent gate detection performance while running in real-time on an MAV equipped with an NVIDIA Jetson TX2. However, in their approach, they used bounding boxes to detect the racing gates which is less accurate compared to a semantic segmentation approach. In the work of Foehn et al. [10], a different method is used to perform gate detection by using a deep neural network to detect the inner corners of the racing gates. Alternatively, the winning entry of the AlphaPilot AIRR Competition 2019 [5], used U-Net [1] to generate segmentation masks for gate detection and has shown that their method is both accurate and efficient running at 75 Hz on a Jetson Xavier.

## III. Methodology

### A. Baseline Network

We start with the U-Net [1] architecture as the baseline for all our experiments as our main goal is to study the impact of different methods that utilize temporal information in terms of accuracy and inference speed. The accuracy is measured using the evaluation metric Intersection over Union (IoU) as defined by Eq. 1, which is a measure for overlap similarity with respect to the ground truth.

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{1}$$

Where $TP$ is the number of pixels labeled as the object corresponding to the object, $FP$ the pixels labeled as the object not corresponding to the object and $FN$ the pixels labeled as background not corresponding to the background.

The baseline architecture consists of a 4-layer network with [32, 64, 128, 256] convolutional filters, kernel size of 3×3, and element-wise summation skip-connections. All layers consist of batch normalization [32] followed by a ReLU activation function except for the output layer which consists of a sigmoid activation function in the range [0, 1]. The input to the network consists of a normalized RGB image in the range [0, 1] and outputs a segmentation mask of the current frame at time step $t$.

### B. Recurrent Neural Network

The baseline architecture is extended with a recurrent module between the encoder and decoder shown in Fig. 2. We evaluate two different recurrent modules, the convRNN and convLSTM [31], and replace the operators with convolutional operators that are more appropriate for processing images. The relationship representing the convRNN and convLSTM cell are given by Eq. 2 and Eq. 3, respectively.

$$\begin{aligned} \boldsymbol{I_t} &= (\boldsymbol{X_t} \circledast \boldsymbol{W_{xi}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hi}} + \boldsymbol{b_i}) \\ \boldsymbol{H_t} &= \tanh(\boldsymbol{I_t}) \\ \boldsymbol{O_t} &= (\boldsymbol{H_t} \circledast \boldsymbol{W_{ho}} + \boldsymbol{b_o}) \end{aligned} \tag{2}$$

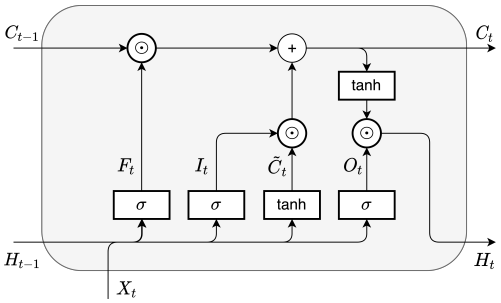


Fig. 3. A representation of the internal operators within a Long Short-Term Memory cell with $\boldsymbol{F_t}$ the forget gate, $\boldsymbol{I_t}$ the input gate, $\boldsymbol{\tilde{C}_t}$ the candidate memory cell, and $\boldsymbol{O_t}$ the output gate.

$$
\begin{aligned}
\boldsymbol{F_t} &= \sigma(\boldsymbol{X_t} \circledast \boldsymbol{W_{xf}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hf}} + \boldsymbol{b_f}) \\
\boldsymbol{I_t} &= \sigma(\boldsymbol{X_t} \circledast \boldsymbol{W_{xi}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hi}} + \boldsymbol{b_i}) \\
\tilde{\boldsymbol{C}}_{\boldsymbol{t}} &= \tanh(\boldsymbol{X_t} \circledast \boldsymbol{W_{xc}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hc}} + \boldsymbol{b_c}) \\
\boldsymbol{O_t} &= \sigma(\boldsymbol{X_t} \circledast \boldsymbol{W_{xo}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{ho}} + \boldsymbol{b_o}) \\
\boldsymbol{C_t} &= \boldsymbol{F_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{I_t} \odot \tilde{\boldsymbol{C}}_{\boldsymbol{t}} \\
\boldsymbol{H_t} &= \boldsymbol{O_t} \odot \tanh(\boldsymbol{C_t})
\end{aligned}
\tag{3}
$$

Where $\boldsymbol{F_t}$ is the forget gate, $\boldsymbol{I_t}$ the input gate, $\tilde{\boldsymbol{C}}_{\boldsymbol{t}}$ the candidate memory cell, $\boldsymbol{O_t}$ the output gate, $\boldsymbol{H_t}$ the hidden state and $\boldsymbol{C_t}$ the memory at the current time step, $\boldsymbol{H_{t-1}}$ the hidden state and $\boldsymbol{C_{t-1}}$ the memory at the previous time step, $\sigma$ the sigmoid activation function, tanh the hyperbolic tangent activation function, ($\circledast$) the convolutional operator, and ($\odot$) the element-wise multiplication operator. The recurrent module consists of 1-layer with 256 convolutional filters with a fixed kernel size of 3×3.

### C. 3D Convolutional Neural Network

The network architecture is identical to the baseline architecture in Fig. 2 except that the 2D convolutional operators are replaced by 3D convolutional operators. Since 3D convolutional operators are used, the input to the network is a set of consecutive images. In our experiments we use frame $\boldsymbol{F_{t-1}}$ and $\boldsymbol{F_t}$ with overlapping images to predict the current segmentation mask for frame $\boldsymbol{F_t}$.

### D. Appearance and Optical Flow Concatenation

We modify the input to the baseline by concatenating the RGB images with different representations of the optical flow map. The optical flow maps are generated by LiteFlowNet [18], where we investigate different optical flow map representations using the magnitude (M), horizontal (U), and vertical (V) components. The first configuration is the RGB image concatenated with the magnitude resulting in a 4-channel map. The second configuration consists of concatenation between the RGB image and the horizontal and vertical flow map resulting in a 5-channel map. The third configuration concatenates the RGB image with the horizontal, vertical, and magnitude flow map resulting in a 6-channel map. In the remainder of this paper, we refer to these types of networks as RGB-M, RGB-UV, and RGB-UVM, respectively.

### E. Appearance and Optical Flow Stream Network

We extend the baseline by creating two parallel network streams as shown in Fig. 2. The top network stream is used to process the optical flow maps generated by LiteFlowNet [18]. We consider three optical flow map configurations as input to the optical flow network stream. The first configuration is the magnitude, resulting in a 1-channel flow map (M). The second configuration concatenates the horizontal and vertical components, resulting in a 2-channel flow map (UV). The third configuration concatenates the horizontal, vertical, and magnitude, resulting in a 3-channel flow map (UVM). In the remainder of this paper, we refer to these types of networks as RGB+M, RGB+UV, and RGB+UVM, respectively. The bottom network stream is used to process the RGB images. Both networks do not share weights and will learn separately how to process the RGB images and optical flow maps, respectively.

### F. Dataset

*1) KITTI MOTS:* We evaluate our experiments on three datasets. The KITTI Multi-Object Tracking and Segmentation (MOTS) [33, 34] dataset consists of segmentation masks of cars and pedestrians, with 3 DoF motion. Since we are mainly interested in video semantic segmentation we discard the tracking part. Moreover, we limit the video semantic segmentation experiments to only segment the cars and remove the segmentation mask of pedestrians. The dataset consists of 21 sequences resulting in a total of 8008 images.

*2) KITTI MOTS Challenge:* The KITTI (MOTS Challenge) [33, 34] dataset consists of segmentation masks of pedestrians. The dataset consists of 4 sequences in which we only use sequences 0002 and 0009 resulting in a total of 1125 images since we want to evaluate the performance on a dataset with a stationary camera.

*3) CyberZoo:* Due to the absence of a publicly available dataset consisting of racing gates, we generate our own dataset in different illumination and environmental conditions. The CyberZoo dataset consists of segmentation masks of racing gates similar in appearance as in the AlphaPilot AIRR Competition with 6 DoF motion. The dataset consists of 26 sequences resulting in a total of 7214 images. To summarize we evaluate our experiments on the datasets with three different camera motion configurations:

- KITTI MOTS — 3 DoF Motion
- KITTI MOTS Challenge — Stationary
- CyberZoo — 6 DoF Motion

To evaluate, compare, and reduce the bias of the different networks, we perform our experiments using a $K$ = 5 cross-validation with 80% training and 20% validation samples on all the datasets.

### G. Training Implementation Details

Our framework is implemented in PyTorch[1], where we use the Adam Optimizer [35] for all of our networks with a learning rate of $10^{-4}$. We use a batch size of 4 and train the networks for 50 epochs. For the loss function, we both use the soft dice and focal loss [36] represented by Eq. 4 and Eq. 5.

$$
\mathcal{L}_{dice} = 1 - \frac{2\sum_{i=0}^{N} \boldsymbol{x_i}\boldsymbol{y_i} + \epsilon}{\sum_{i=0}^{N} \boldsymbol{x_i} + \sum_{i=0}^{N} \boldsymbol{y_i} + \epsilon} \tag{4}
$$

$$
\boldsymbol{p_i} = \begin{cases} \boldsymbol{p} & \text{if } \boldsymbol{y_i} = 1 \\ 1 - \boldsymbol{p} & \text{otherwise} \end{cases}
$$

[1]The source code can be found at https://github.com/tommyvtran97/MAV-Segmentation

TABLE I

COMPARISON BETWEEN THE DIFFERENT METHODS. WE REPORT THE MEAN IOU FOR COMPARISON (HIGHER IS BETTER, ↑). NUMBERS HIGHLIGHTED IN BOLD AND UNDERLINED ARE THE BEST AND SECOND BEST-PERFORMING NETWORK PER DATASET.

| Dataset | U-Net | RNN | LSTM | 3D | RGB-M | RGB-UV | RGB-UVM | RGB+M | RGB+UV | RGB+UVM |
|---|---|---|---|---|---|---|---|---|---|---|
| KITTI (MOTS) | 71.65 ± 4.85 | **79.40** ± 3.73 | <u>79.25</u> ± 4.45 | 70.43 ± 12.6 | 72.54 | 72.01 | 69.82 | 73.90 | 74.29 | 73.00 |
| KITTI (MOTS Challenge) | 79.73 ± 4.59 | 79.37 ± 4.15 | 78.77 ± 3.26 | **80.55** ± 3.78 | 77.14 | 76.86 | 76.66 | <u>80.16</u> | 79.36 | 80.13 |
| CyberZoo | 92.63 ± 2.54 | <u>93.16</u> ± 1.69 | **93.21** ± 1.46 | 92.93 ± 2.03 | 92.93 | 92.79 | 92.66 | 92.99 | 92.96 | 92.86 |
| Inference Speed (ms) | 2.87 | 3.12 | 3.56 | 14.71 | 20.41 | 20.41 | 22.22 | 25.00 | 25.00 | 25.00 |

$$\mathcal{L}_{focal} = -(1 - \boldsymbol{p_i})^{\gamma} log(\boldsymbol{p_i}) \tag{5}$$

Where $\boldsymbol{x_i}$ is the prediction of the network, $\boldsymbol{y_i}$ the ground truth, $\epsilon$ the smoothness factor, $\boldsymbol{p}$ the prediction probability, and $\gamma$ the relaxation parameter. Moreover, for both loss functions we use $\epsilon$ = 1 and $\gamma$ = 2. The final loss function that we use for training is a weighted combination of the soft dice and focal loss as described by Eq. 6, with $\lambda_1$ = 2.

$$\mathcal{L}_{loss} = \lambda_1 \mathcal{L}_{dice} + \mathcal{L}_{focal} \tag{6}$$

For the recurrent modules, the implementation details are slightly different. We split the dataset into non-overlapping sub-sequences with a length of $S$ = 20 because frames that are temporally closer are more relevant in the case of fast camera motion. During training, we use the same optimizer, learning rate, and batch size as the other networks, but train for 200 epochs as the recurrent modules require more time for convergence. Moreover, during training, we unroll the network for $T$ = 5 time steps and perform back propagation through time (BPTT) [6] such that the weights are updated according to the gradients accumulated over the last 5 time steps. For all the datasets we use the same training details and shuffle the datasets as it increases the performance by a substantial margin.

## IV. Results

### A. Experimental Results

We study several methods that utilize information over time and compare it to the baseline (U-Net). The experiments are performed using a $K$ = 5 cross-validation. We consider three types of methods, recurrent networks, 3D CNNs, and optical flow. Moreover, for the optical flow-based methods, we consider three different optical flow map configurations as input to the network. A quantitative comparison of the different methods on three datasets with different camera motion configurations is presented in Table I.

*1) KITTI MOTS Dataset:* The best-performing network of the two-stream method RGB+UV shows an increase of 3.7% in mean IoU over the baseline since the optical flow maps contribute towards a more accurate segmentation mask. When considering the network where the optical flow map is concatenated with the RGB image, the best result is obtained by RGB-M showing an increase of 1.2%. Furthermore, the results of the 3D CNN method show that the performance is worse compared to the baseline. The best-performing networks are the RNN and LSTM with an increase of 10.8% and 10.6% relative to the baseline, respectively. We argue that the small performance difference between the RNN and LSTM can be explained because of the fast motion of the camera. The fast motion will significantly change the pixel location of the objects over time. For example, this means that the pixel location of the objects may be far away from the initial position due to fast motion over time. Frames that are temporally closer are more relevant and therefore the effect of the long-term memory of the LSTM has a relatively small effect on the performance compared to the RNN. The results show that the RNN and LSTM outperform the other methods by a significant margin, which shows the promising capability of these networks to retain relevant information over time to improve the performance of semantic segmentation in videos compared to 3D CNN and optical flow-based methods. In addition, the small performance improvement for optical flow-based methods indicates that it might not be the optimal way to leverage optical flow.

*2) KITTI MOTS Challenge Dataset:* The 3D CNN method performs the best on this dataset showing an improvement of 1% over the baseline. Moreover, we observe that RGB+M also contributes to a modest improvement of 0.5% relative to the baseline. Different from the KITTI (MOTS) dataset, the camera is stationary. Furthermore, we do not see any improvements using the RNN and LSTM over the baseline, indicating that retaining past information on this dataset might not contribute to higher performance.

*3) CyberZoo Dataset:* The CyberZoo dataset contains the most challenging camera motion configuration in comparison to the other datasets including 6 DoF motion. From Table I, the best-performing networks are the RNN and LSTM with a small improvement of 0.6% relative to the baseline. Furthermore, we observe that the 3D CNN and optical flow-based methods are also showing a slight improvement compared to the baseline.

### B. Recurrent Neural Network Memory Analysis

The contribution of using memory for the RNN and LSTM is investigated by evaluating the performance of the network trained with memory by resetting the memory to zero or the average value of the memory during evaluation. We perform our experiments on the KITTI (MOTS) dataset and report the performance using a $K$ = 5 cross-validation and compare the results to the baseline. The results presented in Table III show a significant decrease in performance when the memory is reset to zero. We observe a similar behavior

TABLE II

COMPARISON OF THE TWO-STREAM NETWORK. WE REPORT THE MEAN IOU (HIGHER IS BETTER, ↑). NUMBERS HIGHLIGHTED IN BOLD ARE THE BEST-PERFORMING NETWORK PER DATASET.

| Dataset | U-Net | RGB+RGB | RGB+M | RGB+UV | RGB+UVM |
|---|---|---|---|---|---|
| KITTI (MOTS) | 71.65 ± 4.85 | 72.67 ± 6.10 | 73.90 ± 7.63 | **74.29** ± 7.33 | 73.00 ± 7.39 |
| KITTI (MOTS Challenge) | 79.73 ± 4.59 | 79.84 ± 4.84 | **80.16** ± 3.38 | 79.36 ± 5.00 | 80.13 ± 4.19 |
| CyberZoo | 92.63 ± 2.54 | **93.16** ± 1.84 | 92.99 ± 2.02 | 92.96 ± 1.92 | 92.86 ± 2.00 |

TABLE III

ANALYSIS OF THE CONTRIBUTION OF MEMORY OVER TIME FOR THE RNN AND LSTM BY RESETTING THE MEMORY TO ZERO OR THE AVERAGE VALUE. WE REPORT THE IOU (HIGHER IS BETTER, ↑).

| Memory | U-Net | RNN | LSTM |
|---|---|---|---|
| No Reset | 72.22 ± 5.32 | 79.74 ± 3.62 | 79.34 ± 4.69 |
| Reset to Zero | - | 71.45 ± 6.35 | 70.57 ± 5.89 |
| Reset to Average | - | 74.28 ± 5.27 | 75.15 ± 5.87 |

when the memory is reset to the average value, however, the decrease in performance is less significant. This confirms that the RNN and LSTM are using memory to improve the performance of the segmentation predictions over time.

### C. Optical Flow Analysis

The different optical flow configurations are investigated and their contribution towards the overall performance improvements are compared to the two-stream RGB+RGB network without optical flow. This network replaces the input optical flow map of the top stream as shown in Fig. 2 with an RGB image. Hence, the top and bottom stream will process frame $\boldsymbol{F_{t-1}}$ and $\boldsymbol{F_t}$, respectively. The results show that the optical flow configuration RGB+M or RGB+UV contributes towards a better performance on the KITTI (MOTS) and KITTI (MOTS Challenge) dataset relative to the RGB+RGB network. However, when considering the CyberZoo dataset, we observe that none of the optical flow configurations results in an improvement. This could be due to the camera motion configuration with 6 DoF, leading to inaccurate optical flow estimations as shown in Fig. 4.

Fig. 4. Example of an image pair from the CyberZoo dataset with 6 DoF camera motion and the corresponding optical flow magnitude map generated by LiteFlowNet [18] leading to inaccurate optical flow estimations.

### D. Network Inference Speed Evaluation

Apart from high accuracy, the demand for these computer vision tasks to run in real-time is also very important. For example, in robotic applications such as the gate detection task in ADR. In Table I, the computational cost for the different networks is evaluated on the CyberZoo dataset with a 288×224 resolution using an NVIDIA Geforce RTX 2060. We can observe from Table I that the networks using optical flow are significantly slower in terms of inference speed, which is primarily due to the computational overhead of the optical flow estimation algorithm [18]. Furthermore, we notice that both the RNN and LSTM show competitive inference speed in comparison to the baseline. Hence, when taking into account both the performance in terms of IoU and inference speed, the RNN and LSTM are the best options from all considered methods, demonstrating a good trade-off between performance and inference speed on the KITTI (MOTS) and CyberZoo dataset. In addition, we demonstrate the real-time capabilities by running the network on the NVIDIA Jetson TX2. The RNN network is able to run at 30 fps, showing its real-time capabilities for flying robotic applications.

We perform a sensitivity analysis for U-Net, RNN, and LSTM on the KITTI (MOTS) dataset by changing the size of the network and reporting the performance and inference speed in Fig. 7. We observe that both the RNN and LSTM show better performance even when the baseline is running at the same inference speed.

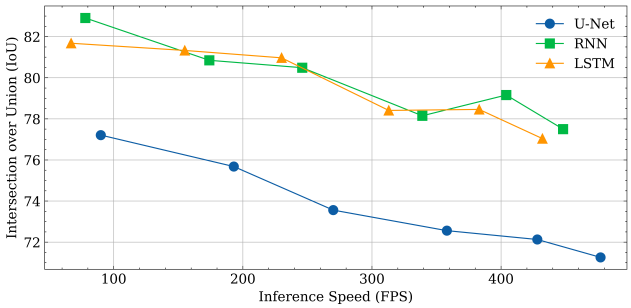


Fig. 7. A sensitivity analysis performed for U-Net, RNN, and LSTM on the KITTI (MOTS) dataset by changing the size of the network. We report the performance in terms of IoU and inference speed in FPS evaluated with a 620×186 resolution on an NVIDIA GeForce RTX 2060.

### E. Quantitative Comparison Network

In Fig. 5, the qualitative results are presented for the best-performing methods on the KITTI (MOTS) dataset compared

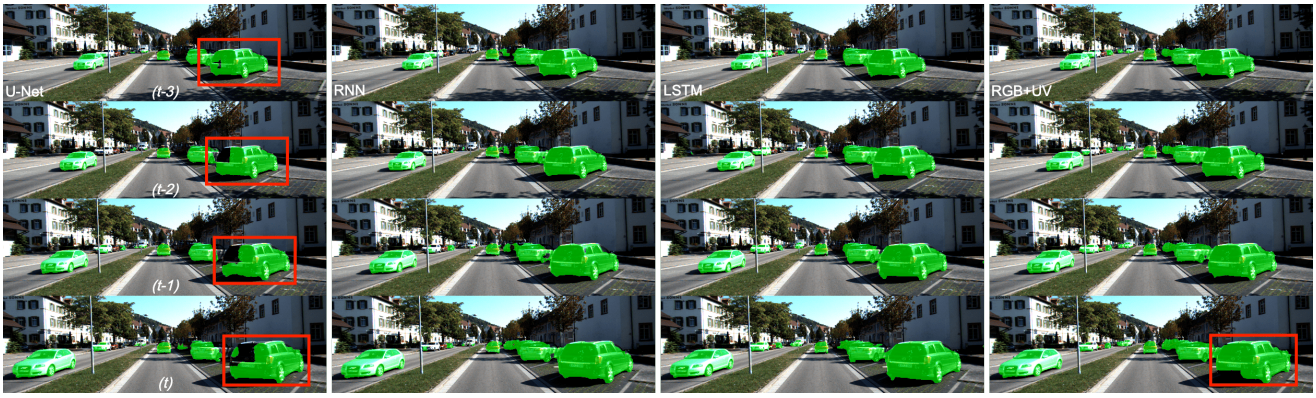


Fig. 5. Comparison on the KITTI (MOTS) dataset of the consistency of semantic segmentation predictions over time. The pixels highlighted in green correspond to the semantic segmentation predictions by the network. A comparison is shown between the baseline and best-performing networks on the KITTI (MOTS) dataset (RNN, LSTM, and RGB+UV), where the red boxes indicate the main differences with respect to the ground truth.

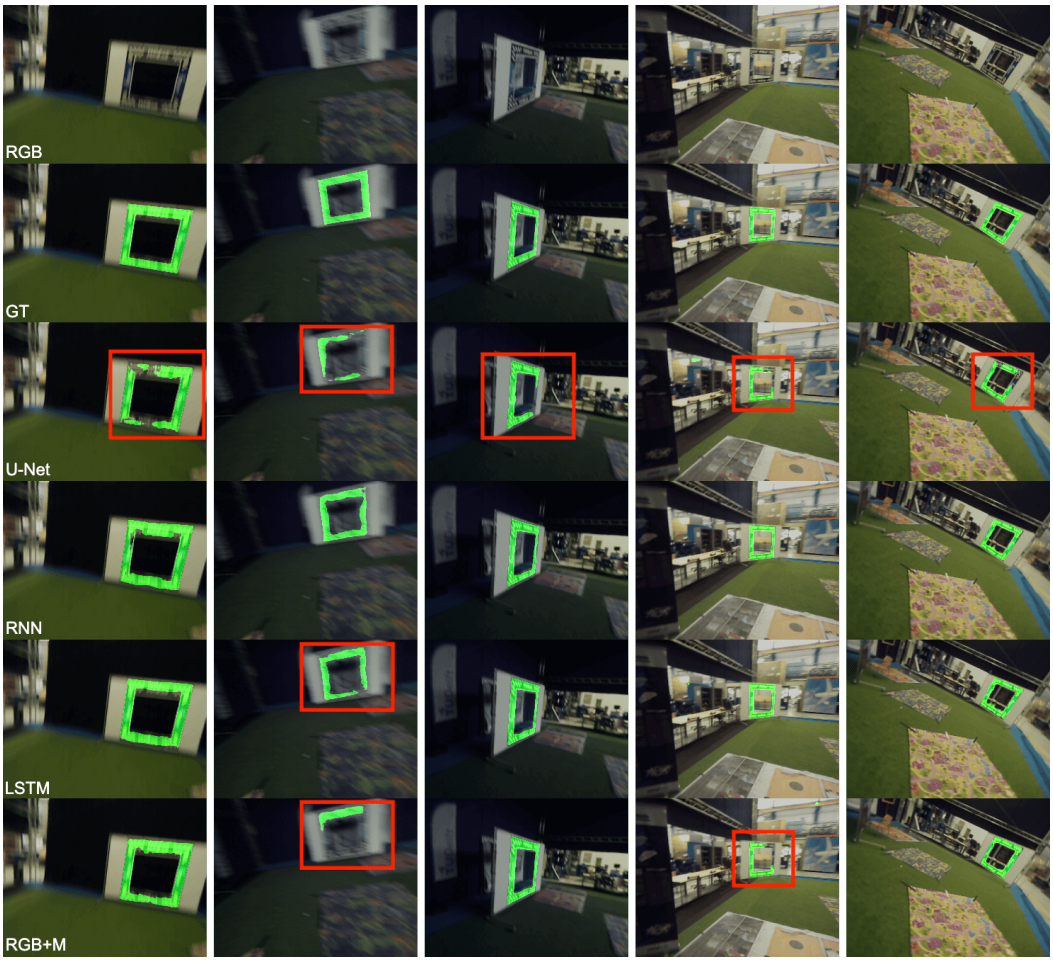


Fig. 6. Comparison on the CyberZoo dataset for the different networks. The pixels highlighted in green correspond to the semantic segmentation predictions by the network. A comparison is shown between the baseline and best-performing networks on the CyberZoo dataset (RNN, LSTM, and RGB+M), where the red boxes indicate the main differences with respect to the ground truth.

to the baseline. From Fig. 5, it can be noticed that for the sequence shown, the baseline shows artifacts when the object moves closer towards the camera compared to the RNN, LSTM, and RGB+UV. The results show that the baseline exhibits inconsistent prediction results over time. The RNN and LSTM show temporal consistency over time. This reduces the number of artifacts in segmentation predictions, which can be explained due to the ability of these networks to retain past relevant information over time. Moreover, we observe that RGB+UV also shows temporal consistency, however, some forms of artifacts are remaining. The results for the CyberZoo dataset are shown in Fig. 6, where we observe that the RNN and LSTM are more robust compared to the baseline. Furthermore, RGB+M also provides good performance, however, it is in some scenarios less robust compared to the RNN and LSTM. Additionally, we notice that the RNN and LSTM show decent performance even with the presence of severe motion blur due to camera motion.

## V. Conclusion

In this paper, we implement several methods utilizing information over time and provide a quantitative comparison against the baseline evaluated on three datasets with different camera motion configurations, the KITTI (MOTS), KITTI (MOTS Challenge), and CyberZoo dataset. We show by comparison, that the best performance relative to the baseline is achieved by leveraging the temporal capability of a simple RNN and LSTM. The results show that the RNN and LSTM are capable of providing temporal consistency over time which contributes towards a higher semantic segmentation quality on the KITTI (MOTS) with 3 DoF and the CyberZoo dataset with 6 DoF motion for flying robotic applications. The largest improvement is achieved on the KITTI (MOTS) dataset with an improvement of 10.8% and 10.6% relative to the baseline for the RNN and LSTM, respectively. The results on the CyberZoo dataset show a similar pattern as for the KITTI (MOTS) dataset resulting in an improvement of 0.6% for both the RNN and LSTM relative to the baseline. Moreover, we show that the RNN can be used for robotic applications such as MAVs in real-time as it achieves competitive performance and inference speed, running at 30 fps on an NVIDIA Jetson TX2.

### A. Additional Qualitative Results

Additional qualitative results are provided for the KITTI (MOTS) dataset in Fig. 8, where we show the best-performing networks, RNN, LSTM, and RGB+UV against the baseline for different sequences. Furthermore, in Fig. 9, we show the performance of the RNN, 3D, and RGB+M network for different sequences for comparison against the baseline on the KITTI (MOTS Challenge) dataset.

## References


[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer International Publishing, 2015, pp. 234–241.

[2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. "Region-Based Semantic Segmentation with End-to-End Training". In: *Computer Vision – ECCV*. Springer International Publishing, 2016, pp. 381–397.

[3] Evan Shelhamer, Jonathon Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. Vol. 39. 2016, pp. 1–1.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*. Vol. 39. 12. 2017, pp. 2481–2495.

[5] Christophe De Wagter, Federico Paredes-Vallés, Nilay Sheth, and Guido de Croon. *The Artificial Intelligence behind the winning entry to the 2019 AI Robotic Racing Competition*. 2021. arXiv: 2109.14985 [cs.RO].

[6] Paul Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE*. Vol. 78. 1990, pp. 1550–1560.

[7] 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". In: *Journal of Field Robotics (JFR)*. Vol. 35. 2018, pp. 146–166.

[8] Shuo Li, Michaël M.O.I. Ozo, Christophe De Wagter, and Guido C.H.E. de Croon. "Autonomous drone race: A computationally efficient vision-based navigation and control strategy". In: *Robotics and Autonomous Systems*. Vol. 133. 2020, p. 103621.

[9] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. "Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning". In: *IEEE Robotics and Automation Letters (RA-L)*. Vol. 3. 3. 2018, pp. 2539–2544.

[10] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. "AlphaPilot: Autonomous Drone Racing". In: *Robotics: Science and Systems XVI, Virtual Event / Corvalis, Oregon, USA, July 12-16, 2020*. 2020.

[11] Andreas Pfeuffer, Karina Schulz, and Klaus Dietmayer. "Semantic Segmentation of Video Sequences with Convolutional LSTMs". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1441–1447.

[12] Sepehr Valipour, Mennatullah Siam, Martin Jagersand, and Nilanjan Ray. "Recurrent Fully Convolutional Networks for Video Segmentation". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 29–36.

[13] Ekrem Emre Yurdakul and Yücel Yemez. "Semantic Segmentation of RGBD Videos with Recurrent Fully Convolutional Neural Networks". In: *IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 367–374.

[14] Shizhou Dong, Zhifan Gao, Sandeep Pirbhulal, Guibin Bian, Heye Zhang, Wanqing Wu, and Shuo Li. "IoT-based 3D convolution for video salient object detection". In: *Neural Computing and Applications*. Vol. 32. 2019, pp. 735–746.

[15] Thangarajah Akilan, Qingming Jonathan Wu, Amin Safaei, Jie Huo, and Yimin Yang. "A 3D CNN-LSTM-Based Image-to-Image Foreground Segmentation". In: *IEEE Transactions on Intelligent Transportation Systems (ITS)*. Vol. 21. 3. 2020, pp. 959–971.

[16] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766.

[17] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1647–1655.

[18] Hui Tak-Wai, Tang Xiaoou, and Loy Chen Change. "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8981–8989.

[19] Zachary Teed and Jia Deng. "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow (Extended Abstract)". In: *Proceedings of the*

*Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. 2021, pp. 4839–4843.

[20] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. "Flow-Guided Feature Aggregation for Video Object Detection". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 408–417.

[21] Raghudeep Gadde, Varun Jampani, and Peter V. Gehler. "Semantic Video CNNs Through Representation Warping". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4463–4472.

[22] Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. "FusionSeg: Learning to Combine Motion and Appearance for Fully Automatic Segmentation of Generic Objects in Videos". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2117–2126.

[23] Mennatullah Siam, Heba Mahgoub, Mohamed Zahran, Senthil Yogamani, Martin Jagersand, and Ahmad El-Sallab. "MODNet: Motion and Appearance based Moving Object Detection Network for Autonomous Driving". In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2859–2864.

[24] Hazem Rashed, Ahmad El Sallab, Senthil Yogamani, and Mohamed ElHelw. "Motion and Depth Augmented Semantic Segmentation for Autonomous Navigation". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 364–370.

[25] Gengshan Yang and Deva Ramanan. "Learning To Segment Rigid Motions From Two Frames". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 1266–1275.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems (ICONIP)*. Vol. 1. 2012, pp. 1097–1105.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9.

[28] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations (ICLR)*. 2015.

[29] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)*. Vol. 115. 3. 2015, pp. 211–252.

[31] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems (ICONIP)*. Vol. 1. MIT Press, 2015, pp. 802–810.

[32] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning (PMLR)*. Vol. 37. 2015, pp. 448–456.

[33] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[34] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. "MOTS: Multi-Object Tracking and Segmentation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[35] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations (ICLR)*. 2015.

[36] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal Loss for Dense Object Detection". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007.

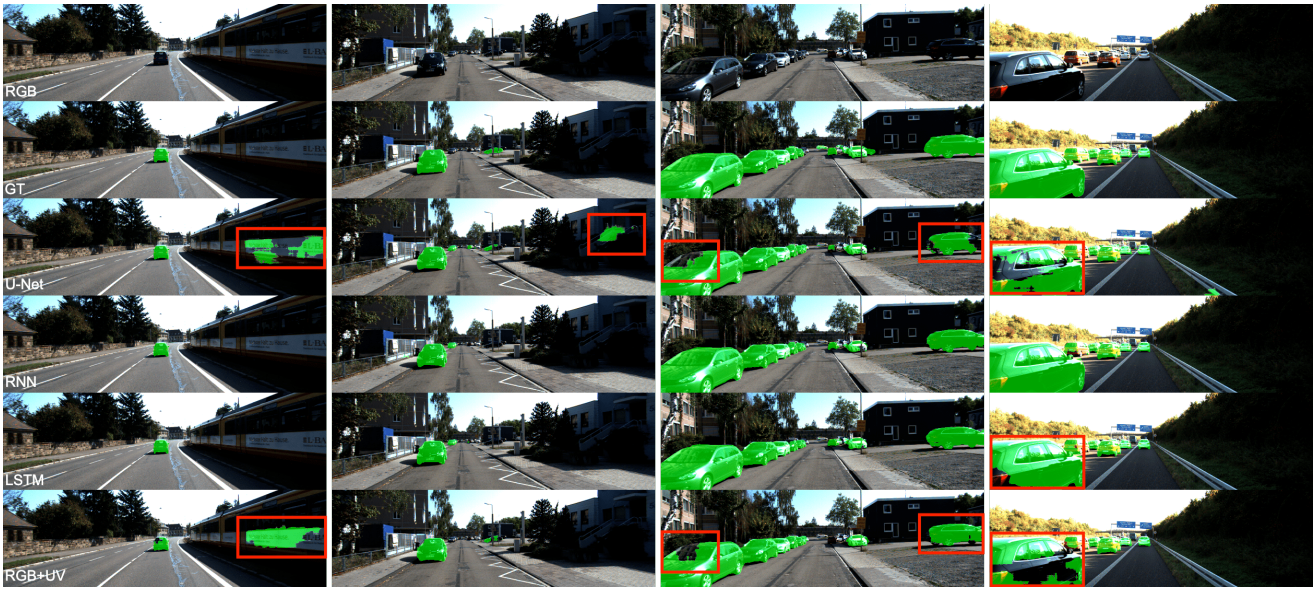


Fig. 8. Additional qualitative results on the KITTI (MOTS) dataset for the different networks. The pixels highlighted in green correspond to the semantic segmentation predictions by the network. A comparison is shown between the baseline and best-performing networks on the KITTI (MOTS) dataset (RNN, LSTM, and RGB+UV), where the red boxes indicate the main differences with respect to the ground truth.

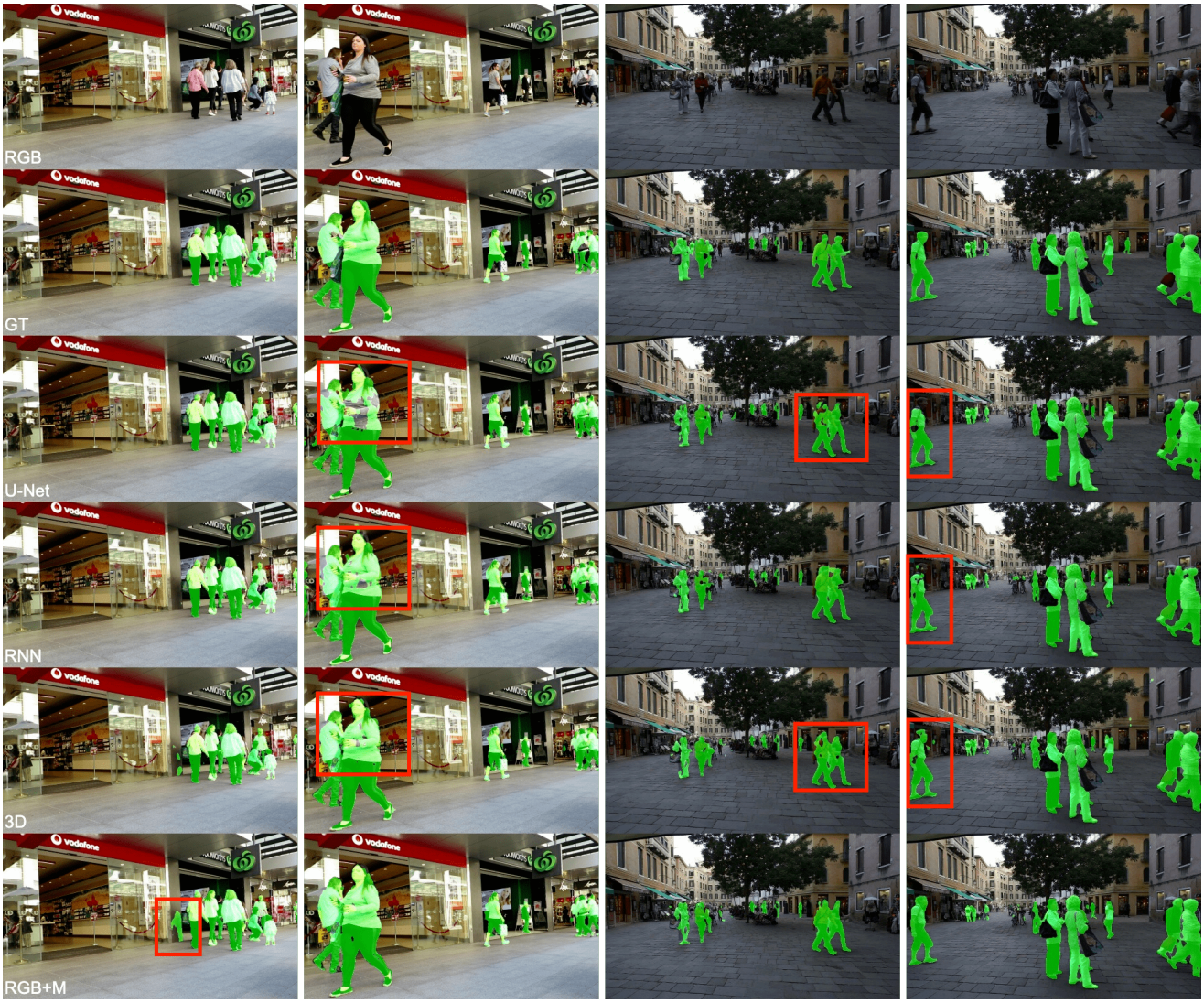


Fig. 9. Additional qualitative results on the KITTI (MOTS Challenge) dataset for the different networks. The pixels highlighted in green correspond to the semantic segmentation predictions by the network. A comparison is shown between the baseline and best-performing networks on the KITTI (MOTS Challenge) dataset (RNN, 3D, and RGB+M), where the red boxes indicate the main differences with respect to the ground truth.

# II

# Literature Study

# 2

# Deep Learning Theory

Deep learning is a part of Machine Learning (ML) in Artificial Intelligence (AI) that artificially mimics the functioning of the human brain by processing data through an Artificial Neural Network (ANN) to generate patterns to solve decision-related problems. The ANN is capable of learning in a supervised and unsupervised way by using labeled and unlabeled data to solve certain problems, which is also known as Deep Neural Learning or Deep Neural Network (DNN). The term DNN is used when a neural network architecture consists of many layers. In deep learning, the computer model performs a classification or regression-related task from images, text, or sound, achieving state-of-the-art performance, and on some occasions outperforming the human.

In this chapter the deep learning theory will be discussed. The most common neural networks for classification and regression-related tasks will be presented, starting with the feed-forward neural network in section 2.1. Then, in section 2.2, the Convolutional Neural Network (CNN) is discussed and finally in section 2.3 the Recurrent Neural Network (RNN) and its variants.

## 2.1. Feed Forward Neural Network

The feed-forward neural network also called Multi-Layer Perceptrons (MLP) is a fully connected network that is fundamental for deep learning models. The main objective of a feed-forward neural network is to approximate a certain mathematical function by minimizing the error [6]. The architecture of a feed-forward neural network is illustrated in Figure 2.1, where $x_i$ is the input, $\phi_j$ a non-linear activation function, and $y_k$ the output. The example illustrates a network with one hidden layer, however, this can be varied depending on the application and the complexity of the problem.

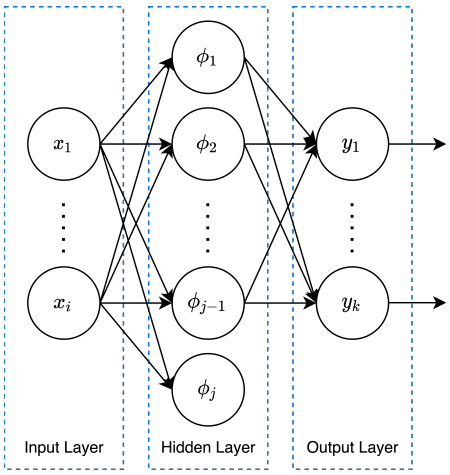


Figure 2.1: Example of a feed-forward neural network with 2 input neurons $x_i$, 4 hidden neurons with a non-linear activation function $\phi_j$ and 2 output neurons $y_k$. Note that the bias is omitted for the input and hidden layer but should be included when calculating the forward pass through the network.

For the example with a single hidden layer feed-forward neural network, a general mathematical expression can be derived, representing the network architecture given by Equation 2.1. Note that Equation 2.1 can be straightforward extended when multiple hidden layers are used in the feed-forward neural network.

$$y_k = \sum_{j}^{\infty} W_{jk}\phi_j\Big(\sum_{i}^{\infty} W_{ij}x_i + b_i\Big) + b_j \tag{2.1}$$

Where $W_{ij}$ represents the weights between the input and hidden layer, $W_{jk}$ the weights between the hidden and output layer, $\phi_j$ the non-linear activation function in the hidden layer, $b_i$ the bias in the input layer, $b_j$ the bias in the hidden layer, and $y_k$ the output in the output layer. Hence, a feed-forward neural network is simply a mapping of input $\boldsymbol{x}$ to output $\boldsymbol{y}$ given by Equation 2.2.

$$\boldsymbol{y} = f(\boldsymbol{x};\boldsymbol{\theta}) \tag{2.2}$$

Where $\boldsymbol{x}$ is the input vector, $\boldsymbol{y}$ the output vector, and $\boldsymbol{\theta}$ the learning parameters (weights and biases). The learning parameters will be updated according to the error generated by the network after the forward pass by using the back-propagation technique.

### 2.1.1. Non-Linear Activation Function

In the network, non-linear activation functions are frequently used to create complex mappings between the input and output of the network, which is fundamental for modeling complex data using images, text, or sound. The most common non-linear activation functions used in typical neural networks are the sigmoid, softmax, hyperbolic tangent, and rectified linear unit (ReLU) function. The sigmoid activation function maps an input to an output in the range [0, 1] and is defined by Equation 2.3.

$$\phi(x) = \frac{1}{1+e^{-x}} \tag{2.3}$$

The softmax activation function is similar to the sigmoid activation function. The input is mapped to an output in the range [0, 1]. However, the softmax function is generally used in applications regarding classification-related problems. In these types of problems, input is given and the network has to classify the output based on the input. In most cases, multiple classes are given in which only the class with the highest probability is selected as the prediction result. Hence, these types of classification problems require a softmax activation function at the end of the network to obtain the class with the highest probability. An example of an application in which a softmax activation function is used to obtain a probability is in the classification of animals in images. Each class is assigned a probability after the softmax function has been applied in which the class with the highest probability is selected as the prediction. The softmax activation function is given by Equation 2.4.

$$\phi_i(x) = \frac{e^{x_i}}{\sum_{j}^{n} e^{x_j}} \tag{2.4}$$

The hyperbolic tangent activation function is slightly different compared to the sigmoid and softmax activation function and maps an input to an output in the range [-1, 1]. This activation function has been used in Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks and is defined by Equation 2.5.

$$\phi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

The ReLU is a linear function for positive values and maps the output to zero for negative values. This means that all negative values that are passed through the ReLU function, which is defined by Equation 2.6, are directly mapped to zero.

$$\phi(x) = \max(0, x) \tag{2.6}$$

Another variation of the ReLU function is the Leaky ReLU function. The main difference is that the Leaky ReLU function defined by Equation 2.7 has a small leak for values < 0, hence negative values will be mapped to small negative values. An overview of the different activation functions is illustrated in Figure 2.2.

$$\phi(x) = \begin{cases} 0.01x & \text{for x} < 0 \\ x & \text{for x} > 0 \end{cases} \tag{2.7}$$

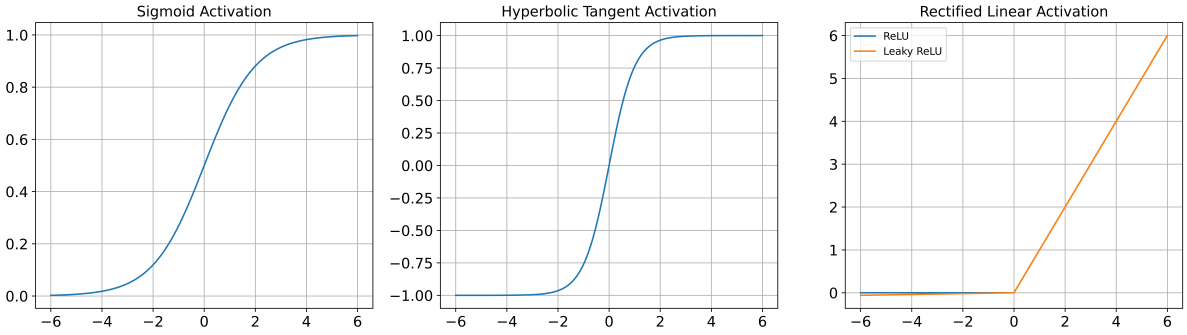


Figure 2.2: An overview of the different activation functions that are frequently used in deep learning applications. The figure on the left depicts the sigmoid activation, the figure in the middle the hyperbolic tangent activation, and the figure on right the (leaky) rectified linear activation.

### 2.1.2. Loss Function

To evaluate the performance of the network, the output of the network has to be compared to the ground truth of the original input. Based on the difference between the output of the network and the ground truth, the error can be defined. Generally, the performance of the network can be evaluated by the loss function. The Mean Squared Error (MSE) loss function is typically used for regression problems, which is the average of the squared difference between the ground truth $y_i$ and the prediction $\hat{y}_i$. The MSE loss function is given by Equation 2.8.

$$\text{MSE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n} \tag{2.8}$$

Another loss function that is frequently used in modern deep learning applications in combination with the softmax activation function in the output layer is the negative log-likelihood function, which is equivalent to the Cross-Entropy (CE) function given by Equation 2.9. The power of the CE function is that it penalizes predictions that are completely wrong resulting in a fast increase in the loss function. A variation of this loss function is the Binary Cross-Entropy (BCE) which is used for object segmentation and is given by Equation 2.10.

$$\text{CE} = -\frac{1}{m} \sum_{i=1}^{m} y_i log(\hat{y}_i) \tag{2.9}$$

$$\text{BCE} = -\frac{1}{m} \sum_{i=1}^{m} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_1) \tag{2.10}$$

### 2.1.3. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimization algorithm that is used to minimize the loss function in Deep Neural Networks (DNNs) by updating the learning parameters $\boldsymbol{\theta}$ in the network. The gradient computed in the SGD serves as an initial estimate for the actual gradient, which is obtained by sampling a mini-batch from the training dataset. The gradient is then used to update the learning parameters. The advantage of the SGD method is that the computational time is independent of the training dataset size as the mini-batch size is fixed [7]. The learning parameters are updated according to Equation 2.11.

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \epsilon \nabla_\theta L(\boldsymbol{x}; \boldsymbol{y}; \boldsymbol{\theta}) \tag{2.11}$$

Where $\boldsymbol{\theta}$ is a vector containing the learning parameters, $\epsilon$ the learning rate, $\nabla_\theta$ the gradient of $\boldsymbol{\theta}$, $L$ the loss function, $\boldsymbol{x}$ the input vector, and $\boldsymbol{y}$ the output vector.

The learning rate determines the rate of change of the network in response to the estimated error in which then $\boldsymbol{\theta}$ is updated. A high learning rate may impose violent oscillations in the cost function and therefore result in a higher training error between iterations. However, in the case of training stagnation, a high learning rate can contribute to escaping a local minimum and continuing to improve the network. For a small learning rate, the training process is rather slow and could result in stagnation during training at a high loss value.

Using a constant learning rate is not optimal, hence several improvements based on the SGD method exist. The method of momentum introduced by Polyak [8] is designed to accelerate the learning process by setting the velocity $\boldsymbol{v}$ to an exponentially decaying moving average of past gradients. The hyperparameter $\rho$ determines the contribution of this rate of decay. Then $\boldsymbol{v}$ is used to determine the updated learning parameters $\boldsymbol{\theta}$. An overview of the SGD-Momentum algorithm is shown in Equation 2.12.

$$\begin{aligned} \boldsymbol{v_i} &= \rho \boldsymbol{v_{i-1}} + (1-\rho)\nabla_\theta L(\boldsymbol{x}; \boldsymbol{y}; \boldsymbol{\theta}) \\ \boldsymbol{\theta}' &= \boldsymbol{\theta} - \epsilon \boldsymbol{v} \end{aligned} \tag{2.12}$$

The AdaGrad algorithm by Duchi et al. [9] adapts the learning rate by scaling the parameters inversely proportional to the square root of the sum of all past squared gradients. The RMSProp algorithm by Tieleman et al. [10] modifies the AdaGrad algorithm by changing the accumulation of past gradients into an exponentially moving weighted average. The Adam algorithm by Kingma et al. [11] is a combination of the RMSProp and Momentum, with the exception that a bias correction is introduced. Both the RMSProp and Adam algorithm are popular choices in deep learning applications.

## 2.2. Convolutional Neural Network

The Convolutional Neural Network (CNN) is a special type of neural network that can process data with a grid topology structure. CNNs are popular in deep learning applications that require image processing since images have a grid structure containing RGB pixels encoded with numerical values. One of the first CNN was introduced by LeCun et al. [12], by designing a CNN that can classify handwritten digits. The network architecture is shown in Figure 2.3 and consists of 3 convolutional, 2 max-pooling, and 2 fully connected layers.

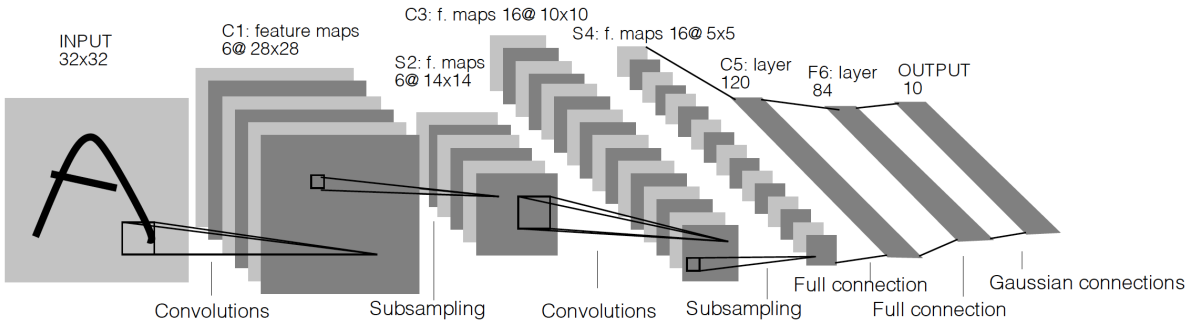


Figure 2.3: The Convolutional Neural Network architecture was designed by LeCun et al. [12] to classify handwritten digits between 0 and 9 using 3 convolutional, 2 max-pooling, and 2 fully connected layers. Figure from Lecun et al. [13].

The convolutional layer is obtained by an element-wise multiplication of a moving kernel or filter over the original input image. The step size of the kernel in the horizontal and vertical direction is known as the stride. The result of this convolution is known as the feature map and captures the most important features from the original input image. When more convolutional layers are used in the network, more detailed features can be filtered from the original image. The pooling layer is responsible for spatially reducing the size of the input, to reduce the number of parameters for computation in the network. The most common pooling layers are the max-pooling and average-pooling which act as a sliding window over the input image and take either the maximum or average value.

In the example in Figure 2.3, the first convolutional layer consists of a $5\times5$ kernel sliding over the original input image with a stride of 1. In this layer, 6 kernels are used which results in 6 feature maps. The second layer consists of a $2\times2$ max-pooling with a stride of 2. This results in a dimension reduction in the layer from $28\times28$ to $14\times14$. Then, another convolutional layer with a $5\times5$ kernel is used with 16 kernels, which results in 16 feature maps. These feature maps are down-sampled using max-pooling resulting in a dimension reduction from $10\times10$ to $5\times5$. Finally, the feature maps are all flattened and fed into the fully connected layers.

The basic principle of the CNN is to extract features using kernels in the convolutional layers and to down-sample the image using max-pooling layers to reduce the number of computations in the CNN. In general, for more complex problems, DNNs are frequently used which consist of multiple convolutional and pooling layers that can extract important high and low-level features.

## 2.3. Recurrent Neural Network

A Recurrent Neural Network (RNN) is a special type of neural network that is used for processing sequential data, for example, language translation, speech, or voice recognition. These types of networks leverage sequential modeling by sharing the same learning parameters over time, allowing the network to retain relevant memory over the sequence of data fed into the network. The architecture of an RNN is shown in Figure 2.4.

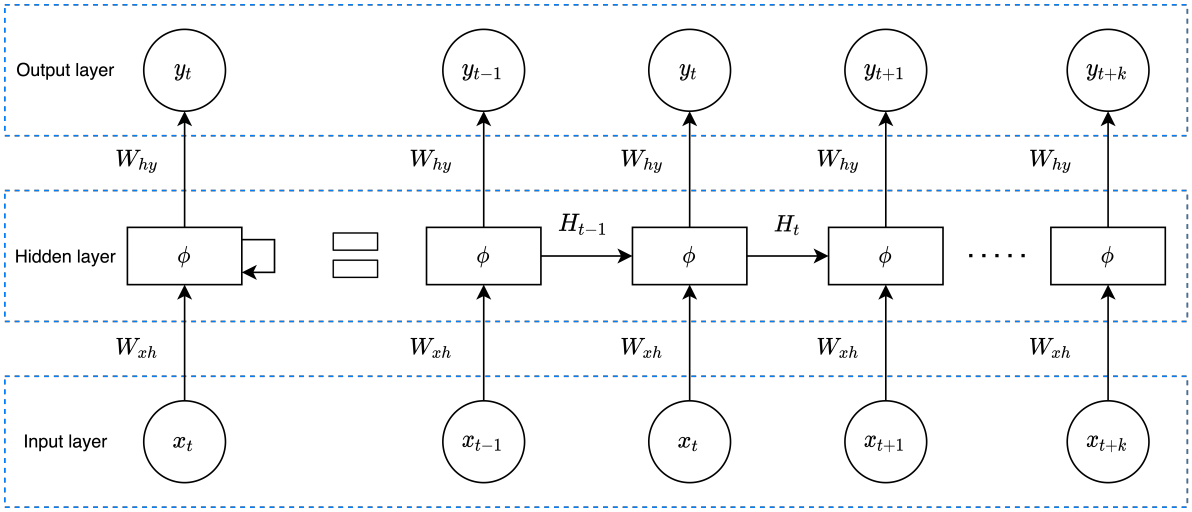


Figure 2.4: Example of a Recurrent Neural Network with a feedback connection in a folded on the left and unfolded situation on the right with $\boldsymbol{X_t}$ the input, $\boldsymbol{W}$ the weights of the network, $\boldsymbol{H_t}$ the output of the hidden layer and $\boldsymbol{Y_t}$ the output at time $t$, $\boldsymbol{H_{t-1}}$ the output of the hidden layer and $\boldsymbol{Y_{t-1}}$ the output at time $t-1$, and $\phi$ the non-linear activation function in the unit cell. Note that the biases are not included in the figure.

From the example shown the learning parameters from the hidden layer in the previous time step $\boldsymbol{H_{t-1}}$ are taken into account in the next time step by including the input of the current time step $\boldsymbol{X_t}$ together with $\boldsymbol{H_{t-1}}$. An advantage of an RNN is that the number of learning parameters does not grow with the sequence length. A general mathematical expression for an RNN is given by Equation 2.13.

$$\begin{aligned} \boldsymbol{H_t} &= \phi(\boldsymbol{X_t W_{xh}} + \boldsymbol{H_{t-1} W_{hh}} + \boldsymbol{b_h}) \\ \boldsymbol{Y_t} &= (\boldsymbol{H_t W_{hy}} + \boldsymbol{b_y}) \end{aligned} \tag{2.13}$$

Where $\boldsymbol{X_t}$ is the input, $\boldsymbol{W}$ the weights of the network, $\phi$ the non-linear activation function in the unit cell, $\boldsymbol{H_t}$ the output of the hidden layer, and $\boldsymbol{Y_t}$ the output.

RNNs are suitable for processing sequential data, however, when the sequence length becomes too long, the gradients tend to vanish or explode when back-propagating through time (BPTT). This is also called the gradient vanishing problem and occurs frequently when using an RNNs with a long sequence length [6].

### 2.3.1. Gated Recurrent Unit

The GRU network is an improved version for long-range relationship learning of the RNN, allowing the network to process long sequence length without encountering the vanishing gradient problem in the RNN. The main principle of a GRU is to introduce gates that can decide which information will be preserved in the next time step. The network architecture of the GRU is illustrated in Figure 2.5.

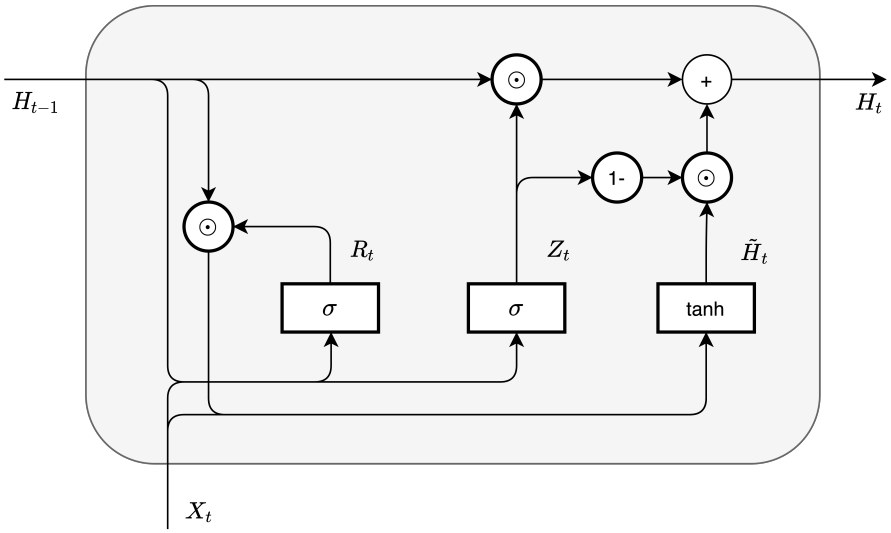


Figure 2.5: Example of a Gated Recurrent Unit with $\boldsymbol{R_t}$ the reset gate, $\boldsymbol{Z_t}$ the update gate, $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}$, the candidate hidden state, $\boldsymbol{H_t}$ the hidden state at time $t$, $\boldsymbol{H_{t-1}}$ the hidden state at time $t-1$, $\sigma$ the sigmoidal and hyperbolic tangent activation function, and $\odot$ the element-wise multiplication operator.

The GRU introduces two types of gates, the reset $\boldsymbol{R_t}$ and update gate $\boldsymbol{Z_t}$. The reset gate is used to ignore previous memories from the hidden layer that are irrelevant, whereas the update gate determines how much of the current input is passed through the network, which filters uninformative inputs. A general mathematical expression for the GRU is defined by Equation 2.14.

$$\begin{aligned}
\boldsymbol{R_t} &= \sigma(\boldsymbol{X_t W_{xr}} + \boldsymbol{H_{t-1} W_{hr}} + \boldsymbol{b_r}) \\
\boldsymbol{Z_t} &= \sigma(\boldsymbol{X_t W_{xz}} + \boldsymbol{H_{t-1} W_{hz}} + \boldsymbol{b_z}) \\
\tilde{\boldsymbol{H}}_{\boldsymbol{t}} &= \tanh(\boldsymbol{X_t W_{xh}} + (\boldsymbol{R_t} \odot \boldsymbol{H_{t-1}})\boldsymbol{W_{hh}} + \boldsymbol{b_h}) \\
\boldsymbol{H_t} &= \boldsymbol{H_{t-1}} \odot \boldsymbol{Z_t} + (1 - \boldsymbol{Z_t})\tilde{\boldsymbol{H}}_{\boldsymbol{t}}
\end{aligned} \tag{2.14}$$

Where $\boldsymbol{R_t}$ is the reset gate, $\boldsymbol{Z_t}$ the update gate, $\tilde{\boldsymbol{H}}_{\boldsymbol{t}}$ the candidate hidden state, $\boldsymbol{H_t}$ the hidden state at the current time step, $\boldsymbol{H_{t-1}}$ the hidden state at the previous time step, $\boldsymbol{X_t}$ the input, and $\odot$ the element-wise multiplication operator.

### 2.3.2. Long Short-Term Memory

The LSTM network is a version of the RNN with an additional memory cell in which information is linearly added or removed from the conveyor belt. The LSTM consists of a forget gate $\boldsymbol{F_t}$, input gate $\boldsymbol{I_t}$, candidate memory cell $\tilde{\boldsymbol{C}}_{\boldsymbol{t}}$, and output gate $\boldsymbol{O_t}$. The architecture of an LSTM network is illustrated in Figure 2.6.

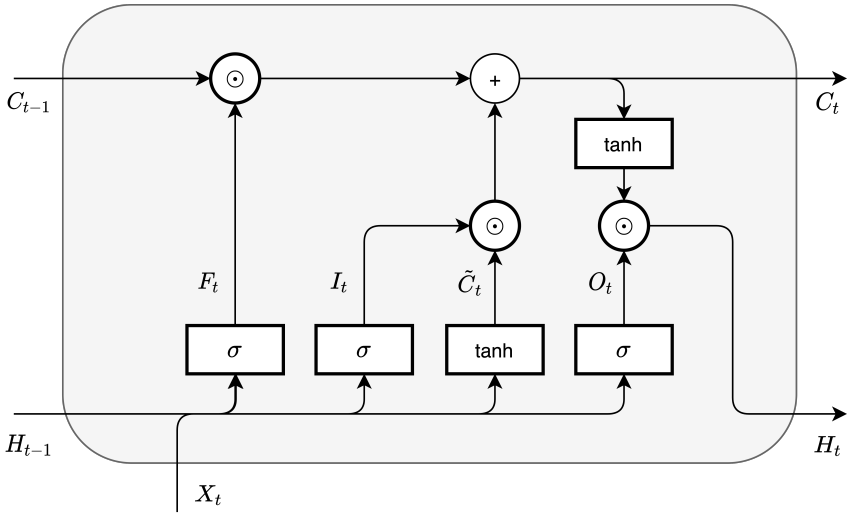


Figure 2.6: Example of Long Short-Term Memory network with $\boldsymbol{F_t}$ the forget gate, $\boldsymbol{I_t}$ the input gate, $\boldsymbol{\tilde{C}_t}$ the candidate memory cell, $\boldsymbol{O_t}$ the output gate, $\boldsymbol{H_t}$ the hidden state and $\boldsymbol{C_t}$ the memory at time $t$, $\boldsymbol{H_{t-1}}$ the hidden state and $\boldsymbol{C_{t-1}}$ the memory at time $t-1$, $\sigma$ the sigmoidal and hyperbolic tangent activation function, and $\odot$ the element-wise multiplication operator.

The key difference is that the LSTM network has three gates and includes a memory cell compared to the GRU network with two gates. The GRU network trains faster and performs better compared to an LSTM network on less training data, due to the less complex structure. However, in cases of large training data and longer learning sequences, the LSTM should achieve better performance due to its complex structure. A general mathematical expression for the LSTM network is defined by Equation 2.15.

$$\begin{aligned}
\boldsymbol{F_t} &= \sigma(\boldsymbol{X_t W_{xf}} + \boldsymbol{H_{t-1} W_{hf}} + \boldsymbol{b_f}) \\
\boldsymbol{I_t} &= \sigma(\boldsymbol{X_t W_{xi}} + \boldsymbol{H_{t-1} W_{hi}} + \boldsymbol{b_i}) \\
\boldsymbol{\tilde{C}_t} &= \tanh(\boldsymbol{X_t W_{xc}} + \boldsymbol{H_{t-1} W_{hc}} + \boldsymbol{b_c}) \\
\boldsymbol{C_t} &= \boldsymbol{F_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{I_t} \odot \boldsymbol{\tilde{C}_t} \\
\boldsymbol{O_t} &= \sigma(\boldsymbol{X_t W_{xo}} + \boldsymbol{H_{t-1} W_{ho}} + \boldsymbol{b_o}) \\
\boldsymbol{H_t} &= \boldsymbol{O_t} \odot \tanh(\boldsymbol{C_t})
\end{aligned} \tag{2.15}$$

Where $\boldsymbol{F_t}$ is the forget gate, $\boldsymbol{I_t}$ the input gate, $\boldsymbol{\tilde{C}_t}$ the candidate memory cell, $\boldsymbol{O_t}$ the output gate, $\boldsymbol{H_t}$ the hidden state and $\boldsymbol{C_t}$ the memory in the current time step, $\boldsymbol{H_{t-1}}$ the hidden state and $\boldsymbol{C_{t-1}}$ the memory in the previous time step, $\sigma$ the sigmoidal activation function, tanh the hyperbolic tangent activation function, and $\odot$ the element-wise multiplication operator.

Another variation of the LSTM is by including peephole connections in the architecture which is introduced by Gers et al. [14] and shown in Figure 2.7. The general mathematical expression is similar to Equation 2.15, however, with additional connections as defined by Equation 2.16.

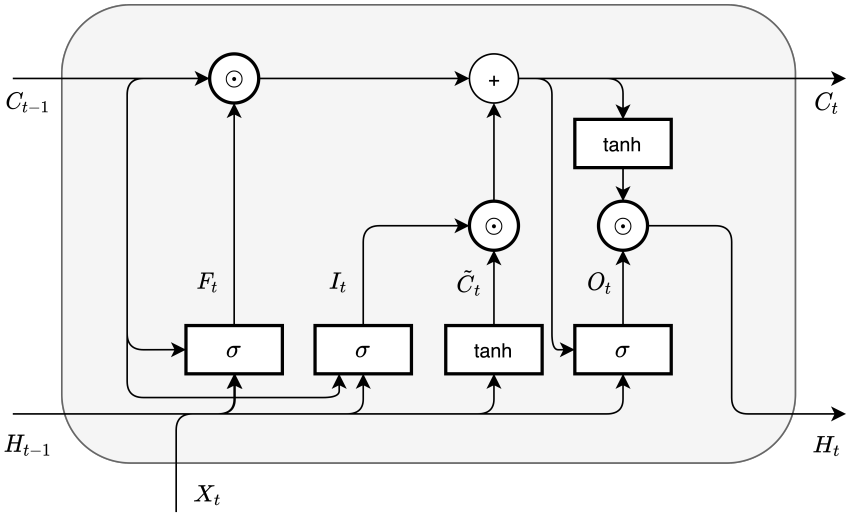


Figure 2.7: Example of Long Short-Term Memory network including peephole connections with $\boldsymbol{F_t}$ the forget gate, $\boldsymbol{I_t}$ the input gate, $\boldsymbol{\tilde{C}_t}$ the candidate memory cell, $\boldsymbol{O_t}$ the output gate, $\boldsymbol{H_t}$ the hidden state and $\boldsymbol{C_t}$ the memory at time $t$, $\boldsymbol{H_{t-1}}$ the hidden state and $\boldsymbol{C_{t-1}}$ the memory at time $t-1$, $\sigma$ the sigmoidal and hyperbolic tangent activation function, and $\odot$ the element-wise multiplication operator.

$$
\begin{aligned}
\boldsymbol{F_t} &= \sigma(\boldsymbol{X_t W_{xf}} + \boldsymbol{H_{t-1} W_{hf}} + \boldsymbol{C_{t-1} W_{cf}} + \boldsymbol{b_f}) \\
\boldsymbol{I_t} &= \sigma(\boldsymbol{X_t W_{xi}} + \boldsymbol{H_{t-1} W_{hi}} + \boldsymbol{C_{t-1} W_{ci}} + \boldsymbol{b_i}) \\
\boldsymbol{\tilde{C}_t} &= \tanh(\boldsymbol{X_t W_{xc}} + \boldsymbol{H_{t-1} W_{hc}} + \boldsymbol{b_c}) \\
\boldsymbol{C_t} &= \boldsymbol{F_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{I_t} \odot \boldsymbol{\tilde{C}_t} \\
\boldsymbol{O_t} &= \sigma(\boldsymbol{X_t W_{xo}} + \boldsymbol{H_{t-1} W_{ho}} + \boldsymbol{C_t W_{co}} + \boldsymbol{b_o}) \\
\boldsymbol{H_t} &= \boldsymbol{O_t} \odot \tanh(\boldsymbol{C_t})
\end{aligned}
\tag{2.16}
$$

A more common variation of the LSTM in image processing to include temporal coherence is the convLSTM introduced by Shi et al. [15]. In the convLSTM, the convolutional operators are denoted with a $\circledast$ defined by Equation 2.17.

$$
\begin{aligned}
\boldsymbol{F_t} &= \sigma(\boldsymbol{X_t} \circledast \boldsymbol{W_{xf}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hf}} + \boldsymbol{C_{t-1} W_{cf}} + \boldsymbol{b_f}) \\
\boldsymbol{I_t} &= \sigma(\boldsymbol{X_t} \circledast \boldsymbol{W_{xi}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hi}} + \boldsymbol{C_{t-1} W_{ci}} + \boldsymbol{b_i}) \\
\boldsymbol{\tilde{C}_t} &= \tanh(\boldsymbol{X_t} \circledast \boldsymbol{W_{xc}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{hc}} + \boldsymbol{b_c}) \\
\boldsymbol{C_t} &= \boldsymbol{F_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{I_t} \odot \boldsymbol{\tilde{C}_t} \\
\boldsymbol{O_t} &= \sigma(\boldsymbol{X_t} \circledast \boldsymbol{W_{xo}} + \boldsymbol{H_{t-1}} \circledast \boldsymbol{W_{ho}} + \boldsymbol{C_t W_{co}} + \boldsymbol{b_o}) \\
\boldsymbol{H_t} &= \boldsymbol{O_t} \odot \tanh(\boldsymbol{C_t})
\end{aligned}
\tag{2.17}
$$

# 3

# Objection Detection

Object detection is a technology in the field of computer vision to detect semantic objects in digital images and videos. Object detection has a wide range of applications in the field of computer vision, including activity recognition, object tracking, and autonomous navigation of robots. All these applications require sophisticated image processing techniques to obtain the required output. However, most of these applications are computationally expensive but expected to run on embedded systems with limited processing capabilities, which remains one of the challenges as of today.

In this chapter, different object detection techniques will be investigated. First, in section 3.1, the classical computer vision techniques in image processing will be discussed which have been used before the applications of deep learning in object detection. Then, in section 3.2 and section 3.3, the methods using deep learning in still images and their limitation are discussed. Finally, in section 3.4, the methods using deep learning in videos are presented.

## 3.1. Classical Computer Vision

In the early days, before the application of deep learning methods in the field of computer vision, object detection was dependent on feature extraction from images using classical computer vision-based methods. One of the object detection frameworks introduced in the early days was by Viola et al. [16] using a machine learning approach. This detection framework can be applied to different object detection classes, however, their main application was to solve the face detection problem. In their work, Haar-like features which consist of simple rectangle features are calculated using integral images by Equation 3.1 allowing for fast computations. The integral image is calculated by considering a certain point in the image ($x$', $y$') and summing all the pixel values on the left and above the point. Then, the AdaBoost framework is used to select the most critical visual features. This detection framework has been applied for face detection, however, it is not able to deal with different object pose variations. For instance, when a person is turning his head, the features can not be matched with the face, and therefore the detection framework tends to fail. Another shortcoming is that features have to be selected for specific objects. This means that different features have to be selected for different object detection applications. For example in real-world applications multiple objects will appear on an image, hence a single feature is not able to detect every object.

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.1}$$

Scale Invariant Feature Transform (SIFT) is a feature detection algorithm by Lowe [17] that extracts features from an object based on a reference image and is stored in a large database. The extracted features from the database are compared with the features on a new image for potential matching candidates. SIFT features are invariant to scaling and rotation, due to their key feature matching properties in images and are robust against distortion, view changes, illumination changes, and noise. However, the main disadvantage of using SIFT features is that it is computationally expensive. This has led to several alternatives, in which the most known is the Speeded Up Robust Features (SURF) by Bay et al. [18] that is 3 times faster than the SIFT algorithm due to the application of integral images and box filters [19].

Corner detection is an algorithm by Harris et al. [20] used in computer vision to extract certain features from images, which is frequently used in applications such as object recognition, motion detection, and video tracking. Later Shi et al. [21] made a small modification in the work of Harris et al. [20] by adapting the scoring function which resulted in better performance in corner detection.

Histogram of Oriented Gradients (HOG) is another algorithm used in object detection by Dalal et al. [22] which focused on the structure and shape of the object. The principle behind the HOG algorithm is to convert pixels to gradient-based oriented representations in localized regions. Then, histograms are generated containing the values of the pixel and their gradient. The HOG algorithm is invariant to geometric and photometric transformation but is not invariant to the rotation of objects.

## 3.2. Detection in Images

Object detection in images is an interesting field in which a lot of research has been done since the introduction of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2010. The main goal of this challenge is to develop an algorithm for object detection and image classification. Since the introduction of deep learning in computer vision-related tasks, researchers have been using deep learning methods in the field of object detection as they have proven to achieve good performance on a large scale in object detection and image classification. In particular, CNNs as they support end-to-end learning as opposed to manually designing a suitable feature extractor.

### 3.2.1. One-Stage Object Detection

One-stage object detection algorithms are using a single CNN to obtain a prediction of the location of the object in an image. The main principle of these types of detectors is that an input image is given to the CNN, the prediction of the network is a bounding box indicating the location of the object in the image. One of the first CNN was introduced by LeCun et al. [12], by designing a CNN that can classify handwritten digits. This has proven the power of CNNs with a relatively simple architecture that can achieve high accuracy in the classification of images with handwritten digits.

Recently, deep learning methods have been applied extensively in the field of object detection on large-scale image datasets. In 2012 AlexNet by Krizhevsky et al. [23] won the ILSVRC. The CNN was designed to classify 1.2 million high-resolution images with 1000 object classes with approximately 60 million learning parameters. However, due to a large number of learning parameters, overfitting was considered a problem. In their work, they addressed the overfitting problem by implementing two methods in their network. The first method consists of data augmentation by generating image translation, horizontal reflections, and modifying the intensity of the RGB channels by performing Principal Component Analysis (PCA) on the images. The second method consists of using dropout in the CNN, which is a technique used to artificially turn off certain neurons in the forward pass of the network with a predetermined probability [23].

Many attempts have been made to improve the original CNN architecture AlexNet. In the work of Zeiler et al. [24] a visualization technique is applied to provide novel insight into the activation of the intermediate layers in the network using deconvolutional networks, to discover new model architectures that achieve higher performance than AlexNet. In the work of Sermanet et al. [25], the OverFeat network is introduced, in which two main modifications are made from the AlexNet. In the first method, a smaller receptive field is utilized. The first convolutional layers are identical to AlexNet except for a smaller stride. A larger stride in the convolutional layers is beneficial for speed, however, it deteriorates the accuracy of the network. The second method is by using multi-scale classification to minimize redundant computations in overlapping regions.

The contributions of Zeiler et al. [24] and Sermanet et al. [25] have led to a better understanding of the effect of the CNN architectures in performance and accuracy. However, the effect of the depth of a CNN is not well understood and has not been studied at this point. In the work of Simonyan et al. [26] another important aspect of the CNN architecture has been investigated, what is the effect of the depth of the CNN. The network that they have designed, by varying the depth of the layers is referred to as the VGG network. In their research, the hyperparameters are fixed and the depth of the network is increased by adding more convolutional layers limited to 19 layers. The convolutional filters consist of a 3×3 kernel with a stride of 1. The results have shown the importance of increasing the depth in CNN architectures which contribute towards a higher accuracy for large-scale object detection and image classification, outperforming previous networks [24, 25] in the ILSVRC with a substantial margin [26].

Another object detection method that has proven to achieve good performance is the YOLO network by Redmon et al. [27]. The main principle behind the YOLO network is to divide the image into an S×S grid. Each grid is responsible for detecting an object if it exists within the grid cell. Then, each grid cell predicts the bounding boxes including a confidence score of the class of the object. The confidence score acts as a metric for the accuracy of the model's prediction. The confidence score is defined as the intersection over union (IoU) by Equation 3.2 given a certain threshold, which is a metric that determines the correlation between the ground truth and the prediction. In addition, the precision and recall metric are also used which is given by Equation 3.3 and 3.4.

$$\text{IoU} = \frac{TP}{TP + FP + FN} \tag{3.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.4}$$

One of the issues with previous networks is that the main focus was on the accuracy of the detectors rather than the inference speed to be run on embedded systems in real-time. Therefore, the YOLO network addressed this problem by specifically designing a real-time object detection system, which achieves high inference speed that can be deployed on embedded systems with limited hardware capabilities. For this purpose, the Fast YOLO network has been designed which is a smaller version of the YOLO network by reducing the number of depth layers. In addition, several improvements have been made upon the YOLO network to increase the performance both in detection accuracy and inference speed [27, 28].

The network from Krizhevsky et al. [23] and Sermanet et al. [25] have been used as backbones for new improved architectures with better performance. One example is the SSD network by Liu et al. [29] shown in Figure 3.1, which generates bounding boxes with a prediction score of the presence of an object class in the bounding boxes using non-maximum suppression (NMS) [30]. The principle behind NMS is to select the bounding box with the largest prediction score and eliminate the remaining bounding boxes. The SSD network uses the VGG network from Sermanet et al. [25] as the backbone network and adds extra feature layers. The key difference compared to the YOLO network is that the SSD network uses multi-scale feature maps for detection.

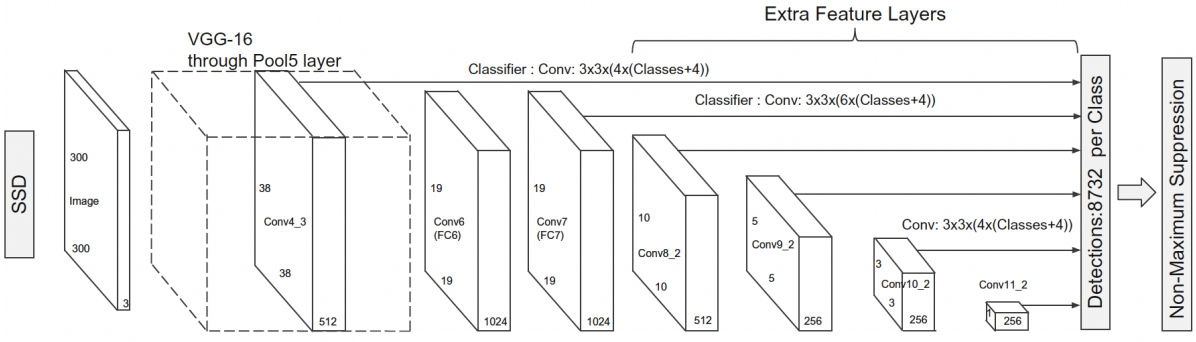


Figure 3.1: The SSD network architecture uses the VGG network as a backbone with additional feature layers. The SSD network uses multi-scale feature maps for object detection followed by Non-Maximum Suppression. Figure adapted from Liu et al. [29].

The presented networks are popular choices for object detection applications due to the simplicity of the network. Although the networks are tailored for large-scale image object detection, these networks can easily be modified for less complicated object detection tasks. For example, the networks can be modified for single object detection in applications such as autonomous navigation of Micro Air Vehicles (MAVs), in particular, in the field of Autonomous Drone Racing (ADR). Due to the relatively small network of one-stage object detectors, attempts have been made to implement the SSD network in ADR, specifically for gate detection [4,

31–33]. This has proven that these types of networks achieve decent gate detection accuracy and inference speed on MAVs in real-time.

### 3.2.2. Two-Stage Object Detection

Two-stage object detection algorithms are using two CNNs. The first CNN generates region proposals containing the objects. Then, the second CNN generates a prediction of the object in the image based on the proposed regions. These types of object detectors are solving two tasks in subsequent order and are therefore referred to as two-stage object detectors.

In the work of Girshick [34] a Fast Region-based Convolutional Neural Network method (R-CNN) is proposed for object detection. Although this method is still a one-stage object detector it uses a proposal of Region of Interest (RoI) pooling layer to obtain a feature vector from the feature map that is generated from the convolutional layers. This feature vector is then fed into the fully connected layers to generate a prediction of the object. The network Faster R-CNN by Ren et al. [35] is an improvement upon the Fast R-CNN network by separating object detection into two stages. The first network generates region proposals of the object of interest and the second network uses the region proposals to localize the object. A general overview of the Faster R-CNN architecture is shown in Figure 3.2.

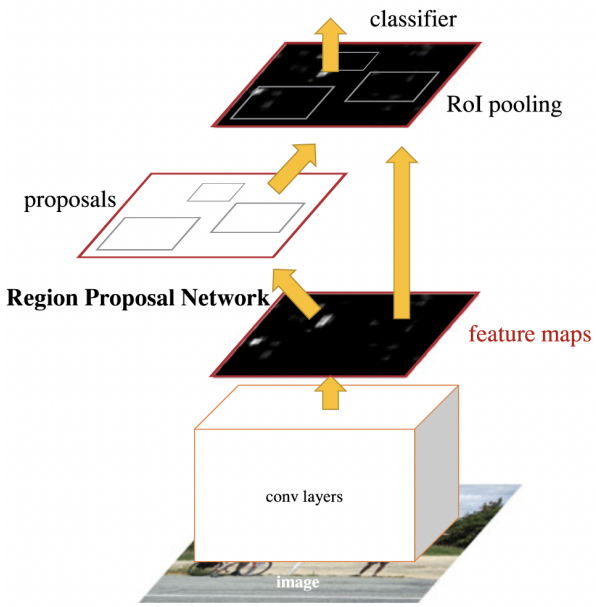


Figure 3.2: The Faster R-CNN network architecture by Ren et al. [35] consists of two stages. The first stage consists of a Region Proposal Network for detection proposal which is then used in the second stage of the network to obtain a prediction of the object. Figure from Ren et al. [35].

In the first stage a Region Proposal Network (RPN) is used which is a fully connected CNN generating detection proposals. Then, in the second stage, the obtained proposals from the RPN are used and fed into the Fast R-CNN to detect the objects. Another detection algorithm is the Mask R-CNN by He et al. [36], which primarily is an extension of the Faster R-CNN by generating segmentation masks of the objects. The Mask R-CNN differs from the Faster R-CNN by adding a branch to obtain a prediction for masked objects.

The two-stage networks are popular choices for object detection applications in which high performance and accuracy are required. These two-stage object detection methods have a more complicated architecture in comparison to one-stage object detection methods. From the perspective of real-time object detection on embedded systems with limited hardware capability, this introduces problems. Due to their complicated architectures and the large number of learning parameters in the network, often these types of object detectors achieve moderate to poor inference speed in real-time. Although the inference speed is more limited in two-stage object detection methods, attempts have been made by applying these object detectors on MAVs for gate detection [37]. However, from the perspective of high-speed ADR, high inference speed in real-time is of utmost importance to perform accurate gate detection, hence these types of networks are considered computationally expensive to be deployed on MAVs.

## 3.3. Limitations of Detection in Images

The presented one-stage and two-stage object detection networks are designed to be trained on still images which means that only spatial information is considered and the temporal coherence is neglected. The latter is an important aspect for object detection in real-time as frames are strongly correlated and provide salient information. Hence, training the network on still images has several limitations in which a common problem is that the detection accuracy deteriorates in the case of motion blur, occlusion, and pose variations of objects in images as shown in the examples in Figure 3.3.


(a) (b) (c)

Figure 3.3: Example of situations in which the accuracy of gate detection deteriorates when the network is trained on still images. Figure (a) shows a situation in which motion blur is present. Figure (b) shows a situation in which the front gate is occluding the gate behind. Figure (c) shows a situation in which the front gate is in a rare pose due to the relatively short distance from the front gate.

From the examples, Figure 3.3 (a) represents a situation in which motion blur occurs in the frames from the onboard camera of the MAV. Depending on the severity of the motion blur, the accuracy of the gate detection may deteriorate. In Figure 3.3 (b) a situation is presented in which the front gate is occluding the gate behind. Due to this situation, a frequent problem is that the gate behind is not detected since the front gate is partially blocking the gate behind the front gate. In Figure 3.3 (c) a situation is shown in which the MAV is relatively close to the gate which causes the gate to be partially visible and thus the pose of the gate is changed which is also a frequent problem in which the detector is not able to successfully detect the gate.

Another limitation that has an impact on the accuracy of the detection is due to training the network. Most object detectors are trained with a clean dataset, meaning that objects in images are usually perfectly centered, do not contain motion blur, occlusion, and pose variations. However, when applying these detectors to videos with a sequence of frames that consist of temporal information, these types of object detectors tend to fail and achieve poor performance [38]. This problem is partially addressed by augmenting the dataset and including images that contain these properties as opposed to clean datasets. However, when considering that the sequence of frames obtained from cameras in real-world applications are strongly correlated and contain valuable information over time, the previously mentioned solution is not optimal. Therefore, a logical step is to include information over time to improve performance.

## 3.4. Detection in Videos

Object detection in videos is a relatively new field in which not a lot of research has been done. In the past, object detection methods were solely based on still images. However, since the introduction of the ImageNet VID challenge in 2015 more research has been done in object detection in videos. The main goal of this challenge is to develop an algorithm for object detection in videos, which is an extension of the previous challenge in object detection for still images. Object detection in videos have proven to be challenging since traditional object detection methods based on still images do not perform well on videos, due to motion blur, occlusion, and pose variations of objects that frequently occur in videos resulting in large temporal fluctuations in the confidence score of the detection as shown in Figure 3.4.

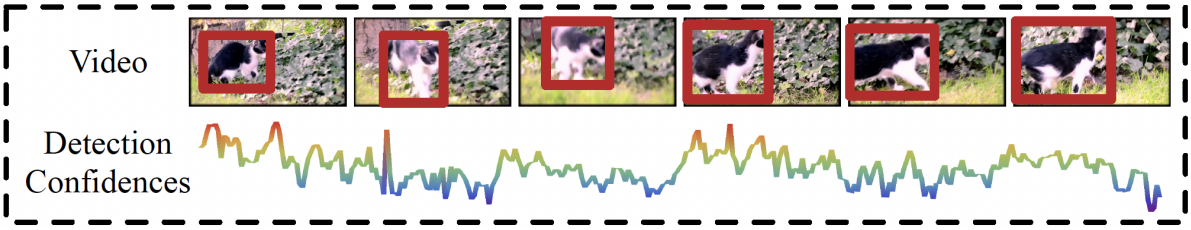


Figure 3.4: Limitations of using a still image object detector on a sequence of images. The detection contains large temporal fluctuation in the detection confidence score mainly due to motion blur and pose variations of the cat in the image. This illustrates the lack of temporal coherence in still image object detection algorithms. Figure from Kang et al. [38].

### 3.4.1. Post Processing

The performance of image object detection algorithms has significantly improved over the past years. However, applying these algorithms to videos remains a challenge. One of the methods that have been applied to increase the performance of image object detection algorithms is Seq-NMS by Han et al. [39] which is similar to the traditional NMS method but extended in the temporal domain. The main principle of Seq-NMS is to include object box proposals from adjacent frames to boost weak detections with the assumption that adjacent frames have similar objects and that bounding boxes are in similar positions and sizes.

Another method to improve temporal consistency is T-CNN by Kang et al. [38] who proposed a deep learning framework based on Faster R-CNN [35] using tubelet re-scoring by incorporating temporal information from adjacent frames. The main difference between Seq-NMS by Han et al. [39] and the T-CNN by Kang et al. [38] is that Kang et al. [38] implemented a tracking algorithm that tracks an object that has a high confidence score in the tubelet, which is then used to improve the confidence score of the detected objects in the video by reducing false positives.

Another option that is used in object detection in videos is extending the algorithms with post-processing methods to improve the detection accuracy. For example, Recurrent Convolutional Neural Network (RCNN), optical flow and 3D CNN as discussed in subsection 3.4.3, subsection 3.4.2, and subsection 3.4.4 respectively, can be extended by using post-processing methods such as Seq-NMS to improve the detection accuracy.

The methods rely on the application of still image object detectors and use post-processing methods for further improving the detection accuracy in videos. However, they do not explicitly use information over time on a sequence of images, but rather focus on the confidence score predictions of the network. Therefore, post-processing methods do not contribute directly towards incorporating salient temporal information in object detection in videos, however, it does improve the accuracy of detection slightly compared to traditional image object detectors without the application of post-processing methods.

### 3.4.2. Recurrent Convolutional Neural Network

One method to include temporal coherence is to use an RCNN. In the work of Ning et al. [40], a new approach for spatially supervised RCNN for object tracking is proposed. Their method is two-fold, first, an input sequence of images is fed into the YOLO CNN to extract visual features for detection then the features are fed into an LSTM network for sequential modeling and tracking the location of the object. The main benefit of the LSTM network over the traditional RNN is that it can capture long-term temporal dependencies. Although object tracking is different from object detection, the border between the two starts to fade when information over time is taken into account. Another relevant work by Valipour et al. [41] uses image segmentation for object detection. Similarly, a sequence of frames is used as input to a fully convolutional network (FCN) and RNN with up-sampling at the end of the network to obtain the desired spatial size for semantic segmentation as illustrated in Figure 3.5.

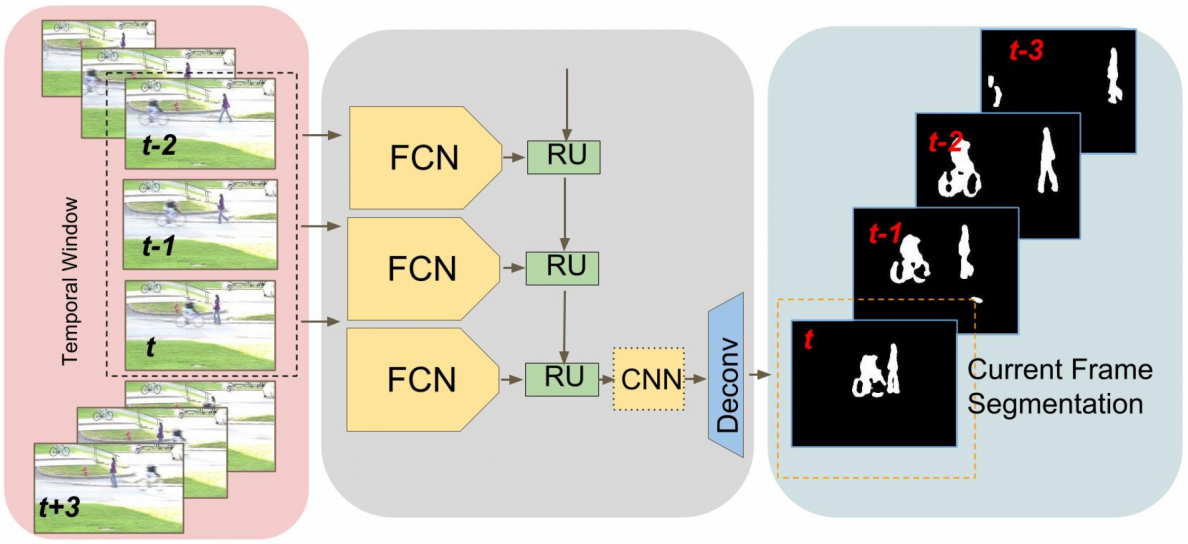


Figure 3.5: A representation of a Recurrent Convolutional Neural Network generating a segmentation mask of the objects. An input sequence is passed through a Fully Convolutional Network and Recurrent Neural Network to include temporal coherence. Figure from Valipour et al. [41].

More recently, research has been done by Wang et al. [42] by studying the effect of including noise in the input sequence on the accuracy of video semantic segmentation using a similar network structure as previously described, but with a convLSTM [15]. Different from the traditional LSTM, the convLSTM includes convolutional operators which are more suitable for processing images. Results have shown that training the network with noise yields improvements in segmentation performance [42]. Besides, many variations of these types of concepts have been made [43–45].

The previously described methods process the frames one by one and use an RNN to apply sequential modeling. The traditional LSTM is a powerful method to handle temporal correlation, however, when considering the spatial domain it contains redundant information, which increases the computational load of the network. An alternative strategy might be to only consider sparse frames in the temporal window to decrease the computational load.

### 3.4.3. Optical Flow

Optical flow is the apparent motion in a specific scene that is caused by the relative motion between the observer and the scene. Optical flow is frequently used in the field of computer vision as it provides useful information about the relative motion of objects that can be used for motion perception and object detection. Optical flow contains information about the type of motion, motion speed, distance covered, heading, the relative depth of objects, and time-to-contact (TTC). The expression describing the relationship between the movement of the camera and optical flow [46] is derived from the reference system illustrated in Figure 3.6 and given by Equation 3.5 and 3.6.

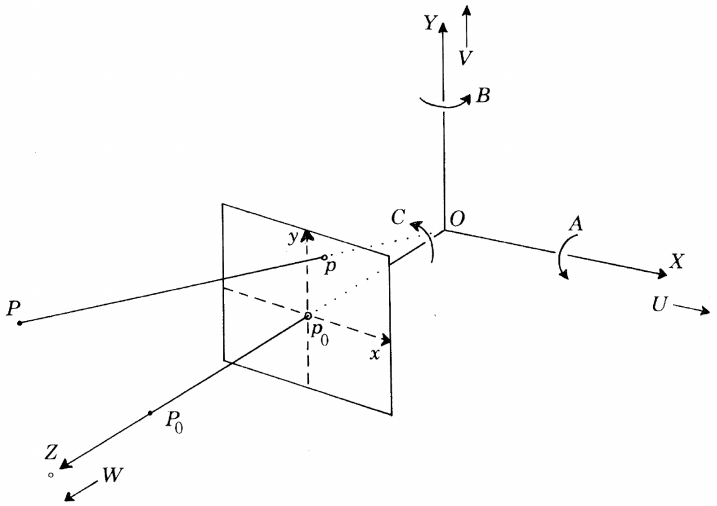


Figure 3.6: Reference system between the movement of the camera and optical flow to derive the optical flow equations. Figure from Longuet-Higgins et al. [46].

$$u = (-\frac{U}{V} - B + Cy) - x(-\frac{W}{Z} - Ay + Bx) = u_T + u_R \tag{3.5}$$

$$v = (-\frac{V}{Z} - Cx + A) - y(-\frac{W}{Z} - Ay + Bx) = v_T + v_R \tag{3.6}$$

Where $(u, v)$ is the optical flow associated to an image $(x, y)$, $(U, V, W)$ the translational velocities, $(A, B, C)$ the rotational rates and $(X, Y, Z)$ the coordinates in the real-world. From the relationship between the movement of the camera and the optical flow, it can be observed that it consists of a translational $(u_T, v_T)$ and rotational $(u_R, v_R)$ components.

When assuming that the rotational rates can be measured using gyroscopes and that the rotational components of the optical flow can be canceled, the final relationship describing the optical flow is given by Equation 3.7 and 3.8.

$$u = -\frac{U}{V} + x\frac{W}{Z} \tag{3.7}$$

$$v = -\frac{V}{Z} + y\frac{W}{Z} \tag{3.8}$$

Only the translational components are remaining which can be used to obtain important information, for example, the Focus of Expansion (FoE). At this point, the optical flow is zero, hence the FoE is described by Equation 3.9.

$$\begin{aligned} x_{FoE} &= \frac{U}{W} \\ y_{FoE} &= \frac{V}{W} \\ \frac{x_{FoE}}{y_{FoE}} &= \frac{U}{V} \end{aligned} \tag{3.9}$$

Using the expression above, the relative velocity in the direction of the principal axis of the camera can be calculated with Equation 3.10. Note that this relative velocity is inversely related to the TTC described by Equation 3.11.

$$\frac{W}{Z} = \frac{u_t}{(x - x_{FoE})} = \frac{v_t}{(y - y_{FoE})} \tag{3.10}$$

$$\tau = \frac{Z}{W} \tag{3.11}$$

To incorporate information over time in the application of object detection, optical flow provides a solution as it can capture the motion of objects between two consecutive frames. In the work of Zhu et al. [47], Deep Feature Flow (DFF) is introduced, which is a framework for video object detection using convolutional sub-networks on sparse key frames and propagating deep feature maps to adjacent frames using flow fields calculated using FlowNet by Dosovitskiy et al. [48]. The main principle of this network is to obtain spatial information using deep feature maps and combining motion information using optical flow to propagate the extracted feature maps from the key frames to the current frame using flow fields, which reduces the number of computations.

Another optical flow method is the flow-guided feature aggregation (FGFA) by Zhu et al. [49]. In their work, they observed that nearby frames have a high response and therefore the feature maps can be propagated to the reference frame to enhance the detection accuracy similar to Zhu et al. [47]. In this method, motion-guided spatial warping is performed using the calculated flow fields to propagate the feature maps from the key frames to the current frame. Then, a feature aggregation module is used which combines the feature maps from adjacent frames and the current frame to obtain an aggregated feature map that provides a higher response, increasing the detection accuracy. An overview of the FGFA method by Zhu et al. [49] is shown in Figure 3.7.

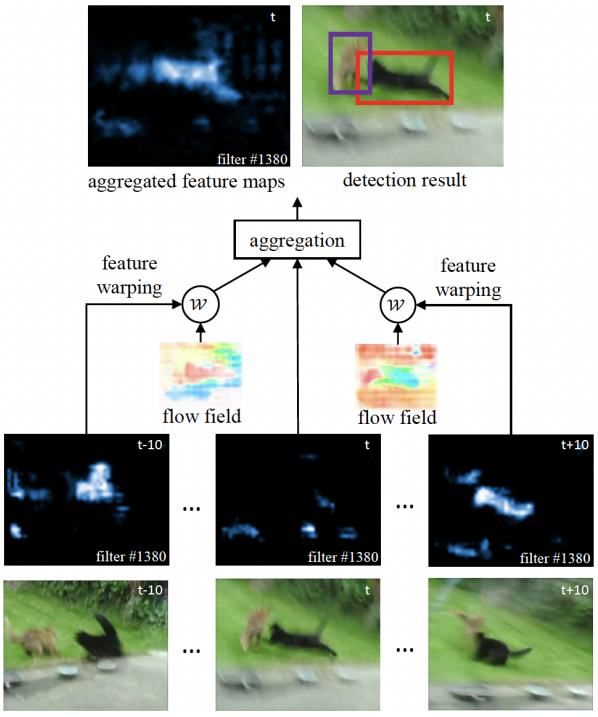


Figure 3.7: The flow-guided feature aggregation method uses adjacent frames to enhance the detection accuracy for frames with weak detection by warping the feature maps from the adjacent frames to the current frame using the calculated flow fields from FlowNet [48]. Figure from Zhu et al. [49].

From Figure 3.7, the current frame contains motion blur which is the main reason for the deep feature map of the current frame containing low activation and hence results in no detection. However, assuming that object motion is relatively small, adjacent frames can be used to retrieve important features which can be used to improve the detection accuracy of the current frame even in the case of severe motion blur as shown in Figure 3.7.

The FGFA method uses calibration on a feature level and addresses the issue regarding motion blur. The limitation of the FGFA is that when the appearance of the object is changing drastically, the flow field estimation will be inaccurate and thus the propagated feature maps using spatial warping are not estimated correctly, which is often the case when objects are occluded. This problem is addressed in the work of Wang et al. [50] by extending the FGFA method with an extra module that calibrates features on an instance level called MANet. In their method, both the pixel-level (feature level) and instance-level calibration are used jointly for detection. The results show that pixel-level aggregation provides better detection accuracy in the case of non-rigid motion patterns which happen in fast motion of objects containing motion blur as a result. On the other hand, instance-level calibration provides better detection accuracy in the case of objects being occluded. This is confirmed by the results in Table 3.1 showing that pixel-level calibration contributes towards an increase in mean average precision (mAP) for fast-moving objects and instance-level calibration an increase in mAP for slow and medium moving objects.

Table 3.1: The performance of multi-frame feature aggregation, pixel-level calibration, and instance-level calibration on the ImageNet VID validation dataset evaluated for slow, medium, and fast-moving objects. The different methods are using the ResNet-101 network for feature extraction. Table adapted from Wang et al. [50].

| Feature Extractor | ResNet-101 | | | | |
|---|---|---|---|---|---|
| Methods | (a) | (b) | (c) | (d) | (e) |
| Multi-frame feature aggregation | | ✓ | ✓ | ✓ | ✓ |
| Pixel-level calibration | | | ✓ | | ✓ |
| Instance-level calibration | | | | ✓ | ✓ |
| mAP (%) | 73.6 | 73.4 | 75.6 | **77.1** | 78.1 |
| mAP (%) (slow) | 81.8 | 83.8 | 85.0 | **85.5** | 86.9 |
| mAP (%) (medium) | 71.3 | 75.7 | 74.9 | **76.1** | 76.8 |
| mAP (%) (fast) | 52.2 | 45.2 | **56.6** | 55.4 | 56.7 |

Both methods consider an optical flow approach for solving the object detection problem in videos. When post-processing methods are included such as Seq-NMS by extending the current optical flow methods, the detection accuracy can be increased. In Table 3.2 the performance of both optical flow methods based on the ImageNet VID validation dataset is shown. The results show that both methods when extended with the post-processing method Seq-NMS provide a slight increase in mAP.

Table 3.2: The performance of optical flow methods used in object detection in videos is measured on the ImageNet VID validation dataset. The * indicates that the method includes the post-processing method Seq-NMS. Table adapted from Zhu et al. [49] and Wang et al. [50].

| Method | Backbone | mAP (%) | mAP* (%) |
|---|---|---|---|
| FGFA | ResNet-101 | 76.3 | **78.4** |
| MANet | ResNet-101 | 78.1 | **80.3** |

The described methods show the potential of using optical flow to capture salient motion information between frames and warping the feature maps to the current frame to increase the detection performance. However, the flow field is calculated on a feature level whereas it is more interesting to capture the motion on an object level. Moreover, the output of the methods is a bounding box that depicts the object of interest in a rather inaccurate way. Generating segmentation masks would be more interesting as it can capture the object more accurately on a pixel level using a binary segmentation mask. Besides, the optical flow methods described do not provide any relevant depth information. Hence, from a temporal perspective, more relevant information can be retrieved using optical flow such as divergence, FoE, and depth information.

Additionally, calculating the optical flow might be computationally expensive depending on the optical flow network, especially for real-world applications with limited hardware capabilities. In the work of Kong et al. [51], a comparison is done between different state-of-the-art optical flow methods indicating the performance and inference speed on a desktop NVIDIA GeForce GTX 1080 Ti GPU and embedded NVIDIA Jetson TX2 on the Sintel and KITTI 2012 dataset. Note that different optical flow methods are all fine-tuned in the same way and trained with the same dataset for a fair comparison [51]. The results in Table 3.3 show that

PWC-Net, LiteFlowNet, and FastFlowNet have the fastest inference speed on an embedded NVIDIA Jetson TX2. Therefore, using a lightweight optical flow network is a good option to reduce the number of computations on embedded systems to capture motion.

Table 3.3: A comparison of the performance and inference speed of different state-of-the-art optical flow methods on a desktop NVIDIA GeForce GTX 1080 Ti GPU and embedded NVIDIA Jetson TX2. The comparison of the different fine-tuned optical flow methods is done on both the Sintel and KITTI 2012 dataset. The indicated default metric used in the table is the end-point error (EPE). Figure adapted from Kong et al. [51].

| | Sintel | | Sintel Final | | KITTI 2012 | | Parameters | FLOPs | Time (s) | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | train | test | train | test | train | test | (M) | (G) | 1080Ti | TX2 |
| FlowNet2-ft [52] | 1.45 | 4.16 | 2.01 | 5.74 | 1.28 | 1.8 | 162.52 | 24836.4 | 0.116 | 1.547 |
| SpyNet-ft [53] | 3.17 | 6.64 | 4.32 | 8.36 | 4.13 | 4.7 | 1.20 | 149.8 | 0.050 | 0.918 |
| PWC-Net-ft [54] | 2.02 | 4.39 | 2.08 | 5.04 | 1.45 | 1.7 | 8.75 | 90.8 | 0.034 | **0.485** |
| LiteFlowNet-ft [55] | 1.35 | 4.54 | 1.78 | 5.38 | 1.05 | 1.6 | 5.37 | 163.5 | 0.055 | **0.907** |
| FastFlowNet-ft [51] | 2.08 | 4.89 | 2.71 | 6.08 | 1.31 | 1.8 | 1.37 | 12.2 | 0.011 | **0.176** |

### 3.4.4. 3D Convolutional Neural Network

2D CNNs have been used frequently in object detection tasks in still images, however, these types of networks only utilize the spatial domain. In video-related tasks, in which frames are highly correlated with adjacent frames, salient temporal information can be used to increase the performance and accuracy of object detection in videos. In the work of Tran et al. [56] an efficient approach for spatial-temporal learning is proposed by using 3-dimensional convolutional layers to solve action recognition problems in videos. It was found that a 3×3×3 kernel for all layers achieved the best result in all the explored architectures by Tran et al. [56] and outperformed various 2D CNN networks in video analysis tasks. A visual representation of a 3D CNN is shown in Figure 3.8.

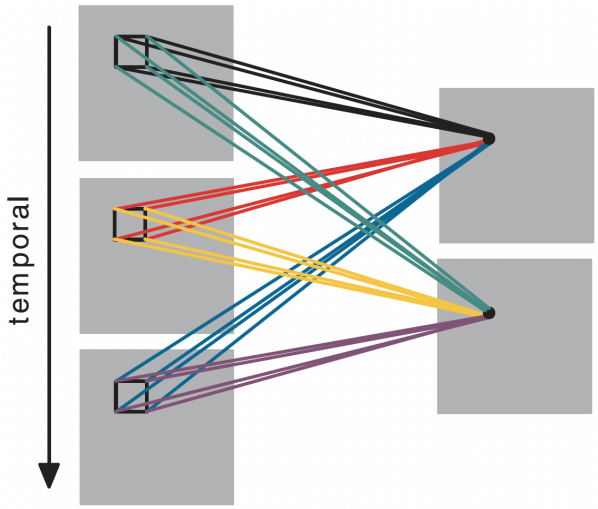


Figure 3.8: A representation of a 3D Convolutional Neural Network. In comparison with a 2D Convolutional Neural Network, an additional dimension time is incorporated. Figure from Ji et al. [57].

3D CNN has also been applied in object detection in videos by Dong et al. [58]. In their work, they describe the limitation of computing power in devices as surveillance cameras and smartphones for supporting existing salient object detection methods. In contrast to the method described by Tran et al. [56], both 2D and 3D CNNs are used to represent the spatial and temporal features called 3D convolution-based X-shape structure by generating a segmentation mask which is illustrated in Figure 3.9.

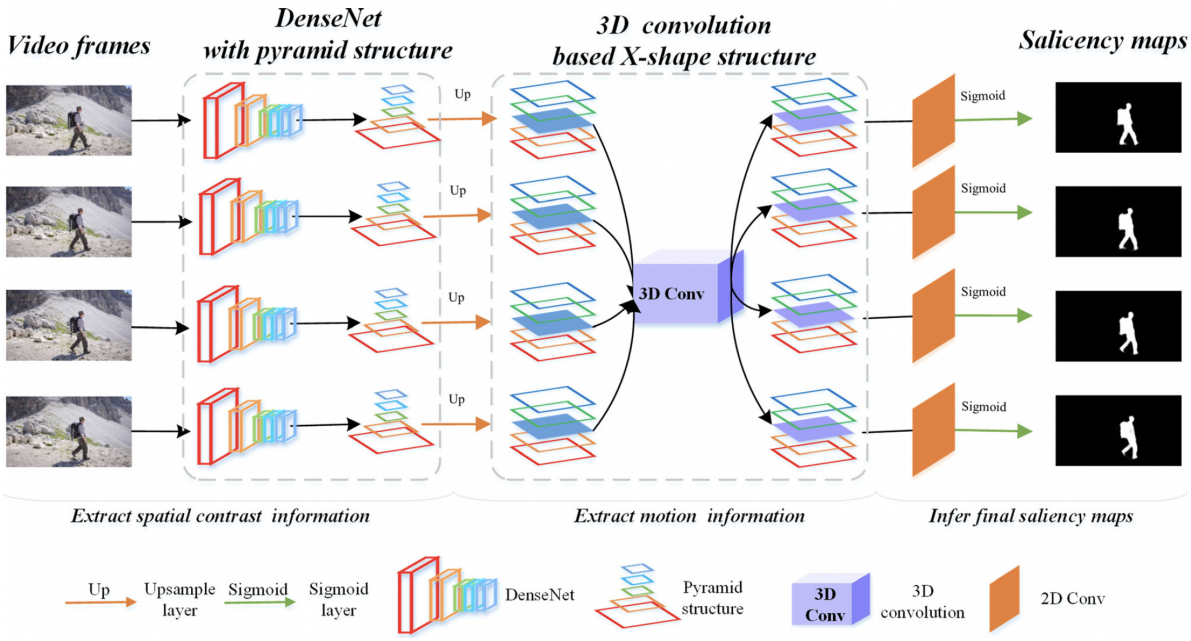


Figure 3.9: A representation of the network architecture by Dong et al. [58] using both 2D and 3D CNN X-shape structure to generate a segmentation mask of the object of interest. Figure from Dong et al. [58].

Another application of 3D CNN has been used for fall detection in the sector of public healthcare by Lu et al. [59]. In their work, a 3D CNN is combined with a modified version of a soft attention-based LSTM network [60, 61] for motion estimation in video sequences. The main principle of this method is that a 3D CNN is used to generate motion-based feature maps which are then fed into a soft attention-based LSTM network that can preserve both spatial and temporal information and locate the RoI for each frame in the video for fall detection. A similar approach has been done by Akilan et al. [62]. In their work, a 3D CNN and LSTM network is used to create a segmentation mask for object detection. They compared the number of learning parameters with a traditional convLSTM and found that replacing a convLSTM with a 3D CNN reduces the number of learning parameters by 84% [62].

### 3.4.5. Multi-Stream Network

Another way to perform object detection is by utilizing multiple stream networks. In the work of Siam et al. [63], MODNet, a two-stream network is used for object detection and motion segmentation by using an RGB image and optical flow channel as shown in Figure 3.10.

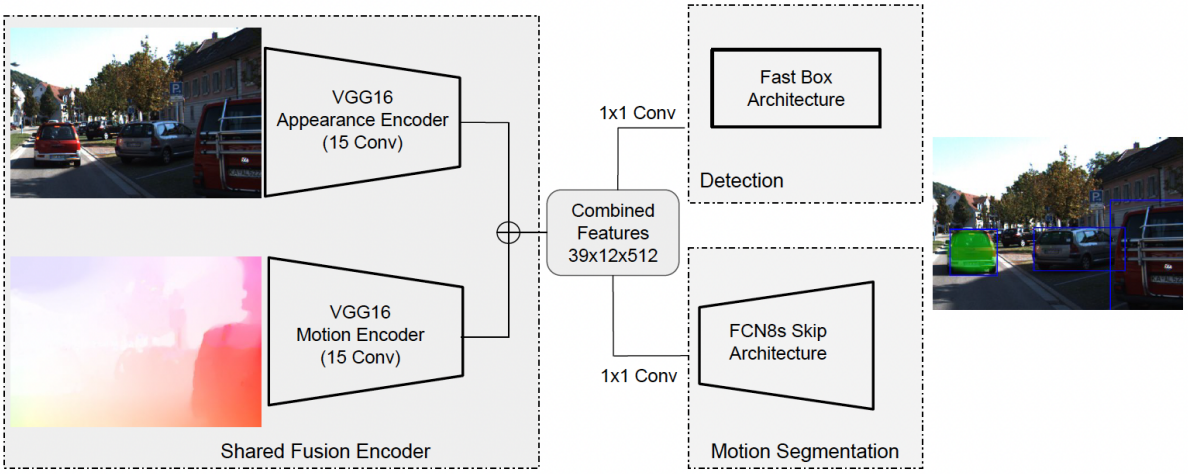


Figure 3.10: A representation of MODNet two-stream network by Siam et al. [63] with an RGB image and optical flow channel fused for object detection and motion segmentation. Figure from Siam et al. [63].

A quantitative analysis is performed on the KITTI dataset by comparing the two-stream network with an image pair and optical flow stream network shown in Table 3.4. The results show that the two-stream network utilizing RGB image and optical flow outperforms the image pair and optical flow stream network with a substantial margin in both recall and IoU. This is to be expected as the RGB image and optical flow channel combination is a better feature and motion representation to the network [63].

Table 3.4: The performance of the two-stream method using RGB image and optical flow as input compared to image pair and optical flow stream on the KITTI dataset. Table adapted from Siam et al. [63].

| Method | Precision | Recall | IoU |
|---|---|---|---|
| Stream (OF) | 70.4 | 45.7 | 50.4 |
| Stream (Image Pair) | 76.4 | 67.7 | 56.0 |
| Stream (RGB + OF) | 74.1 | **76.4** | **60.3** |

In the study performed by Rashed et al. [64], the performance of the network is analyzed by comparing the network with motion and depth information to standard RGB images. In their work, the motion information is extracted using FlowNet2 [52] and the depth information using monoDepth [65]. It was found that combining motion or depth information with RGB images provides better performance compared to standard RGB images as it can utilize information in the scene for object detection. Another interesting outcome of this study is that fusing RGB images, motion, and depth information does not yield the best results. An explanation could be that the network is learning certain information during training but does not explicitly use that information in the prediction. Another reason could be that the standard network used is not able to fuse all information. The best results are achieved by using RGB images in combination with either motion or depth information [64].

Recent work by Rashed et al. [66] expands the network with an ego-motion channel utilizing the motion of the vehicle for object detection. Instead of including depth information [64], their model predicts a segmentation mask of the objects using RGB images, optical flow, and vehicle motion tensor (VMT) as input to the network [66]. A quantitative analysis is performed using different architectures and fusion techniques (early-fusion, mid-fusion, and multi-scale feature concatenation). The results show that using RGB images, optical flow, and VMT with a mid-fusion architecture and without shared weights provide the best performance [66]. This illustrates the strong complementary relationship between optical flow and VMT fusion on the performance in contrast to optical flow and depth information fusion as explored by Rashed et al. [64].

In general solving video-related problems in real-world applications can be divided into five categories; post-processing, RCNN, optical flow, 3D CNN, and multi-stream network methods. The most interesting category in object detection in real-time, especially on embedded systems with limited hardware capability is the RCNN, optical flow, 3D CNN, and multi-stream networks since they allow an end-to-end learning process.

Besides, these methods require lower computational cost and are more supported by embedded systems in real-time such as autonomous navigation of MAVs depending on the complexity of the architecture. In this research, the focus will be on end-to-end deep learning methods for autonomous navigation of MAVs to perform gate detection in real-time using image sequences to include both spatial and temporal information to improve detection accuracy.

# 4

# Gate Detection for Micro Air Vehicles

Autonomous navigation of MAVs has gained popularity in recent years. Due to the relatively small size of MAVs, they are able to navigate through an unknown environment in a fast and agile way using onboard resources. Since the introduction of the ADR competition in 2016 at the IROS robotics conference in Daejeon, South Korea, researchers have developed and implemented various smart algorithms on MAVs in the field of high-speed ADR. One of the most important tasks in ADR is perception. Perception allows the MAVs to perceive their environment by detecting the gates in the race track. The visual information obtained by the MAV through cameras is processed onboard, allowing the MAV to navigate with high-speed through the gates in a versatile way.

In this chapter, different methods in the perception of gates that have been used in the field of high-speed ADR will be investigated. The methods can be divided into two main areas. First, in section 4.1, methods in which features are extracted from images using classical computer vision techniques for gate detection are discussed. Then, in section 4.2, the methods using deep learning for gate detection are presented.

## 4.1. Perception using Classical Computer Vision

In recent years, many advancements have been made in the field of object detection and image classification. However, researchers had to rely on classical computer vision-based methods, since deep learning applications could not be utilized on MAVs due to the limited computational power available onboard. These computer vision-based methods rely on the structure, shape, or color of a certain object by extracting characteristic features. This means that certain characteristic features have to be specified manually to detect objects during onboard image processing.

In the context of high-speed ADR, the MAVs have to navigate through a pre-defined set of gates to complete the race track successfully. Particularly in high-speed ADR, perceiving the gates in the race track correctly is an important factor to avoid direct collisions. When considering gates in ADR, they usually have a rectangular shape and have a distinct color which makes it easy to select characteristic features to detect gates. For example, the corner of the gate is an interesting characteristic feature that can be used for gate detection. Another example is to utilize the color of the gate for detection when the color of the gate is known prior.

In the IROS 2016 competition, Jung et al. [67] presented a gate detection algorithm. The algorithm is based on the color information of the gate by detecting the edges of the gate. First, the frame captured from the camera onboard the MAV is obtained and the frame is converted to a Hue-Saturation-Value (HSV) to capture the orange color of the gate. Then, a Canny edge detection method [68] is used to detect the edges of the gate in which the edges are then dilated to generate a mask of the detected gate. Using the provided information about the location of the corners of the gate, the relative pose of the gate can be derived. In their approach, the center of the gate is calculated based on the coordinates of the corners of the gates. The coordinates of the center of the gate are then used to steer the MAV towards the gate using a control algorithm.

In the work of Li et al. [3], a computationally efficient vision-based navigation and control strategy is proposed which is used in the ADR competition challenge 2017. They introduced a novel gate detection algorithm called the 'Snake Gate' detection, which is based on color detection. The principle of this algorithm is by

random sampling in the image until a certain point $\boldsymbol{P_o}$ matches with the target color of the gate. If this is the case the algorithm will continue to search up and down until $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are found such that it is below the minimum length threshold $\sigma_L$ defined as $\|\boldsymbol{P_1}$ - $\boldsymbol{P_2}\|$. In the same way, when $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are found as initial points, it will continue to search from left to right until $\boldsymbol{P_3}$ and $\boldsymbol{P_4}$ are found. A visual representation of the Snake Gate algorithm is shown in Figure 4.1.

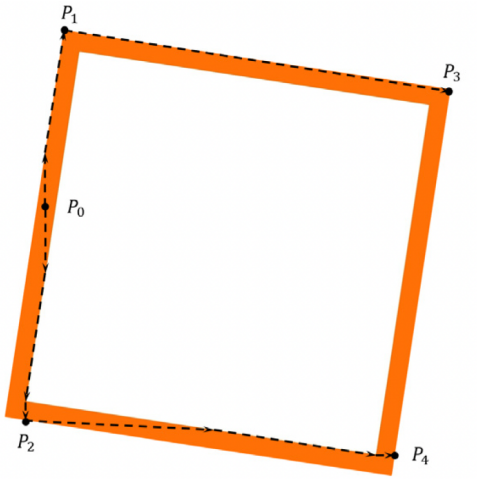


Figure 4.1: A visual representation of the Snake Gate algorithm. A random point $\boldsymbol{P_0}$ is sampled, in which the algorithm starts to search up and down until $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are found. Then, $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ are used as initial points to search left and right until $\boldsymbol{P_3}$ and $\boldsymbol{P_4}$ are found. Figure from Li et al. [3].

Once the location of the corners $\boldsymbol{P_1}$, $\boldsymbol{P_2}$, $\boldsymbol{P_3}$, and $\boldsymbol{P_4}$ of the gates are found, the pose of the gate relative to the camera can be calculated using the onboard Attitude Heading Reference System (AHRS) measurements. A limitation of the algorithm is that when the distance between the MAV and the gate is close (<1 m) the algorithm is not able to detect the location of the corners, since the gate is only partially visible. To address this problem, a second detection algorithm is introduced by Li et al. [3] called the histogram gate side detection. The detection algorithm samples the target color of the gate. Due to the distinct orange color of the gate, two peaks will appear which corresponds to the left and right sidebar of the gate frame as shown in Figure 4.2. The position of the left and right sidebar of the gate is then used to calculate the relative pose between the gate and the MAV.

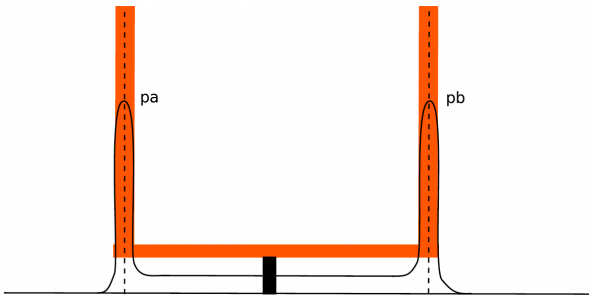


Figure 4.2: A visual representation of the histogram gate side detection algorithm. The algorithm samples the target color orange for each column. Due to this approach, two peaks will appear at $pa$ and $pb$ which indicate the location of the sidebars of the gate. The relative pose between the gate and MAV is determined based on the location of the sidebars. Figure from Li et al. [3].

Comparing the methods from Jung et al. [67] and Li et al. [3], both use gate detection methods based on the color information of the gate. However, the method by Jung et al. [67] specifically uses the Canny edge method [68], whereas Li et al. [3] determines the location of the corners of the gate to generate a mask of the gate. Besides, the method by Li et al. [3] is computationally efficient when deployed on an MAV with limited computational power, which is favorable. Both methods strongly rely on the information of the gate color, however, in real-world applications the environmental conditions will change, which poses several problems

making both methods not very robust. First, both methods are using handcrafted features in this case color detection. When navigating the MAV in a different environment with a different gate color, the features have to be manually adapted which is labor-intensive and inefficient. In the case of gates consisting of multiple colors, the algorithm will not work. Second, when using color detection, strong illumination changes in the environment can decrease the detection accuracy, resulting in poor detection performance.

## 4.2. Perception using Deep Learning

The classical computer vision-based methods have proven to show decent performance in gate detection in ADR. However, due to the way in which features have to be manually selected for processing images onboard of an MAV, this method is not robust and can have a challenging time in gate detection in a changing environment. Therefore, deep learning applications have been used in the field of high-speed ADR due to their robustness against changing environments and end-to-end learning.

Jung et al. [32] presented a visual detection method on a high-speed MAV using deep learning techniques. In their approach, they introduced a CNN using the SSD network designed by Liu et al. [29], which used the VGG network [26] as the base network. Instead of using the SSD network, they modified the architecture by removing high-level feature layers and low-level feature maps in the network. The motivation for this action is because the original network was designed to detect multiple objects, whereas the objective during ADR is to detect gates, hence unnecessary convolutional layers can be removed to increase the inference speed of the network.

As a follow up on their previous work, they investigate the learning parts of the network to increase the detection accuracy and performance which is considered an important factor in the context of high-speed ADR. Therefore, in the work of Jung et al. [4], a different base network is proposed for the CNN architecture which is AlexNet [23], achieving an inference speed of 28.95 frames per second (FPS) equipped on an NVIDIA Jetson TX2. A representation of the network architecture used is shown in Figure 4.3 which is called the ADR-Net.

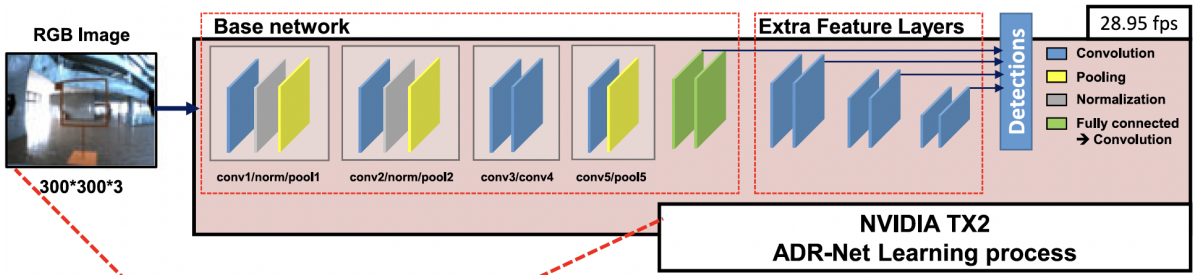


Figure 4.3: A representation of the network architecture used by Jung et al. [4] using AlexNet [23] as base network for gate detection in which high-level features and low-level feature maps are removed to increase the inference speed on the NVIDIA Jetson TX2. Figure from Jung et al. [4].

The ADR-Net was compared to the VGG-16 network and AlexNet both using the SSD as base network as shown in Figure 4.4. The results show that both the VGG-16 and AlexNet have a high detection rate and average precision (AP). However, the inference speed of both networks is relatively low. The inference speed is not sufficient to be deployed on an MAV for real-time detection. On the other hand, the ADR-Net has a relatively lower detection rate and AP but shows a higher inference speed. Although the AP and detection rate is not as high in comparison to the VGG-16 and AlexNet, this shows some promising results and the potential of using deep learning methods that can achieve good performance in gate detection in ADR.

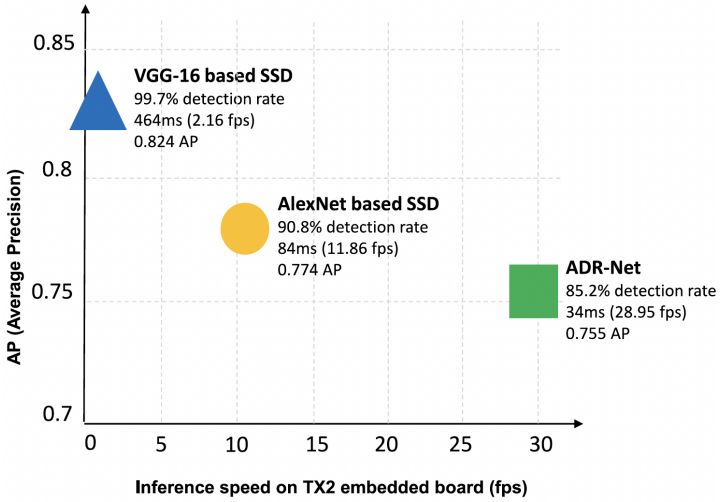


Figure 4.4: A performance comparison in the average precision and inference speed between the VGG-16, AlexNet, and ADR-Net. The results show that ADR-Net has a lower average precision compared to the VGG-16 and AlexNet but has a higher inference speed on an NVIDIA Jetson TX2. Figure from Jung et al. [4].

In the previous method by Jung et al. [4], a CNN was used to directly detect the gates. In the approach by Kaufmann et al. [69], a CNN is used to predict the pose relative to the closest gate perceived by the MAV with its uncertainties. Their CNN architecture is based on the DroNet architecture [70], which is used to navigate an autonomous drone through the streets of a city. The predictions of the network are incorporated with a Visual Inertial Odometry (VIO) within the Extended Kalman Filter (EKF) to obtain a global pose estimate of the gate locations, allowing to adapt to a dynamic environment. Instead of flying through a static race track, the gates were moved sideways when the gate was in the sight of the MAV, enabling the MAV to directly adjust its flight path to safely adapt to the situation and still navigate through the gate.

An interesting approach by detecting the corners of the observed gate is investigated by Foehn et al. [71]. In their work they use the network architecture, U-net [72] that is trained to detect multiple gates to compensate for the drift in the state estimation to build a global map with gates. The network is used to determine the location of the inner corners of the gates. In the case when only one gate is present this should not pose any problems as the corners of the gate can directly be associated with the correct gate. However, when multiple gates are present, the corners of the gates should be associated with the correct gate to accurately detect the gates. This problem is addressed by additionally training the network to extract Part Affinity Fields (PAF). The PAFs are vector fields along the edges of the corners of the gates and points in the direction of the adjacent corners associated with the correct gate. An illustration of this principle is shown in Figure 4.5.

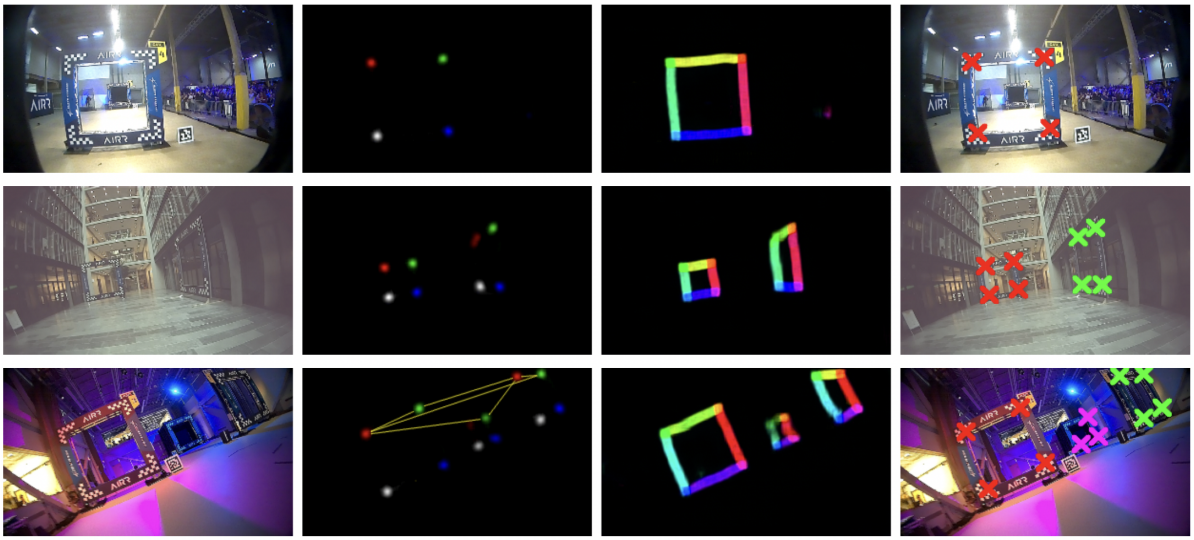

Figure 4.5: A visual representation of the detection of corners for multiple gates in a frame using the U-Net architecture [72]. The detected corners in the frame are then associated with the correct gate using Part Affinity Fields. Figure from Foehn et al. [71].

In the bottom row of Figure 4.5, three gates are present in the frame. The output of the neural network gives a set of corners indicated with red, green, blue, and white dots. Then, the PAF method is used to correctly associate the corners with the correct gate which results in a visual representation of the masked gates that are present in the frame.

When considering the different deep learning methods that have been used for gate detection in ADR, it can be seen that these types of deep learning methods predict a bounding box of the location of the gate, the relative pose of the gate with respect to the MAV, or the location of the corners of the gates. All these deep learning methods are trained using still images without temporal coherence, in which most of the images are augmented to account for motion blur and illumination changes. This is done such that the network can correctly detect the gates even in challenging conditions in which severe motion blur and illumination have a strong effect on the images that have to be processed onboard.

However, considering real-world applications such as ADR, a sequence of frames are obtained from the onboard camera of the MAV. This means that there is a strong correlation between the sequence of frames from the onboard camera of the MAV. The sequence of frames can provide useful temporal information about the motion of the object, which is not considered when training the network with still images, therefore the temporal domain is ignored which is an important factor when detecting gates from a sequence of frames obtained from the onboard camera of the MAV.

Therefore, the main limitation of the current deep learning methods for gate detection in ADR is that the temporal domain is not incorporated, hence useful information over time is absent. The main contribution of this research is to gain a better understanding of the effect of incorporating temporal information on the detection accuracy in deep-learning-based object detection and apply it to a better gate detection approach in ADR.

# 5

# Literature Synthesis

This chapter concludes the literature study phase of the research. The topic of this research entails the deep learning applications using image sequences for object detection in high-speed Autonomous Drone Racing (ADR). The literature has been divided into two main areas that have been extensively reviewed. First, the existing deep learning applications in object detection in images and videos are investigated. Second, the existing deep learning applications in the field of ADR are reviewed. The main findings of both areas from the literature study will be summarized in section 5.1 and 5.2.

## 5.1. Object Detection

Different methods have been used in the field of object detection to identify the object of interest in a variety of applications. In the early days, classical computer vision methods were used based on feature extraction on images. Examples of classical computer vision methods include Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), corner detection, and Histogram of Oriented Gradients (HOG). However, these methods are either computationally expensive to be run on embedded systems or not robust against environmental or illumination changes due to their feature extraction properties.

Since the introduction of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010, many different methods were proposed to solve object detection-related problems, in which the most notable method uses deep learning in the field of object detection. The most common deep learning method consists of a Convolutional Neural Network (CNN) with a reasonable architecture complexity that is able to achieve good performance in detection accuracy in object detection such as the SSD network [29] and the VGG network [26]. The two main deep learning frameworks in object detection can be divided into two areas; one-stage object detectors and two-stage object detectors. The main difference is that the latter uses an intermediate Region Proposal Network (RPN) to determine a set of region object proposals before showing the final detection results such as the Faster R-CNN [35]. One-stage object detectors prioritize inference speed whereas two-stage object detectors emphasize detection accuracy. For example, the Fast YOLO network [28] is specifically designed to achieve high inference speed due to its lightweight network architecture. Both object detectors show decent detection performance when considering single images. However, when applied to videos that contain sequences of frames, the algorithm tends to fail since it is unable to deal with situations containing motion blur, occlusion, or pose variations of objects [38]. The main reason for this limitation is the way the network is trained. Most deep learning networks are trained using a clean dataset without the appearance of previously mentioned properties. A partial solution to this problem is to include data augmentation during training.

Because videos contain sequences of frames it becomes a natural choice to include temporal information during the learning process of the network. To include temporal information in the network, four main areas were considered. First, Recurrent Convolutional Neural Network (RCNN) methods were considered. These types of methods consist of a combination of CNNs and Recurrent Neural Networks (RNNs) [40, 44] to capture temporal information. Second, optical flow methods were considered to include temporal information as it provides information about the motion of objects between frames. The Flow-Guided-Feature-Aggregation (FGFA) method by Zhu et al. [49] considers adjacent frames by propagating the feature maps from the adjacent frames to the current frame using flow fields. The feature maps are then aggregated such that the

generated feature maps contain a higher activation for detection. In this way, frames with weak activation due to motion blur, occlusion, and pose variations can be enhanced by using adjacent frames that contain higher activation. Additionally, recent optical flow methods have shown good performance while achieving promising inference speed on embedded systems [51]. The third method that has been used recently is a 3D CNN [58, 62], which is an extension of the 2D CNN in the temporal domain. This is a relatively simple method in which a sequence of frames can directly be processed by the 3D CNN. However, applications of 3D CNNs in object detection are rather limited. The fourth method consists of a multi-stream network that fuses a combination of RGB images, optical flow, depth information, and ego-motion to provide the network with salient information to improve detection accuracy. The fusion of RGB images, optical flow, and ego-motion showed the most promising results [66].

The advancement in Artificial Intelligence (AI) has led to sophisticated deep learning methods in the field of object detection achieving state-of-the-art performance. From the literature, the most promising methods found are RCNN, optical flow, 3D CNNs, and multi-stream networks that can directly capture temporal information from sequences of frames.

## 5.2. Gate Detection for Micro Air Vehicles

In the context of high-speed ADR, Micro Air Vehicles (MAVs) have to navigate through a pre-defined set of gates to complete the race track successfully. Jung et al. [32] introduced a gate detection algorithm based on the color information of the gate. Li et al. [3] developed an efficient vision-based algorithm and control strategy called the Snake Gate algorithm. Both methods are based on color segmentation, however, the main limitations are that the features have to be selected and that the methods are not robust against illumination changes which can impact the detection accuracy. Deep learning applications have been used in the field of ADR as it has shown promising results.

Jung et al. [4] presented a visual detection method based on deep learning. They introduced a CNN using the SSD network by Liu et al. [29] and modified the architecture by removing high and low-level features named ADR-Net. The results showed that the ADR-Net achieved decent performance while achieving a high inference speed that is compatible with real-time detection on MAVs. Foehn et al. [71] used a CNN based on the U-Net [72] architecture to predict the location of the corners of the gates, then Part Affinity Fields (PAF) were applied to associate the predicted corners to the correct gate.

Both methods show the potential of achieving good performance using deep learning in the field of ADR. However, the methods described are based on single images which mean that only spatial dependencies are considered and that the temporal domain is not exploited. Frequently, in high-speed ADRs, the detection accuracy deteriorates due to the fast motion of the MAVs resulting in motion blur, occlusion, and pose variations of objects. These properties are the main culprit of poor detection accuracy when considering high-speed autonomous navigation of MAVs.

# 6

# Final Remarks

In this literature review, the most recent knowledge and applications of object detection in the field of autonomous navigation of Micro Air Vehicles (MAVs) are explored, which serve as a fundamental basis for the research on the applications of deep learning using image sequences for MAVs. This is achieved by performing an extensive literature study in two main areas; object detection and gate detection for MAVs. The literature regarding object detection reviews the different methods used for object detection for different applications in general. The literature regarding gate detection for MAVs considers the different approaches that have been taken in gate detection in the context of Autonomous Drone Racing (ADR).

In the literature about object detection it was found that it can be divided into two main areas; object detection in images and object detection in videos. In both areas, deep learning methods are used for object detection as it has proven their potential by achieving good detection performance. The fundamental difference between both areas is the network learning approach taken. In the case of object detection in images, it is natural to train the deep learning algorithms using uncorrelated images as the main goal is to detect the objects in a specific image. On the contrary, object detection in videos contains sequences of images that are correlated. Hence object detection in this area includes the temporal domain, in which sequences of frames should be utilized in the deep learning algorithms. To support the importance of the temporal domain in object detection, it was found that the performance of deep learning object detection algorithms trained on still images achieve poor performance on the application of videos due to motion blur, occlusion, and pose variations of objects. This proves that the temporal domain is of utmost importance for object detection in real-world applications involving videos such as autonomous navigation of MAVs.

Different deep learning algorithms are explored that provide reasonable performance using image sequences. The literature provided methods that utilize image sequences which can be divided into four main areas; Recurrent Convolutional Neural Networks (RCNN), optical flow, 3D convolutional neural networks (CNN), and multi-stream networks. Recurrent Convolutional Neural Networks (RCNN) are often used as they are able to handle long-term sequential modeling which can hold information over time using memory cells such as a Long Short-Term Memory (LSTM). However, the exact behavior and intuition of the LSTM are not well understood. Optical flow captures the relative motion of objects between two consecutive frames and can be utilized to include temporal information over a sequence of frames, providing the network with more information to increase detection performance. Optical flow methods are often used to directly capture the motion of objects but also to warp feature maps such that feature maps with weak activation can be increased that are subjected to motion blur, occlusion, or pose variations of objects. However, existing methods using optical flow only utilize scene flow and do not take into account ego-motion or include depth information. 3D CNN is an extension of a 2D CNN that is able to naturally include the temporal domain by providing the network with a sequence of frames without relying on external methods. Multi-stream methods utilize different channels that contain a combination of RGB images, optical flow, depth information, and ego-motion providing the network with more relevant information to increase performance. In this literature, it was found that the RCNN, optical flow, 3D CNN, and multi-stream network methods provide important temporal information that can be utilized and are considered most suitable for deep learning using image sequences. In addition, it was found that generating a segmentation mask representing the object of interest is more accurate than bounding box proposals.

The detection accuracy and inference speed of the discussed deep learning algorithms depend on the application. The deep learning algorithms discussed were mostly designed to classify and detect 1000 different classes on publicly available datasets, hence the complexity of the network is higher which results in higher detection accuracy, however, the inference speed is compromised. In this research, the application will be in the field of autonomous navigation of MAVs, in particular, gate detection in the context of ADR. Therefore, the different deep learning approaches such as RCNN, optical flow, 3D CNN, and multi-stream networks will be used as the basis for developing a deep learning algorithm with a lightweight network that is able to achieve good detection accuracy and inference speed.

In the literature about gate detection in ADR, it was found that most recent methods use deep learning approaches. Most interestingly, these methods show decent performance during high-speed ADR. These deep learning methods are trained on still images that are heavily augmented to account for conditions occurring during high-speed ADR such as motion blur, occlusion, and pose variations of objects. However, the deep learning methods are applied to still images rather than sequences of images. This means that the temporal domain is not exploited and therefore leaving room for improvement in this field of research.

In conclusion, this literature review has shown the different deep learning approaches taken in the field of object detection. More research has been done recently in object detection in videos which presents a new challenge to find ways to include temporal information in the detection of objects in videos. The contribution of this research is to gain a better understanding of the effect of incorporating temporal information on the detection accuracy in deep-learning-based object detection and apply it to a better gate detection approach in ADR.

# III

# Conclusions

# 7

# Conclusions

The literature study presented in Part II serves as the foundation of the research in which the results are presented in Part I in the form of a scientific article. In this research, based on the quantitative comparison of the different methods using information over time for semantic segmentation for Micro Air Vehicles (MAVs), it was found that the Recurrent Neural Network (RNN) is the best performing network in terms of accuracy and inference speed. The RNN shows a good trade-off between accuracy and inference speed which can be used in real-time for semantic segmentation applications using MAVs. The main conclusions in the article are given as answers to the research questions that were initially proposed in chapter 1.

**Q1)** What are the fundamental differences between training a deep learning algorithm with a single input image compared to image sequences?

The main difference between training a deep learning algorithm with a single input image for example U-Net, is that the images are processed independently without any correlation over time. One of the main limitations of this approach is that information over time is not utilized in the segmentation predictions of the network. Applications in the real world often require a stream of consecutive images that are in sequences. These images are correlated over time and therefore leveraging this information in the prediction is one of the main factors to improve the performance of semantic segmentation in videos. Furthermore, deep learning algorithms trained with single input images often suffer from performance degradation when applied to videos due to motion blur from fast camera motion, incomplete segmentation masks for distant objects, and objects that are occluded.

**Q2)** Which deep learning algorithms are best suited for detection using image sequences?

The first method that utilizes information over time is a recurrent network. These types of networks are using memory over time to retain relevant past information in which the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are frequently used. A common approach with these recurrent networks is to replace the operators of the LSTM and GRU with convolutional operators as it is more suitable for image processing. The second method uses 3D Convolutional Neural Networks (CNNs) as it allows to perform convolution both in the spatial and temporal direction. The third method leverages optical flow as it provides useful information about the motion of the object.

**Q3)** What deep learning algorithms perform the best in terms of detection accuracy and inference speed?

Based on the quantitative results from this research, it was found that the RNN in which the operators are replaced by convolutional operators provides the best trade-off in terms of detection accuracy and inference speed. The RNN shows good performance on the KITTI (MOTS) and CyberZoo dataset while achieving competitive inference speed with respect to the single-frame-based semantic segmentation method.

**Q4)** How does the algorithm perform in real-time when deployed on a high-speed autonomous MAV?

The RNN performs the best in terms of detection accuracy and inference speed. Since the network is able to achieve high inference speed on a Desktop GPU, the performance of the network in real-time is measured by running the network on an NVIDIA Jetson TX2. The results showed that the network is able to run 30 FPS in real-time.

Overall, this research has shown that with a simple solution by using an RNN, that the performance of semantic segmentation in videos can be improved. Furthermore, it has been shown that the RNN is able to provide good performance while at the same time achieving high inference speed, which makes it compatible to be deployed on MAVs with limited processing power for computer vision tasks such as semantic segmentation.

# A

# Additional Figures

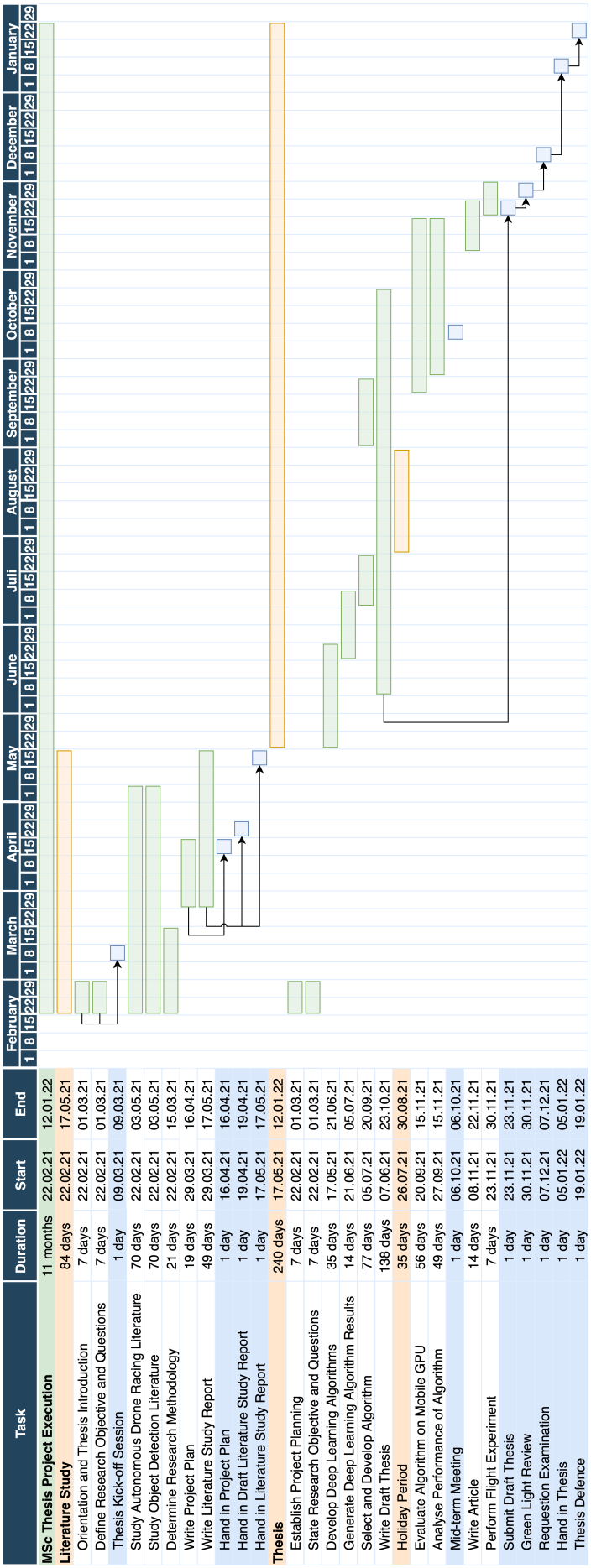


Figure A.1: An overview of the MSc thesis project planning

# Bibliography


[1] Hyungpil Moon, Yu Sun, Jacky Baltes, and Si Jung Kim. “The IROS 2016 Competitions [Competitions]”. In: *IEEE Robotics Automation Magazine*. Vol. 24. 1. 2017, pp. 20–29.

[2] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. “Metric monocular SLAM and colour segmentation for multiple obstacle avoidance in autonomous flight”. In: *Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*. 2017, pp. 234–239.

[3] Shuo Li, Michaël M.O.I. Ozo, Christophe De Wagter, and Guido C.H.E. de Croon. “Autonomous drone race: A computationally efficient vision-based navigation and control strategy”. In: *Robotics and Autonomous Systems*. Vol. 133. 2020, p. 103621.

[4] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. “Perception, Guidance, and Navigation for Indoor Autonomous Drone Racing Using Deep Learning”. In: *IEEE Robotics and Automation Letters*. Vol. 3. 3. 2018, pp. 2539–2544.

[5] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. “Object Detection With Deep Learning: A Review”. In: *IEEE Transactions on Neural Networks and Learning Systems*. Vol. 30. 11. 2019, pp. 3212–3232.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[7] Leon Bottou. “Large-Scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of COMPSTAT’2010*. Physica-Verlag HD, 2010, pp. 177–186.

[8] Boris Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *USSR Computational Mathematics and Mathematical Physics*. Vol. 4. 5. 1964, pp. 1–17.

[9] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.” In: *The Journal of Machine Learning Research*. Vol. 12. Jan. 2010, pp. 257–269.

[10] Tijmen Tieleman, Geoffrey Hinton, et al. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning*. Vol. 4. 2. 2012, pp. 26–31.

[11] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations (ICLR)*. 2015.

[12] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems*. Vol. 2. Morgan-Kaufmann, 1990.

[13] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE*. Vol. 86. 11. 1998, pp. 2278–2324.

[14] Felix Gers and Juergen Schmidhuber. “Recurrent nets that time and count”. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 3. 2000, pp. 189–194.

[15] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems (ICONIP)*. Vol. 1. MIT Press, 2015, pp. 802–810.

[16] Paul Viola and Michael Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2001, pp. I–I.

[17] David Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 1999, pp. 1150–1157.

[18] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. 2006, pp. 404–417.

[19] Darshana Mistry and Asim Banerjee. “Comparison of Feature Detection and Matching Approaches: SIFT and SURF”. In: *GRD Journals- Global Research and Development Journal for Engineering*. Vol. 2. 2017, pp. 7–13.

[20] Chris Harris and Mike Stephens. “A Combined Corner and Edge Detector”. In: *Proceedings of the Alvey Vision Conference*. 1988, pp. 23.1–23.6.

[21] Jianbo Shi and Carlo Tomasi. “Good features to track”. In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1994, pp. 593–600.

[22] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 2005, pp. 886–893.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 25. 2012.

[24] Matthew Zeiler and Rob Fergus. “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision – ECCV*. 2014, pp. 818–833.

[25] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann Lecun. “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks”. In: *International Conference on Learning Representations (ICLR)*. 2013.

[26] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations (ICLR)*. 2015.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[28] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525.

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander Berg. “SSD: Single Shot MultiBox Detector”. In: *Computer Vision – ECCV*. 2016, pp. 21–37.

[30] Alexander Neubeck and Luc Van Gool. “Efficient Non-Maximum Suppression”. In: *18th International Conference on Pattern Recognition (ICPR)*. Vol. 3. 2006, pp. 850–855.

[31] Donghwi Kim, Hyunjee Ryu, Jedsadakorn Yonchorhor, and David Hyunchul Shim. “A Deep-learning-aided Automatic Vision-based Control Approach for Autonomous Drone Racing in Game of Drones Competition”. In: *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*. Vol. 123. 2020, pp. 37–46.

[32] Sunggoo Jung, Hanseob Lee, Sunyou Hwang, and David Hyunchul Shim. “Real Time Embedded System Framework for Autonomous Drone Racing using Deep Learning Techniques”. In: *AIAA Information Systems-AIAA Infotech @ Aerospace*. 2018.

[33] J. Arturo Cocoma-Ortega, L. Oyuki Rojas-Perez, Aldrich A. Cabrera-Ponce, and J. Martinez-Carranza. “Overcoming the Blind Spot in CNN-based Gate Detection for Autonomous Drone Racing”. In: *Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*. 2019, pp. 253–259.

[34] Ross Girshick. “Fast R-CNN”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. Vol. 39. 2015.

[36] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. “Mask R-CNN”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[37] Florian Ölsner and Stefan Milz. “Catch Me, If You Can! A Mediated Perception Approach Towards Fully Autonomous Drone Racing”. In: *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*. Vol. 123. 2020, pp. 90–99.

[38] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, and Wanli Ouyang. "T-CNN: Tubelets With Convolutional Neural Networks for Object Detection From Videos". In: *IEEE Transactions on Circuits and Systems for Video Technology*. Vol. 28. 10. 2018, pp. 2896–2907.

[39] Wei Han, Pooya Khorrami, Tom Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Humphrey Shi, Jianan Li, and Shuicheng Yan. "Seq-NMS for Video Object Detection". In: (Feb. 2016).

[40] Guanghan Ning, Zhi Zhang, Chen Huang, Xiaobo Ren, Haohong Wang, Canhui Cai, and Zhihai He. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 2017, pp. 1–4.

[41] Sepehr Valipour, Mennatullah Siam, Martin Jagersand, and Nilanjan Ray. "Recurrent Fully Convolutional Networks for Video Segmentation". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 29–36.

[42] Bowen Wang, Liangzhi Li, Yuta Nakashima, Ryo Nawasaki, Hajime Nagahara, and Yasushi Yagi. "Noisy-LSTM: Improving Temporal Awareness for Video Semantic Segmentation". In: *IEEE Access*. Vol. 9. 2021, pp. 46810–46820.

[43] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. "Online Video Object Detection Using Association LSTM". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2363–2371.

[44] Menglong Zhu and Mason Liu. "Mobile Video Object Detection with Temporally-Aware Feature Maps". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 5686–5695.

[45] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 39. 4. 2017, pp. 677–691.

[46] Hugh Christopher Longuet-Higgins and Kvetoslav Prazdny. "The interpretation of a moving retinal image". In: *Proceedings of the Royal Society of London. Series B. Biological Sciences*. Vol. 208. 1980, pp. 385–397.

[47] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. "Deep Feature Flow for Video Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4141–4150.

[48] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766.

[49] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. "Flow-Guided Feature Aggregation for Video Object Detection". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 408–417.

[50] Shiyao Wang, Yucong Zhou, Junjie Yan, and Zhidong Deng. "Fully Motion-Aware Network for Video Object Detection". In: *Computer Vision – ECCV*. 2018, pp. 557–573.

[51] Lingtong Kong, Chunhua Shen, and Jie Yang. "FastFlowNet: A Lightweight Network for Fast Optical Flow Estimation". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 10310–10316.

[52] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1647–1655.

[53] Anurag Ranjan and Michael Black. "Optical Flow Estimation Using a Spatial Pyramid Network". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2720–2729.

[54] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8934–8943.

[55] Hui Tak-Wai, Tang Xiaoou, and Loy Chen Change. "LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8981–8989.

[56] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. “Learning Spatiotemporal Features with 3D Convolutional Networks”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4489–4497.

[57] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. “3D Convolutional Neural Networks for Human Action Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. Vol. 35. 1. 2013, pp. 221–231.

[58] Shizhou Dong, Zhifan Gao, Sandeep Pirbhulal, Gui-Bin Bian, Heye Zhang, Wanqing Wu, and Shuo Li. “IoT-based 3D convolution for video salient object detection”. In: *Neural Computing and Applications*. Vol. 32. 2020.

[59] Na Lu, Yidan Wu, Li Feng, and Jinbo Song. “Deep Learning for Fall Detection: Three-Dimensional CNN Combined With LSTM on Video Kinematic Data”. In: *IEEE Journal of Biomedical and Health Informatics*. Vol. 23. 1. 2019, pp. 314–323.

[60] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Rich Zemel, and Yoshua Bengio. “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. 2015, pp. 2048–2057.

[61] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. “Action Recognition using Visual Attention”. In: *International Conference on Learning Representations (ICLR) Workshop*. 2016.

[62] Thangarajah Akilan, Qingming Jonathan Wu, Amin Safaei, Jie Huo, and Yimin Yang. “A 3D CNN-LSTM-Based Image-to-Image Foreground Segmentation”. In: *IEEE Transactions on Intelligent Transportation Systems*. Vol. 21. 3. 2020, pp. 959–971.

[63] Mennatullah Siam, Heba Mahgoub, Mohamed Zahran, Senthil Yogamani, Martin Jagersand, and Ahmad El-Sallab. “MODNet: Motion and Appearance based Moving Object Detection Network for Autonomous Driving”. In: *21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2859–2864.

[64] Hazem Rashed, Ahmad El Sallab, Senthil Yogamani, and Mohamed ElHelw. “Motion and Depth Augmented Semantic Segmentation for Autonomous Navigation”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 364–370.

[65] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. “Unsupervised Monocular Depth Estimation with Left-Right Consistency”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6602–6611.

[66] Hazem Rashed, Ahmad El Sallab, and Senthil Yogamani. “VM-MODNet: Vehicle Motion aware Moving Object Detection for Autonomous Driving”. In: *IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 1962–1967.

[67] 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”. In: *Journal of Field Robotics*. Vol. 35. 2017.

[68] John Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 8. 6. 1986, pp. 679–698.

[69] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. “Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing”. In: *International Conference on Robotics and Automation (ICRA)*. 2019, pp. 690–696.

[70] Antonio Loquercio, Ana I. Maqueda, Carlos R. del-Blanco, and Davide Scaramuzza. “DroNet: Learning to Fly by Driving”. In: *IEEE Robotics and Automation Letters*. Vol. 3. 2. 2018, pp. 1088–1095.

[71] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. “AlphaPilot: Autonomous Drone Racing”. In: *RSS: Robotics, Science, and Systems*. 2020.

[72] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, pp. 234–241.