TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Verkehrswissenschaften „Friedrich List “, Institut für Verkehrstelematik

Diplomarbeit

Semantische Segmentierung optischer Sensordaten für Anwendungen in der Binnenschifffahrt

eingereicht von

Lukas Hösch

geb. am 19.03.1996 in Stuttgart

Hochschullehrer: **Prof. Dr.-Ing. O. Michler**

Betreuende: **Dipl.-Ing. Albrecht Michler,**
M.-Eng. Xinyu Zhang,
Dr.-Ing D. Medina

Dresden, den 26. August 2022 _____

Lukas Hösch

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Verkehrswissenschaften „Friedrich List" - Professur ITVS

Aufgabenstellung zur Diplomarbeit

im Studiengang Verkehrsingenieurwesen

Für Herrn cand. Ing. Lukas Hösch

Thema: **Semantische Segmentierung optischer Sensordaten für Anwendungen in der Binnenschifffahrt**

Thema (engl.): Semantic Segmentation for Inland Waterway Vessels

1. Prüfer/in: Prof. Dr.-Ing. Oliver Michler

2. Prüfer/in: Dipl.-Ing. Albrecht Michler

Hochschulbetreuer/in: Prof. Dr.-Ing. Oliver Michler

Praxisbetreuer/in: Daniel A. Medina M. Sc. (DLR Neustrelitz)

Zukünftige autonome Fahrzeuge, Roboterplattformen und andere sicherheitskritische Anwendungen teilen hohe Anforderungen an Umgebungserfassung, präzise Navigationsinformationen und Verkehrslagedarstellungen. Im Hinblick darauf ist eine vollständige Umgebungswahrnehmung eine entscheidende Voraussetzung für die Planung von Aktionen und den sicheren Betrieb in realen Umgebungen. Diese Arbeit befasst sich mit dem Wahrnehmungssystem von Binnenschiffen, wobei als Grundlage Daten mehrerer Sensoren integriert verwendet werden sollen. Insbesondere sollen die von LiDARs und Kameras erfassten Punktwolken und Bilder fusioniert werden, um eine semantische Segmentierung zu ermöglichen. Semantische Segmentierung bezeichnet hierbei die Fähigkeit, Objekte und Strukturen in der Umgebung eines halbautonomen Binnenschiffs während seines Betriebs zu erkennen, klassifizieren und verstehen. Die Hauptaufgabe dieser Arbeit besteht in der Implementierung von Algorithmen zur Wassersegmentierung und zum optischen Fluss auf der Grundlage von LiDAR- und optischen Sensoren, wobei der Schwerpunkt auf deren Umsetzung für Binnenschifffahrtsszenarien liegt.

Die Aufgaben im Einzelnen:

- Einarbeitung in die Grundlagen des Robotic Operating System (ROS) zur Erfassung und Verarbeitung von Daten der relevanten Sensorsysteme
- Recherche zum Forschungsstand und Einarbeitung in aktuellen Deep-Learning-Lösungen zur semantischen Segmentierung
- Implementierung eines Algorithmus zur Wassersegmentierung auf Basis von LiDAR- und optischen Sensoren
- Entwicklung eines Flussschätzers unter Verwendung der oben genannten Sensoren
- Durchführung von Messkampagnen
- Validierung der Ergebnisse und Bewertung aus verkehrstelematischer Anwendungsperspektive

Oliver Michler — Digital unterschrieben von Oliver Michler, DN: c=DE, st=Sachsen, l=Dresden, o=Technische Universitaet Dresden, cn=Oliver Michler, Datum: 2022.02.01 23:52:33 +01'00'

Albrecht Michler — Digital unterschrieben von Albrecht Michler, Datum: 2022.02.01 16:50:12 +01'00'

Datum Unterschrift Prüfer/in

Bei der Bearbeitung ist die „Richtlinie für die Anfertigung der Diplomarbeit" der Fakultät Verkehrswissenschaften „Friedrich List" zu beachten.

Eintragungen des Prüfungsamts

Arbeitsbeginn am: 09.02.2022 erhalten: Lukas Hörl

Einzureichen am: 09.07.2022

Unterschrift Studierende/r

Bibliographic evidence

Lukas Hösch

Diplomarbeit
Semantische Segmentierung optischer Sensordaten für Anwendungen in der Binnenschifffahrt

Technische Universität Dresden
Fakultät Verkehrswissenschaften “Friedrich List”
Institut für Verkehrstelematik

Studiengang Verkehrsingenieurwesen

94 Seiten, 37 Abbildungen, 5 Tabellen, 102 Quellenangaben

Abstract:
Inland waterway transport (IWT) is an extremely important backbone for heavy good transportation with severe economical influence and the potential for the reduction of traffic-related greenhouse gas emission. As IWT is expected to increase, updated chart data is required. Traditional survey methods are intense in cost and time. This work presents a processing scope for self-updating inland waterway charts. The required data can be gathered through optical sensors, that are fitted on IWT vessels.
In semantic segmentation, every pixel in a RGB image is assigned to a defined class. This machine-learning problem is used to distinguish between various objects in a

(IWT related) scene and thus to survey the infrastructure. For this task, the new BerlinIWT dataset is proposed. Existing datasets in this field may contain more examples, but do not provide an adequate number of classes. Training a neural network on the datasets *MaSTr1325* and BerlinIWT leads to remarkable results.
Spatial mapping information is completed with LiDAR (light detection and ranging) data. The acquired 3D point clouds provide precise distance information with a reasonable maximum range. The sensor compensates the flaws of (stereo) cameras, that are suitable for scene understanding, but inappropriate for distance measurements. The most suitable technique for the combination of LiDAR and camera data is discussed. For the ongoing scope towards simultaneous localisation and mapping (SLAM), two different methods for optical flow estimation are compared.
Finally, further processing steps are pointed out and the application is discussed with respect to a traffic-telematics related use-case.

Note of thanks

Daniel, thank you for addressing new topics with such matter of course, courage and faith. We have worked our way through and I would like to keep going. Thank you, Ralf, for always having my back and assisting in challenges that seemed impossible.
Thank you, Albrecht and Xinyu, for your valuable remarks and your time for the meetings with me. I have had a great supervision from both of you.
For the measurement campaign, many thanks to Astrid, Tom, Pawel, Uwe and Xiangdong. This part of the work wouldn't have been possible without the support of you all. Thank you, Dominic, Nele, Philipp and Raik for withstanding two weeks with me in the same office and of course for your valuable work. Special thanks also to Lars for your support in the dependency hell and Jetson heaven. I don't know where I would have been without you. Alonso, thank you for your valuable work and the great atmosphere. Thank you Filippo for your support on site and your socialising initiatives. Thanks to Thoralf for your support as a department lead and the opportunities you provided. I'm glad working in this department.
Many thanks to Jorge who provided all his knowledge and hints about machine learning! Also thanks to Heinrich, who never got tired analysing and discussing the challenges. Finally I would like also to thank Niklas and all of my friends for dragging me away from the laptop when ever necessary and possible.

Contents

Propositions for the thesis

1. Commercial shipping on inland waterways requires well-updated and precise chart data. Traditional survey means are time-consuming and expensive.
2. A significant number of inland waterway vessels equipped with appropriate sensors is able to provide spatial mapping information faster and at higher efficiency.
3. Semantic segmentation, performed by artificial neural networks, is essential to identify the surrounding and can be carried out best on RGB images.
4. LiDAR sensors are a decent possibility to provide precise spatial mapping information, that is required for generating a self-updating chart.

List of Figures

List of Tables

List of Symbols

α	learning rate
β	moving average parameter RMS Prop
β_1	hyperparameter for Adam
β_2	hyperparameter for Adam
β_l	batch norm parameter to be learned
β_r	reflectance of the target's surface
ϵ	parameter preventing zero-division
η	overall system efficiency in a laser-rangefinder
γ	batch norm parameter to be learned
λ	wavelength
λ_l	parameter, that controls information flow from 2D pixel to 3D point
μ	mean value
σ^2	covariance value
$a^{[l]}$	generic activation function in layer l
a_f	type of activation function
A_r	area of receive aperture (Blende) at range r
b	bias
$\mathbf{B}^{[l]}$	biases in layer l
B_r	modulation bandwidth
c	number of channels
c_r	speeed of light
d_c, ϕ_c, θ_c	spherical coordinates in LiDAR point cloud
d_r	decay rate

$\mathbf{d}$	displacement
e_p	current epoch
E_p	total energy of a transmitted pulse laser
f	filter dimension (height and width)
$\mathbf{f}$	filter (kernel)
f_{if}	frequency shift
f_d	Doppler frequency shift
f_m	signal of video frame
g	activation function
$\mathbf{G_l}$	graph used by LDLS for 2D / 3D mapping
h	number of hidden units
H	height
I_i	frame i of a consecutive video
$\mathbf{I}$	identity matrix
$J(w)$	generic cost function
K	number of layers
l	layer
L	generic loss function
m	number of training examples
m_l	object instance
M_l	number of object instances
n	image dimension (height and width)
$n_r = 1$	refraction index
o	activation threshold
p	amount of rows and columns that are added by padding
p_f	center pixel for optical flow estimation
p_m	polynomial of video frame
p_W	value of parameters inside vector $\mathbf{W}^{[l]}$
P_r	received power
$P(\mathbf{x}_{c,i})$	set of image pixels
$\mathbf{p}_{l,j}$	coordinate vector of 2D pixel i
q_i	point of tracked feature

r	distance to target
s	stride
t	time
t_w	waveform period
T_r	transmission loss trough the transmission medium
u	unit (neuron)
v	velocity
w	weight
W	width
$\mathbf{W}^{[l]}$	weights in layer l
x	input
$\mathbf{x}$	input matrix (e.g. image)
$\mathbf{X}$	input vector
x_f, y_f	pixel coordinates of I_i
$\mathbf{X}^{\{t\}}$	Mini-batch containing training examples
x_c, y_c, z_c	Cartesian coordinates in LiDAR point cloud
$\mathbf{x}_{c,i}$	coordinate vector of 3D LiDAR point i
y	desired output
$\hat{y}$	predicted output
$\mathbf{y}$	desired output matrix (e.g. true mask)
$\hat{\mathbf{y}}$	predicted output matrix (e.g. predicted mask)
y_l	pixel of the mask
$\mathbf{Y}$	vector containing desired outputs
$\mathbf{Y}^{\{t\}}$	Mini-batch containing desired output
$\mathbf{z_l}^{(m)}$	vector containing labels for object instances

List of Abbreviations

ANN Artificial Neural Networks

ASPP Atrous Spatial Pyramid Pooling

Adam Adaptive Moment Estimation

BEV Bird's Eye View

CNN Convolutional Neural Networks

CVAT Computer Vision Annotation Tool

DLR *Deutsches Zentrum für Luft- und Raumfahrt* / German Aerospace Center

FC fully connected

FMCW Frequency-Modulated Continuous Wave

FoV Field of View

GNSS Global Navigation Satellite System

GPU Graphical Processing Unit

HSV Hue, Saturation, Value

IMU Inertial Measurement Unit

IWT Inland Waterway Transport

IoU Intersection over Union

LiDAR light detection and ranging

MEMS Microelectromechanical systems microscanning

ML Machine Learning

NIR near-infrared

NN Neural Network

PNT Position, Navigation and Timing

PPP Precise Point Positioning

RGB red, green, blue

RINEX Receiver Independent Exchange

RMS Prop Root-Mean-Square Propagation

ROS Robotic Operating System

RTK Real-time Kinematics

ReLu Rectified Linear unit

ResNet Residual Network

SLAM Simultaneous Localization and Mapping

SNR Signal to Noise Ratio

SOW Spree-Oder-Wasserstraße

ToF Time of Flight

USV Unmanned Surface Vehicles

VHF Very High Frequency

1. Introduction

Inland Waterway Transport (IWT) is a cost-efficient, safe and environmentally friendly mode of transport with low energy consumptions [1]. On the inland waterways of Germany, the total transport volume of goods amounted to 222.7 million tons in 2017. This value is expected to grow by approximately 22 % until 2030 / 2035 [1], which is a challenging development for the current state of the associated infrastructures. IWT is capable of an considerably valuable contribution to the reduction of external effects of transport systems [2]. The average external costs concerning the emission of greenhouse gases from IWT are about 40 % lower than the corresponding value of road traffic with heavy good vehicles. It is important to mention, that the lack of data might slightly compromise the comparability. Still, also the multi-functionality of inland waterways reduces the external cost caused by habitat damage.
Generally, accidents on inland waterways are comparably rare. However, collisions with bridges or other infrastructure might occur [3], which would impede the general traffic flow on the associated waterway. The effects of a waterway being blocked by vessels suffering from an accident were illustrated by the container ship *Ever Given*, that caused severe traffic obstruction in the *Suez canal*. The seven days blocking in march 2021 of this important merchant route caused painful economical effects [4]. Even though such effects may not arise with comparable extend in case of a similar scenario on inland waterways, an impact might be noticeable. The existing risk of collision is expected to increase with the growing traffic load.

1.1 Overall Idea of the Project

IWT makes an important contribution to the environmental goals as well as to the economic development. In order to minimise the risk of collision, precise and well updated chart data is a crucial parameter. Traditional means and techniques for inland waterway surveying are expensive and time-consuming.
Available (low-cost) optical sensors may be used for general surveying along the waterway, but also to collect measurements concerning the relevant infrastructure such as bridges and waterway locks. The used sensors might indeed suffer from a lack of accuracy, compared to high performance survey devices used by particular measurement teams. With a relevant number of inland waterway vessels equipped with the associated sensors, the precise measurement values can be estimated numerically from various datasets of the same waterway section. Obtaining measurements from a certain amount of vessels on their daily journeys along the inland waterway network ensures the currency of the gathered information. Furthermore, without the need to carry out specific measurement proceedings, the obstruction to IWT is reduced and relevant information is obtained with low effort in cost and time. Broadcasting the measured data to other vessels or web-interfaces also allows for a self-updating inland waterway chart, contributing to easier and safer navigation along inland waterways.
The development of a self-updating inland waterway chart requires several important cornerstones. An overview of the project idea is illustrated in Fig. 1.1. First of all, the data acquisition is performed by one or more optical sensors, gathering precise range measurements (such as Light detection and Ranging (LiDAR) sensors), as well as data that serves for semantic scene understanding (provided by single or stereo cameras). The collection of LiDAR point clouds and stereo images needs to be processed for essential scene understanding and object classification. The specific shape of a bridge can be modelled mathematically, resulting in a precise description of the on-scene shapes and ranges. Precise global mapping (and map matching) is used to find the exact geographic position of the detected object. Finally, the updated charts can be published through an appropriate platform.

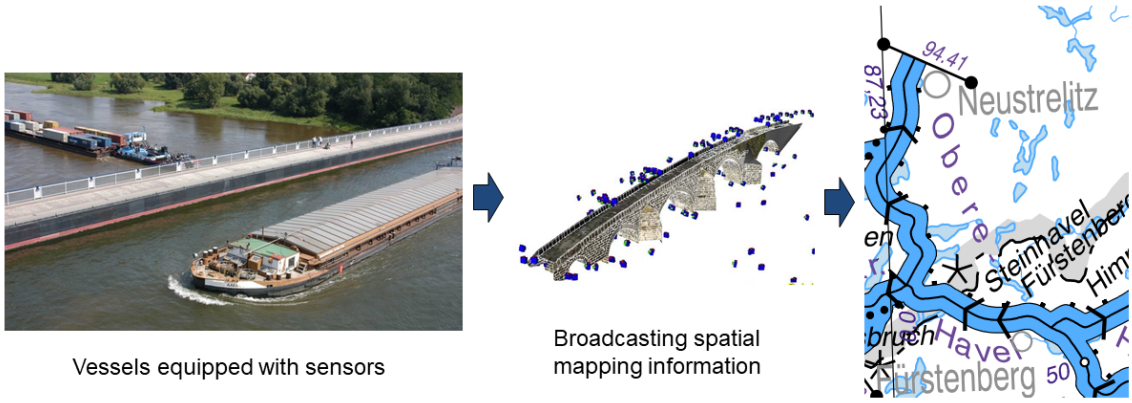


Figure 1.1: Scope of the generation of a self-updating inland waterway chart. Sensor data is retrieved by inland vessels, translated into spatial mapping information and broadcasted to the chart interface. Images retrieved from (left to right) [5], [6] and [7].

1.2 Contribution of this work

As a first step towards self-updating inland waterway charts, optical sensor data needs to be acquired and processed to gain awareness of the traffic depiction and the infrastructure surrounding the vessel. Using visual perception systems, the following information levels can be distinguished:

- Physical description: pose, speed and shape of objects
- Semantic description: categories of objects
- Intention prediction: likelihood of the object's behaviour

The main focus of this work is semantic description based on the input of the mentioned optical sensors. In general, this task is mostly performed on RGB images, as they contain more (accurate) semantic information than LiDAR point clouds [8].
Semantic Segmentation describes the goal of assigning a pre-defined class-value to every pixel of the image [9]. As a state-of-the-art approach, this goal is achieved by machine learning (ML), in particular by training an artificial neural network. The result is a so-called segmentation mask, visualizing all classified pixels. The classes can be defined with respect to the application of the segmentation algorithm.

A LiDAR sensor emits laser beams and measures the reflection received back from the environment [10]. The output from a 3D sensor consists of a point cloud, containing the 3-dimensional position and the intensity of the received reflection for every point detected. By this way, spatial information of the environment can be retrieved mostly independently of light conditions and at high precision up to ranges of 200 m (depending on the specific device in use), outperforming stereo camera range estimation in both, maximum range and accuracy [11]. In general, LiDAR sensors are often bulkier and considerably expensive than stereo cameras. Object recognition on point-clouds is a challenging task and the large input datasets are computationally intense. In contrast, stereo cameras allow for semantic scene understanding at comparably low effort, but poor range estimation. The combination of the two sensors can therefore leverage the flaws of each other [11].
The goal of this work is to establish and compare methods to perform Semantic Segementation in inland waterway environments. Regarding the use of sensor data, the main focus will therefore be on RGB images for accurate semantic description, using the precise range measurements by LiDAR point clouds as an additional information to achieve accurate spatial mapping.

1.3 Outline

An introduction and motivation for the topic was given in this Chapter. Chapter 2 provides an overview of state-of-the-art ML techniques applied for semantic segmentation. In Chapter 3, the applied model is presented. Furthermore, an overview over the valuable dataset MaSTr1325 is given. Section 3.3 describes the generation of the BerlinIWT dataset, developed in this work to assist in IWT semantic segmentation. Finally, the results of the presented methods are discussed. Chapter 4 presents state-of-the-art methods for spatial mapping assisted by LiDAR sensors and discusses methods for aligning 3D point clouds to RGB images. Furthermore, an overview of optical flow estimation techniques is given in Section 4.4. Finally, Chapter 5 concludes the work and evaluates the application in a traffic-telematics related frame.

2. Semantic Segmentation on Images

Semantic image segmentation describes the task of grouping image regions together, that belong to the same semantic class and then assigning each pixel to one of the pre-defined classes [12]. This procedure is also known as pixel-wise classification [13]. Semantic segmentation plays a major role in image understanding problems, that are often used in a plethora of applications, including surveillance, computer graphics and autonomous vehicles.

2.1 Related Work

With the application of Convolutional Neural Networks (CNN), the abilities in recognition tasks as image classification and bounding box detection have substantially improved over the last years [14]. The great variety of available neural networks assists in the development of fine-grained labelling [15].
Automotive applications are a popular use-case for semantic segmentation algorithms. A large field of research is dedicated to autonomous road vehicles [16] and the associated task of road segmentation [17] as well as obstacle detection [18]. For the data-hungry CNNs, appropriate training data is a crucial parameter. For the particular case of self-driving cars, the KITTI benchmark has evolved to one of the most popular datasets [19], [20]. The data is fully annotated, publicly available and contains measurements from a LiDAR sensor, two stereo cameras (colour and grayscale) and an Inertial Measurement Unit (IMU). The datasets are recorded during daytime in different road-based environments. UrbanLoco [21] is another publicly available, fully annotated dataset for the application on autonomous vehicles. In contrast to the KITTI dataset, UrbanLoco is collected in urban environments only, addressing especially the

challenge of urban canyons and tunnels. Measurements from LiDAR sensors, cameras, IMUs and Global Navigation Satellite System (GNSS) receivers are included. The authors of [22] provide a comparably small dataset containing radar, LiDAR and camera data.

With the development of so-called Unmanned Surface Vehicles (USV), autonomous applications enter the maritime domain. An obstacle detection approach for stereo camera semantic segmentation for USVs is presented by [23]. The algorithm is further strengthened by the additional use of an IMU [24]. In contrast to the approach used on open water applications, Roboat [25] is developed as an autonomous surface vehicle for urban waterways. The authors address localization, path planning and obstacle avoidance by using LiDAR, camera and IMU information in areas that suffer from limited GNSS availability. In addition, Roboat II [26] can carry out sophisticated path planning as well as accurate Simultaneous Localization and Mapping (SLAM). Also for the application of USVs, labelled data is a crucial requirement for training data-driven approaches [27]. Just like road segmentation for self-driving cars, water segmentation is a significant ability for USVs for the identification of navigable areas. Different approaches to this task are evaluated in [28]. The authors of [29] train and test deep learning algorithms to address the use-case of river segmentation for flood monitoring. In [30], a model for water detection is proposed to be applied to self-operating outdoor robots. In [13], different colour spaces (e.g. Hue, Saturation, Value (HSV) and RGB) are evaluated with respect to their robustness in water segmentation, especially for scenarios with rapidly changing light conditions. While the variety of datasets related to self-driving cars is large, comparable datasets for the maritime domain are very rare. The authors of [31] address this problem by providing a high-resolution dataset collected in a nordic lake environment. The dataset contains 600 manually labelled images with a variety of weather conditions and can be used for water segmentation only. The USVInland dataset [32] is collected on various sections of inland waterways by LiDAR sensors, stereo cameras, a millimeter-wave radar, GNSS and IMUs. It contains 700 images that can be used for water segmentation. Inland waterway environments hold challenges such as complex distribution of obstacles, possible GNSS outages, the reflection of bank-side structures as well as possible fog over the water surface [32]. Even though, these challenges might not be experienced with comparable frequency

and extent when sailing in open water, the MaSTr1325 dataset [33], collected in the coastal waters of Koper (Slowenia) is also of great interest. Indeed, one major aim of this research is to distinguish robustly between sky and water surface by using the assistance of an IMU. However, for gathering reliable spatial mapping information, performing water segmentation is not sufficient.

2.2 Applied Methodology

The commonly known Artificial Neural Networks (ANN) are inspired on the behaviour of biological nervous systems [35]. CNNs are a type of ANN, that is primarily used to recognise patterns in images. In general, an ANN consists of an input layer, e.g. the pixel values of the image and an output layer, e.g. the values of the final segmentation mask. In between those layers, a variety of hidden layers is located to perform the actual task by processing the values transmitted from the input layer. The behaviour (and thus the result) of a Neural Network (NN) depends on the parameters of the network. Namely, a weight $w \in \{-1, 1\}$ is assigned to the connection between two neurons and a bias $b \in \mathbb{R}$ is allocated to every neuron, that is part of a hidden layer. A neuron can be thought of as a function, that transfers its input to subsequent neurons only if a certain activation threshold o is reached. Considering a fully connected network, as it is depicted in Fig. 2.1, the activation function of specific a neuron u can be expressed as [44]

$$o(w_1 o_1 + w_2 o_2 + ... + w_j o_j - b_u) \tag{2.1}$$

The activation function of a neuron in layer l as part of a fully connected network can thus be generalized to

$$a^{(l)} = a_f(w a^{(l-1)} + b) \tag{2.2}$$

The type of activation function a_f is to be chosen [37]. A popular state-of-the-art choice is the *ReLu* function, describing the *rectified linear unit* function as depicted in Fig. 2.2.

2.2.1 Modes of Learning

The goal of the "learning" or "training" process of a NN is to find the weights and biases that fit best for the desired output. Depending on the application of the NN,

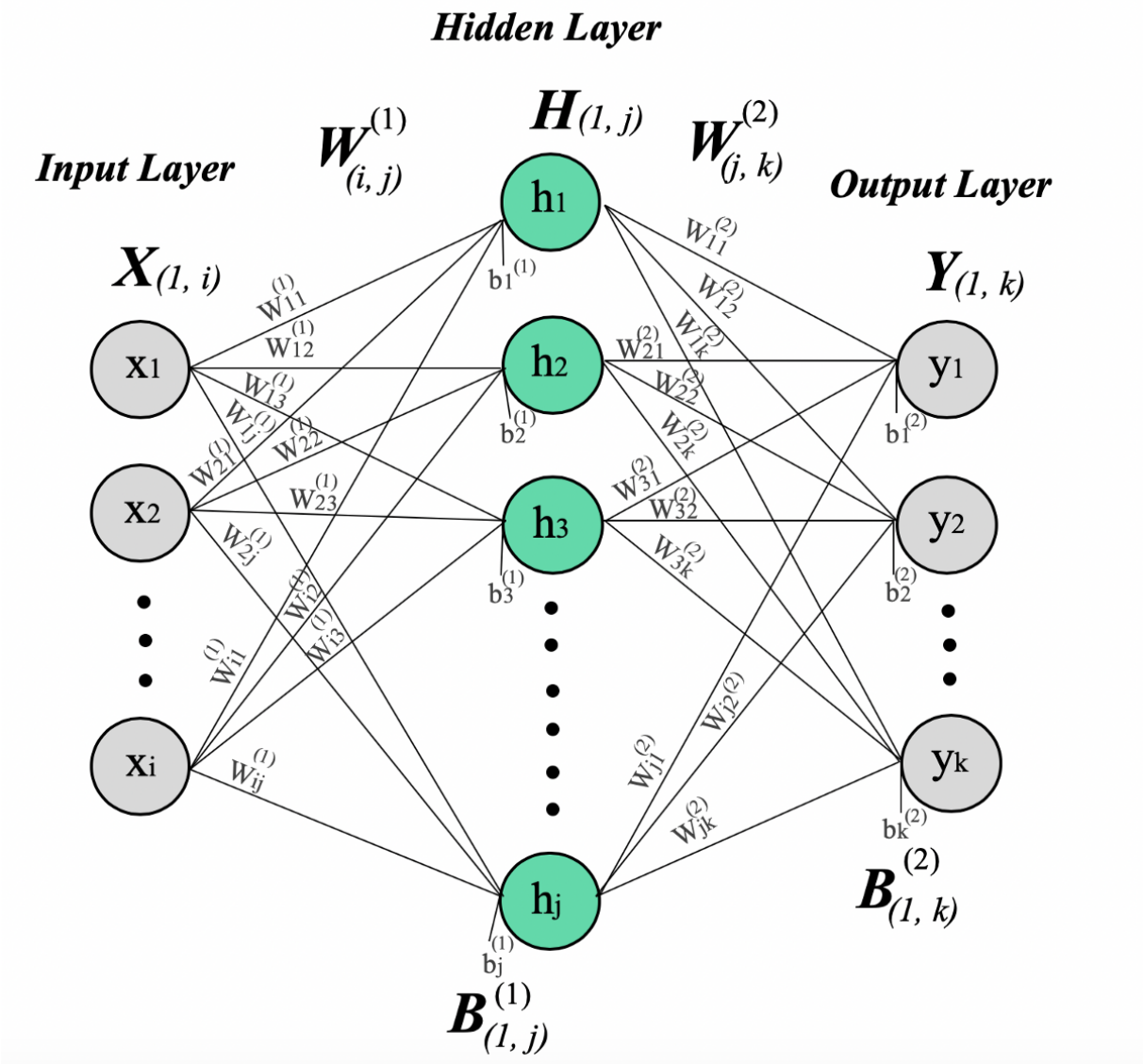


Figure 2.1: Basic structure of a neural network (NN) [36]. Here, a three layer NN is depicted with input dimension i and output dimension k.

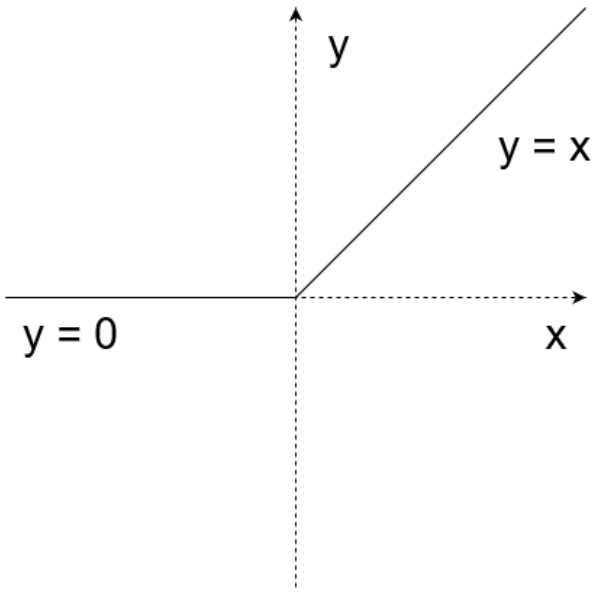


Figure 2.2: The *Rectified linear unit (ReLu)* function is one of the most commonly used activation functions for NN.

either supervised or unsupervised learning may be used.
Supervised learning is mostly used in semantic segmentation on RGB images. The network is trained on a distinct training dataset that contains a desired output (for this use-case a segmentation mask) for every input image. The objective in this form of training is to minimise the overall classification / labelling error of the model. To achieve this, the weights and biases are adapted after every training example so that the segmentation mask predicted by the model matches the true mask as closely as possible.
Unsupervised learning training datasets do not include any labels [35]. Instead, the model is trained to minimise an associated cost function. This technique is used in feature extraction [38], [39], traffic engineering, internet traffic classification, anomaly detection as well as quality of service optimization.

2.2.2 Learning Process

Usually, the input of a NN is passed from the input layer through all hidden layers towards the output layer, the so called *forward propagation*. The so-derived output can then be compared to the desired output [40]. During training, *backward propagation* is

used to adapt the parameters of the NN and achieve the desired output.
A single example is expressed as x with its desired output y. In contrast, the expressions $\mathbf{x}$ and $\mathbf{y}$ illustrate an input and an output matrix (e.g. an image and a desired output mask), respectively. A defined loss function $L(\hat{y}, y)$ models how well one single estimate of the NN $\hat{y}$ fits the desired output example y. A generic cost function can be expressed as

$$J(w, b) = \int L(z, w) dp(z), \tag{2.3}$$

where

$$\begin{aligned} z^{[l]} &= w^{[l]} g^{[l-1]} + b^{[l]} \\ g^{[l]} &= ReLu(z^{[l]}) \end{aligned} \tag{2.4}$$

models the costs over all training examples. The expression $z^{[l]}$ is computed with respect to the parameters w and b, as well as the activation function $g^{[l]}$ of layer l. The minimisation of the cost function $J(w, b)$ in this representation containing the empirical distribution $dp(z)$ would be too costly in time and computational effort. Therefore, *gradient descent* is applied in order to find a local minimum of the cost [41]:

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}, y). \tag{2.5}$$

A commonly used loss function is referred to as *cross entropy*, which is especially interesting to classification problems [42], and described by

$$L(\hat{y}, y) = y^{(i)} \log \hat{y}^{(i)}. \tag{2.6}$$

As described later in Section 2.2.3, the derivative of the cost function influences the evolution of the parameters inside the NN. The partial derivative of Eq. (2.5) yields the likelihood of the predicted classes, which can be compared to the true class. Further derivation of the mathematical details can be consulted at [42].
In practical implementations, mathematical operations are not carried out explicitly by iterating over each of the training examples. Instead, *vectorization* is applied: all training examples $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ are stacked together to create one single vector.

$$\mathbf{X} = \left[x^{(1)}, x^{(2)}, \ldots, x^{(m)} \right] \tag{2.7}$$

The vectors $\mathbf{W}^{[l]}$ and $\mathbf{B}^{[l]}$ contain the weights and biases associated with all hidden units h in layer l. The parameters $\mathbf{W}^{[l]}$ and $\mathbf{B}^{[l]}$ are initialized with random numbers. Throughout the process of gradient descent, the parameters are adjusted by the NN to minimize the function $J(\mathbf{W}, \mathbf{B})$. For illustration purposes, an example for a single iteration of a small NN with two hidden layers is described next. Thus, consider:

- The parameter vectors $\mathbf{W}^{[1,2]}$ and $\mathbf{B}^{[1,2]}$ contain the weights and biases for the first and second layer.
- The cost function is defined by $J(\mathbf{W}^{[1]}, \mathbf{B}^{[1]}, \mathbf{W}^{[2]}, \mathbf{B}^{[2]}) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}, y)$ over all m training examples.

The input $\mathbf{X}$ is propagated forward through the network:

$$\mathbf{Z}^{[1]} = \mathbf{W}^{[1]}\mathbf{X} + \mathbf{B}^{[1]} \tag{2.8}$$

$$\mathbf{A}^{[1]} = g^{[1]}(\mathbf{Z}^{[1]}) \tag{2.9}$$

$$\mathbf{Z}^{[2]} = \mathbf{W}^{[2]}\mathbf{A}^{[1]} + \mathbf{B}^{[2]} \tag{2.10}$$

$$\mathbf{A}^{[2]} = g^{[2]}(\mathbf{Z}^{[2]}) \tag{2.11}$$

where $\mathbf{A}^{[l]}$ represents the result of the activation function $g()$ and the current layer $\mathbf{Z}^{[l]}$. For parameter adjustment, the result of the output layer is compared to the desired output $\mathbf{Y}$. The parameters are then adjusted during back propagation:

$$d\mathbf{Z}^{[2]} = \mathbf{A}^{[2]} - \mathbf{Y} \tag{2.12}$$

$$d\mathbf{W}^{[2]} = \frac{1}{m} d\mathbf{Z}^{[2]} \mathbf{A}^{[1]\top} \tag{2.13}$$

$$d\mathbf{B}^{[2]} = \frac{1}{m} \sum d\mathbf{Z}^{[2]} \tag{2.14}$$

$$d\mathbf{Z}^{[2]} = \mathbf{W}^{[2]\top} d\mathbf{Z}^{[2]} \cdot g^{[1]}(\mathbf{Z}^{[1]}) \tag{2.15}$$

$$d\mathbf{W}^{[1]} = \frac{1}{m} d\mathbf{Z}^{[1]} \mathbf{X}^{\top} \tag{2.16}$$

$$d\mathbf{B}^{[1]} = \frac{1}{m} \sum d\mathbf{Z}^{[1]} \tag{2.17}$$

Finally, all weights and biases in each layer l are adjusted using gradient descent by

$$\begin{aligned} \mathbf{W}^{[l]} &= \mathbf{W}^{[l]} - \alpha d\mathbf{W}^{[l]} \\ \mathbf{B}^{[l]} &= \mathbf{B}^{[l]} - \alpha d\mathbf{B}^{[l]} \end{aligned} \tag{2.18}$$

with

$$\begin{aligned} d\mathbf{W}^{[l]} &= \frac{dJ}{d\mathbf{W}^{[l]}} \\ d\mathbf{B}^{[l]} &= \frac{dJ}{d\mathbf{B}^{[l]}} \end{aligned} \tag{2.19}$$

where the learning rate α is to be determined as a *hyperparameter* in advance. More details can be found in section 2.2.5.

2.2.3 Optimization Algorithms

To train a NN, the cost function (2.5) is to be minimized. *Gradient Descent* is an algorithm that finds a set of variables that minimize a given target function (such as the cost function). It computes the gradient of the cost function to perform a parameter update for all of the training examples $\mathbf{X}$ with their associated labels $\mathbf{Y}$ as stated by Eq. (2.18) and (2.19). In fact, updating the parameters of the network with respect to every single training example, instead of computing the gradient descent of the cost function for the entire training set at on time, avoids redundant computation and saves computational time [43]. Note that if the learning rate is set too high, the gradient descent algorithm might not converge to a minimum of the cost function. Nevertheless, it has been shown, that by slowly decreasing the learning rate the convergence to a minimum is almost certain [43].
Gradient descent can also be applied to smaller subsets of the dataset (so called *mini-batches*). This may result in a less time consuming computation, as the empirical process can be optimized on smaller parts of the training set. The dataset, consisting of the input $\mathbf{X}$ and the desired output $\mathbf{Y}$ is divided into mini-batches $\mathbf{X} = \mathbf{X}^{\{1\}}, \ldots, \mathbf{X}^{\{t\}}$ and $\mathbf{Y} = \mathbf{Y}^{\{1\}}, \ldots, \mathbf{Y}^{\{t\}}$. According to Eqs. (2.8) and (2.11), each mini-batch is propagated

through the network by

$$\begin{aligned} \mathbf{Z}^{[1]} &= \mathbf{W}^{[1]}\mathbf{X}^{\{t\}} + \mathbf{B}^{[1]} \\ \mathbf{A}^{[1]} &= g^{[1]}(\mathbf{Z}^{[1]}) \\ &\vdots \\ \mathbf{A}^{[K]} &= g^{[K]}(\mathbf{Z}^{[K]}) \end{aligned}$$

The costs for the current training example i are computed by

$$J^{\{t\}} = \frac{1}{m}\sum_{i=1}^{l} L(\hat{y}^i, y^i) \tag{2.20}$$

and the gradients of $J^{\{t\}}$ using $\mathbf{X}^{\{t\}}$ and $\mathbf{Y}^{\{t\}}$ by the Eqs. (2.18) and (2.19). During one *epoch* of training, this process is carried out for all of the mini-batches formed by the training set [43].

Gradient descent can be speeded up using *RMS Prop* (Root-mean-square-prop). RMS Prop restricts the direction for the convergence steps to avoid time consuming oscillations during the process. In this case, a larger learning rate might be applied to achieve a faster convergence [43].

$$\mathbf{W} = \mathbf{W} - \alpha\frac{d\mathbf{W}}{\sqrt{S_{dW}} + \epsilon} \tag{2.21}$$

$$\mathbf{B} = \mathbf{B} - \alpha\frac{d\mathbf{B}}{\sqrt{S_{dB}} + \epsilon} \tag{2.22}$$

with

$$\mathbf{S}_{dW} = \beta\mathbf{S}_{dW} + (1-\beta)d\mathbf{W}^2 \tag{2.23}$$

$$\mathbf{S}_{dB} = \beta\mathbf{S}_{dB} + (1-\beta)d\mathbf{B}^2. \tag{2.24}$$

The moving average parameter β and the additional hyperparameter ϵ need to be set. A popular choice is $\epsilon = 10^{-8}$ to avoid division by zero [43].

Finally, the well known optimizer *Adaptive Moment Estimation (Adam)* combines different well-working optimization methods. The iterative process is initialized with the parameters values $\mathbf{V}_{dW}, \mathbf{S}_{dW}, \mathbf{V}_{dB}, \mathbf{S}_{dB}$ as vectors of zero. First, exponentially decaying averages of past gradients are stored. This approach is also known as momentum term [43]:

$$\mathbf{V}_{dW} = \beta_1\mathbf{V}_{dW} + (1-\beta_1)d\mathbf{W} \tag{2.25}$$

$$\mathbf{V}_{dB} = \beta_1 \mathbf{V}_{dB} + (1 - \beta_1) d\mathbf{B} \tag{2.26}$$

As stated earlier, RMS Prop is used as an estimate for the variance [43]. The variables S_{dW} and S_{dB} are derived from (2.23) and (2.24), where β is replaced by the hyper-paramter β_2. As the values $\mathbf{V}_{dW}, \mathbf{S}_{dW}, \mathbf{V}_{dB}, \mathbf{S}_{dB}$ are initialized to zero, the algorithm is prone to be biased towards zero. To counteract this effect, bias correction is applied as follows.

$$\mathbf{V}_{dW}^{corr} = \frac{\mathbf{V}_{dW}}{1 - \beta_1^t} \tag{2.27}$$

$$\mathbf{V}_{dB}^{corr} = \frac{\mathbf{V}_{dB}}{1 - \beta_1^t} \tag{2.28}$$

$$\mathbf{S}_{dW}^{corr} = \frac{\mathbf{S}_{dW}}{1 - \beta_2^t} \tag{2.29}$$

$$\mathbf{S}_{dB}^{corr} = \frac{\mathbf{S}_{dB}}{1 - \beta_2^t} \tag{2.30}$$

Finally, the weight and bias are updated

$$\mathbf{W} = \mathbf{W} - \alpha \frac{\mathbf{V}_{dW}^{corr}}{\sqrt{\mathbf{S}_{dW}^{corr}} + \epsilon} \tag{2.31}$$

$$\mathbf{B} = \mathbf{B} - \alpha \frac{\mathbf{V}_{dB}^{corr}}{\sqrt{\mathbf{S}_{dB}^{corr}} + \epsilon} \tag{2.32}$$

with the learning rate α to be determined and tuned, $\beta_1 = 0.9$ as default value, $\beta_2 = 0.999$ as recommended value and $\epsilon = 10^{-8}$ to prevent zero division. The parameter ϵ, however, does not play a major role on the performance. The Adam optimizer is well-known and widely used among the ML community. It is an effective method to compute adaptive learning rates for each parameter.

2.2.4 Convolutional Neural Networks

A CNN is an ANN, where at least one layer performs convolutional operations. In this convolutional layer, the input is propagated through at least one *filter* (also named *kernel*) instead of a simple multiplication with the activation function. For the size of the filter, $(3, 3)$ is a popular choice. In general, the parameters inside the filter are

learned by the network. An example is provided below.

$$\mathbf{f} = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.33}$$

This small example depicts a filter detecting upper horizontal edges in images: the value -1 represents black pixel values, 1 white and 0 grey coloured pixels. Even though also different filter sizes can be chosen, the filter is usually considerably smaller than the image. After processing the receptive field of the filter for the first time, it is

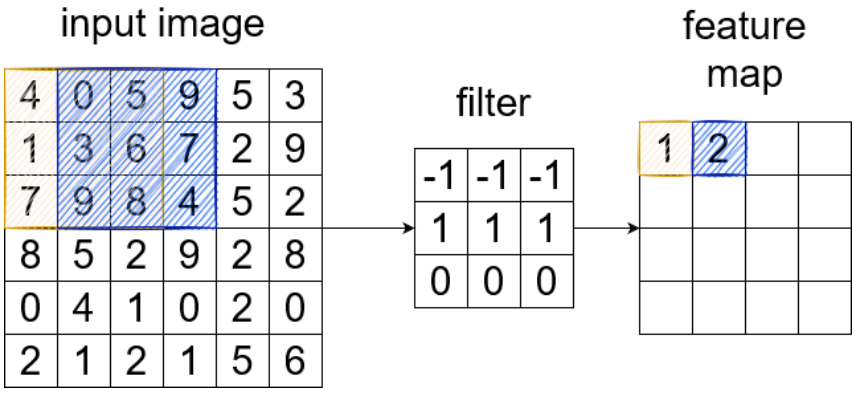
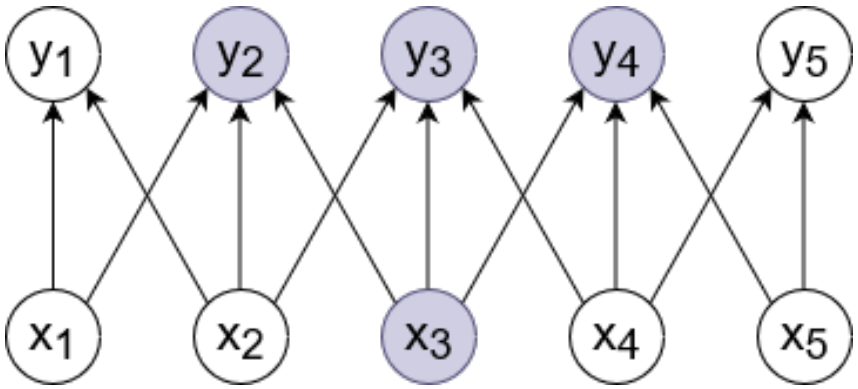


Figure 2.3: Convolutional operation with stride of 1

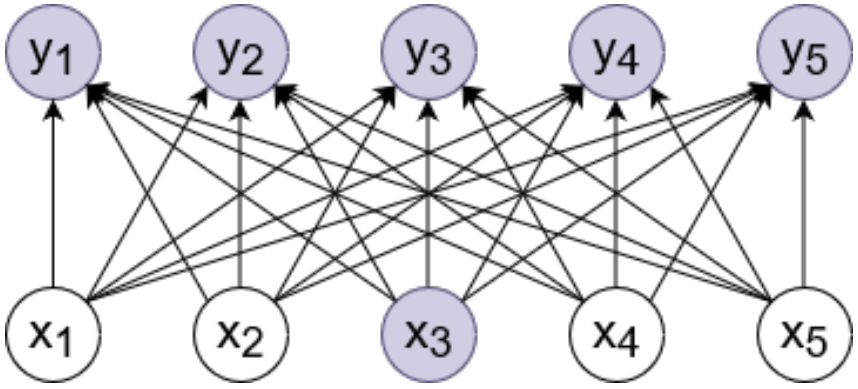
shifted (by s pixels, the so called *stride*) to the next subset of pixels as depicted in Fig. 2.3. As a result of each of these convolutional operations, one pixel value of the output feature map is computed. Sifting the filter by only one pixel over the input image results in large overlapping receptive fields [35]. Therefore, the stride can be increased in order to reduce the overlap within the convolutions.
In ML, a *convolution* is referred to an operation as follows [44]:

$$\mathbf{S}(i,j) = (\mathbf{K} * \mathbf{I})(i,j) = \sum_m \sum_n \mathbf{I}(i+m, j+n)\mathbf{K}(m,n) \tag{2.34}$$

which is also known as *cross-correlation*. This convolution without kernel flipping (cross-correlation) is implemented in many ML libraries [44]. In this work, the above stated operation shall be referred to as convolution, as it is the case in a broad spectrum of the ML community, although this notation may interfere with the notion of "convolution" coming from the signal processing verbose. While the filter size is pre-defined,



(a) *Convolutional layer*. The output **y** is formed by a convolution with a kernel of width 3, so three output units are affected [44].



(b) *Fully connected layer*. Connections are not sparse, instead both layers are fully connected.

Figure 2.4: *Sparse connectivity* in comparison to a fully connected layer with the impact of x_3 highlighted. The output **y** is formed by a matrix multiplication with a kernel of width 3 [44].

the actual filter values are to be learned by the network. Therefore, the network will also learn whether the kernel is flipped or not. To detect a large variety of visual features, the amount of hidden layers is usually considerably high in a CNN. This is also referred to as *deep network* [45]. To avoid excessively high complexity of a CNN, the fully connected network is adapted by the concept of sparse connections. As depicted in Fig. 2.4, a convolutional layer contains less connections than a fully connected (FC) layer [46]. Some of the available connections are assigned with weight 0 and are no longer part of the network. For the remaining connections, fewer parameters need to be computed and the complexity of the system decreases without any drawbacks on the performance of the network [47].

Parameter sharing is another approach to reduce the computational costs while training a NN. For instance, discovering that a certain filter is useful in a certain region of the image leads to the use of the same filter for other regions [35]. Based on this assumption, the weights and biases are constrained accordingly. The decreased amount of parameters to be learned by the NN leads to improved learning characteristics.
As the input image is convolved with the filter, the dimensions of the output feature map will change with respect to the original input. Considering an image of size (n, n) convolved with a filter of size (f, f), the output feature map will have dimensions $(n - f + 1, n - f + 1)$. This operation is denoted as *valid convolution* [45]. To prevent feature maps from shrinking, *padding* can be applied. Padding describes the method of adding p rows and columns to the input image in order to produce a feature map of dimensions (n, n). In order to perform this *same convolution*, the parameter p can be derived by

$$p = \frac{f - 1}{2} \tag{2.35}$$

with $f \in$ odd values only. As the added values are usually equal to 0, this is also referred to as *zero-padding* [45]. In general, the height n_H and width n_W of the output feature map can be computed by

$$(n_H, n_W) = \left(\left\lfloor \frac{n + 2p - f}{s} + 1 \quad , \quad \frac{n + 2p - f}{s} + 1 \right\rfloor \right) \tag{2.36}$$

with dimensions of the input image (n, n), filter size (f, f), stride s and padding p. Colour images (e.g. RGB images) are of dimension (n, n, c) with c representing the number of channels. Every pixel value is thus encoded in each of the (mostly three) colour channels. In this case, the convolution is performed over all the colour channels by applying filters of size (f, f, c), so the associated feature map has only one channel. In practical applications, n_c filters are applied within the same convolutional layer, so different features can be detected in one single layer and among all three channels. As every single convolution leads to a different feature map, the resulting feature maps are stacked together, yielding a feature map of dimension (n_H, n_W, n_c) [44].
Pooling helps to focus on the most relevant features detected and can be applied as separate operation within a layer of a CNN [44]. The feature map obtained from the previous operation is divided in sub-regions (e.g. of size (2,2) pixels) as depicted in Fig. 2.5. From each sub-region, only one value will be kept for subsequent operations.

This value can be either the average of the defined sub-region (*average pooling*) or the maximum value of the sub-region (*max pooling*), which is the more popular choice. Furthermore, pooling is a valuable tool to achieve a model that is more robust against

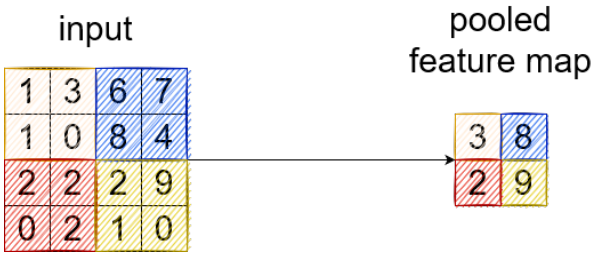


Figure 2.5: *Max pooling* is a technique used to detect the most relevant features of the map. Of each sub-region indicated, only the maximum value is kept for further processing. [44].

small changes of the input feature map [44]. *Invariance to translations* states that, for small translations of the feature map, most pooled outputs will not change [44]. In order to keep the dimensions of the feature maps, padding can be combined with pooling [48].

Finally, Fig. 2.6 represents the structure of a small CNN as it could be implemented to recognize hand-written digits from the MNIST dataset [49]. The MNIST dataset is designed to train networks to map a hand-written digit to the associated digital one. The Fig. 2.6 derived from [35] depicts a structure, that tends to be common among the ML community [35]: the input image (here of size $(28, 28)$ [49]) is first processed by a convolutional layer using a ReLu activation function. The following pooling layer detects the most relevant features from the previously computed feature map. A FC layer with ReLu activations is used for up-sampling. The purpose of the last FC layer is to map the detected features to one of the pre-defined possible outputs: in this case a number between 0 and 9.

2.2.5 Train / Test Splitting

Rigorous training on a representative dataset is an important factor for the performance of a NN. In general, CNNs perform best when trained on a large dataset, or at least on a dataset with strong variations in the data.

The complete dataset is typically split into: i) *training set* is the largest subset and used

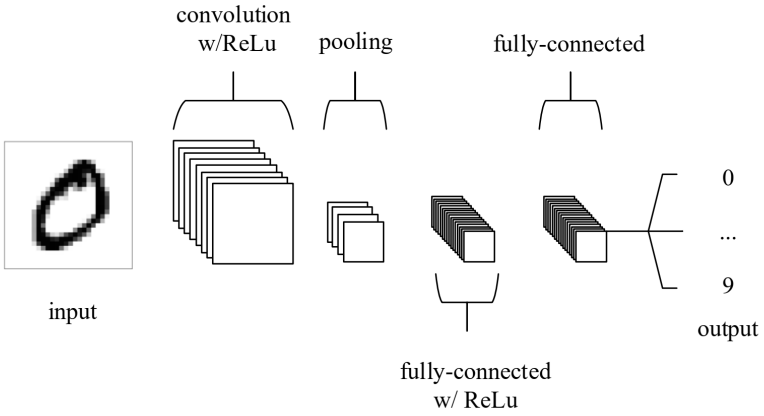


Figure 2.6: Example of a small CNN as presented by [35]. This example may be used to recognize hand-written digits, as presented in [49].

for the training process; ii)*development set* can be used during training for validation of the training results and to compare the performance of different algorithms. It is important to keep this subset separate from the training set, as this is the only possibility to evaluate the performance using data new to the algorithm. iii) The *test set*, that is used for the final evaluation of the readily trained algorithm. The size of the mentioned subsets depends on the total number of examples available from the whole dataset: $100 - 10.000$ examples can be split into subsets of 60 %, 20 %, 20 % for training, development and testing, respectively. In some cases, the test set may even be discarded and the development set may be used for testing. Then, the training set would consist of 70 % and the development set of 30 % of the whole dataset. For larger datasets, i.e. in the order of 10^6 examples, a splitting of 98 %, 1 %, 1 % or even 99.5 %, 0.4 %, 0.1 % for training, development and testing, respectively, can be applied.

2.2.6 Overfitting and Regularization

The error measures for training and development sets are a valuable source for performance evaluation. A high error in both training and development set indicates a high estimation *bias*. The term bias does not address any difference in the training and development set error: it only describes the error level in both sets. The problem of a high bias can be addressed by using a larger network or extending the training process.

The difference of the training set error and development set error is referred to as

train set error	low	high	high	low
dev set error	high	high	even higher	low
bias	low	high	high	low
variance	high	low	high	low

Table 2.1: Overview of bias and variance in relation to training and development set error.

variance. A high variance can be observed, e.g., when a low training set error in combination with a high development set error occurs. This effect is also named overfitting: the NN has been trained excessively on the (limited) training set. Its capabilities to generalize the learned behaviour are therefore lacking. Overfitting behaviour is illustrated in the first column of Tab. 2.1. Furthermore, the table provides an overview over other possible combinations of bias and variance and possible behaviour of the network. Besides overfitting, this includes "underfitting" behaviour (second column of Tab. 2.1), a combination of over- and underfitting (third column) and finally the desired behaviour exhibiting decent generalization abilities (last column).
From an implementation point-of-view, the easiest way to counteract overfitting is the usage of a bigger dataset for training. However, due to lack of training data or time to acquire it, this might not be always possible. In this case, regularization methods, such as L_2 Norm (also called *weight decay*) can be applied [50]. The cost function is re-defined by

$$J(\mathbf{W}^{[l]}, \mathbf{B}^{[l]}, \ldots, \mathbf{W}^{[K]}, \mathbf{B}^{[K]}) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^{K} ||\mathbf{W}^{[l]}||_F^2 \tag{2.37}$$

where the second part of the equation $\frac{\lambda}{2m} \sum_{l=1}^{K} ||\mathbf{W}^{[l]}||_F^2$ implements the regularization with Frobenius norm $||\mathbf{W}^{[l]}||_F^2$ of $\mathbf{W}^{[l]}$. The update step for the weights and biases in the network defined by Eq. (2.19) is adapted [50], so that

$$d\mathbf{W}^{[l]} = \frac{dJ}{d\mathbf{W}^{[l]}} + \frac{\lambda}{m} \mathbf{W}^{[l]} \tag{2.38}$$

The computation of the weights and biases by gradient descent remains unchanged [50] as in Eq. (2.18):

$$\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha d\mathbf{W}^{[l]} \tag{2.39}$$

Overfitting can also be prevented by other regularization methods such as *data augmentation*. Within this process, images are adapted during the training process by horizontal or vertical flipping, random rotation and random zooming or cropping. Also, random adjustments on colour, brightness, contrast and saturation can be used to augment the dataset. Even though these methods seek to increase the variety of the dataset, the training set might remain slightly redundant. However, data augmentation is a very inexpensive regularization method [51]. Most built-in libraries apply data augmentation during the training process, so that performance will not suffer from larger datasets. Augmentation may only be applied to the training, not to development or test set.
Early stopping is a technique, consisting on finalizing the training process as soon as the error computed on the validation set starts to increase during training. A further decreasing training error, but increasing development error would exhibit overfitting. With early stopping, the model is able to generalize its inputs fairly well [51].

2.2.7 Batch Normalization

Looking at the different hidden layers of a network, the distribution of the layer input might vary as the parameters of the previous layers change [52]. This effect is called *internal covariate shift* and can make the training process slow and hard to tune. It is counteracted by the approach of *batch normalization*, also called *batch norm*, that aims at normalizing the activations of every layer in the network. For the intermediate values inside the l-th layer of the NN, $z^{(i)}, \ldots, z^{(m)}$, mean and variance can be computed by [52]

$$\mu = \frac{1}{m} \sum_i z^{(i)} \tag{2.40}$$

$$\sigma^2 = \frac{1}{m} \sum_i (z^{(i)} - \mu)^2 \tag{2.41}$$

The normalized inputs are the derived from

$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.42}$$

In order not to normalize the values necessarily to a zero-mean normal distribution of unit variance, $\tilde{z}^{(i)}$ is computed as

$$\tilde{z}^{(i)} = \gamma z_{norm}^{(i)} + \beta_l \tag{2.43}$$

with the parameters γ and β_l to be learned by the network. The mean of $\tilde{z}^{(i)}$ can thus be set to an arbitrary value controlled by β_l and γ [52].
To implement batch norm in a deeper NN (a NN containing various hidden layers), the normalized value $\tilde{z}^{(i)}$ is computed from the output of the layer $z^{(i)}$. The value $\tilde{z}^{(i)}$ is then propagated to the subsequent layer in place of $z^{(i)}$. Batch norm is usually applied to mini-batches, so that $\tilde{z}^{(i)}$ is computed for every mini-batch $\mathbf{X}^{\{t\}}$ separately. Whereas the parameters β_l and γ are additional parameters to be learned by the network, the bias parameter $b^{[l]}$ is zeroed out during the normalization process [52]. The associated parameter vector $\mathbf{B}^{(l)}$ can therefore be neglected if batch norm is applied. During the implementation the optimization algorithm, the parameters $\mathbf{W}^{[l]}, \boldsymbol{\beta}_l^{[l]}, \boldsymbol{\gamma}^{[l]}$ are calculated as in Eq. (2.19) [52]:

$$\boldsymbol{\beta}_l^{[1]} = \boldsymbol{\beta}_l^{[1]} - \alpha d\boldsymbol{\beta}_l^{[1]}.$$

Batch norm reduces the amount that the parameters need to be adapted during the learning process and layers can learn more independently from each other [52]. Even though the scaling on μ and σ^2 adds a small amount of noise to the function $z^{[l]}$, which introduces a small regularization effect, batch norm is not a regularization method [52].

2.2.8 Hyperparameter Choice

The training process of NN is heavily influenced by the choice of *hyperparameters*. Those hyperparameters need to be chosen carefully and may also evolve throughout the training process.

- The *learning rate* α determines the convergence speed for the optimization and, therefore, how fast the network learns. More precisely, the weights $\mathbf{W}^{[l]}$ of the network are adjusted while performing gradient descent (as described in Section 2.2) with respect to a cost function J. As stated by Eq. (2.18), the weights are updated during backpropagation by $\mathbf{W}^{[l]} = \mathbf{W}^{[l]} - \alpha \frac{dJ}{d\mathbf{W}^{[l]}}$ [53]. Indeed, the learning rate is the most important hyperparamter to configure the training environment

of a NN [53]. In many cases, the learning rate is determined experimentally [53]: multiple networks with different learning rates are computed in order to compare their performance. Due to the limitation of computational capacity, this might not always be possible. Another approach is to follow the training process of the model closely to recognize, how different learning rates affect the training process. Nevertheless, the ML community has converged on the idea of using algorithms which systematically adapt the learning rate throughout the training process [53].

- Momentum term can be thought of as the second important hyperparameter. With the application of the Adam optimization algorithm, the parameters are usually chosen to $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 10^{-8}$, as described in Section 2.2.3.

- Splitting the dataset into smaller mini-batches generally is a valuable opportunity to enhance faster training. Whereas larger mini-batches might be beneficial for a faster training process, a smaller batch size leads to more accurate estimators[54]. Furthermore, accuracy gains are expected to be achieved after fewer number of epochs when training with smaller mini-batches compared to larger batch-sizes.

- *Learning rate decay* is applied to assure the convergence of the optimization problem to a minimum. This might not always be the case if the learning rate is kept constant [55]. Several approaches can be used in order to adjust the learning rate throughout the training process.

$$\alpha = \frac{1}{1 + d_r e_p} \alpha_0 \tag{2.44}$$

with the *decay rate* d_r being the hyperparameter for this process. It is multiplied by the number of the current epoch e_p. The adapted and previous learning rate are represented by α and α_0, respectively. Alternative approaches are to adjust the learning rate exponentially

$$\alpha = 0.95^{e_p} \alpha_0 \tag{2.45}$$

or introducing another hyperparameter k

$$\alpha = \frac{k}{\sqrt{e_p}} \alpha_0 \tag{2.46}$$

The learning rate can also be modified in relation to the minibatch size t

$$\alpha = \frac{k}{\sqrt{t}}\alpha_0 \tag{2.47}$$

Finally, the learning rate can also be adjusted in a discrete way in relation to the number of epochs or even manually.

Further hyperparameters such as the number of hidden units and the number of layers are defined by the network architecture, applied to a particular use-case.

3. Proposed Solution for IWT Semantic Segmentation

The following section describes the process of training a semantic segmentation model for an IWT environment. As a first approach, the segmentation task in this work will lay the focus on RGB images only.

3.1 Used Model

The DeeplabV3 model is an extension of the Deeplab algorithm and is suggested by [56]. The algorithm is developed as a Deep Convolutional Neural Network (DCNN) for the task of semantic segmentation. An approach called *atrous convolution* is used to control the resolution of the image without any additional learning parameters [56], [57].
Referring to the word "trous", which means "holes" in French, atrous convolution can be thought of as a convolution with a filter that consists of holes (values equal to zero) between the parameters. By adjusting the size of the holes in the filter (the so called *rate*), its FoV can be adapted. Standard convolution is equivalent to atrous convolution with rate $= 1$.
DeeplabV3 is built on the basis of ResNet (residual network) [58], a powerful approach for object detection and semantic segmentation. By proposing residual learning, the authors of [58] introduced the valuable approach of adding so-called *skip-connections* to a NN. Up to a certain threshold, adding more layers a NN does indeed improve the training accuracy. However, beyond this threshold, deeper networks (networks that consist of an increased number of hidden layers) exhibit a worse performance due to

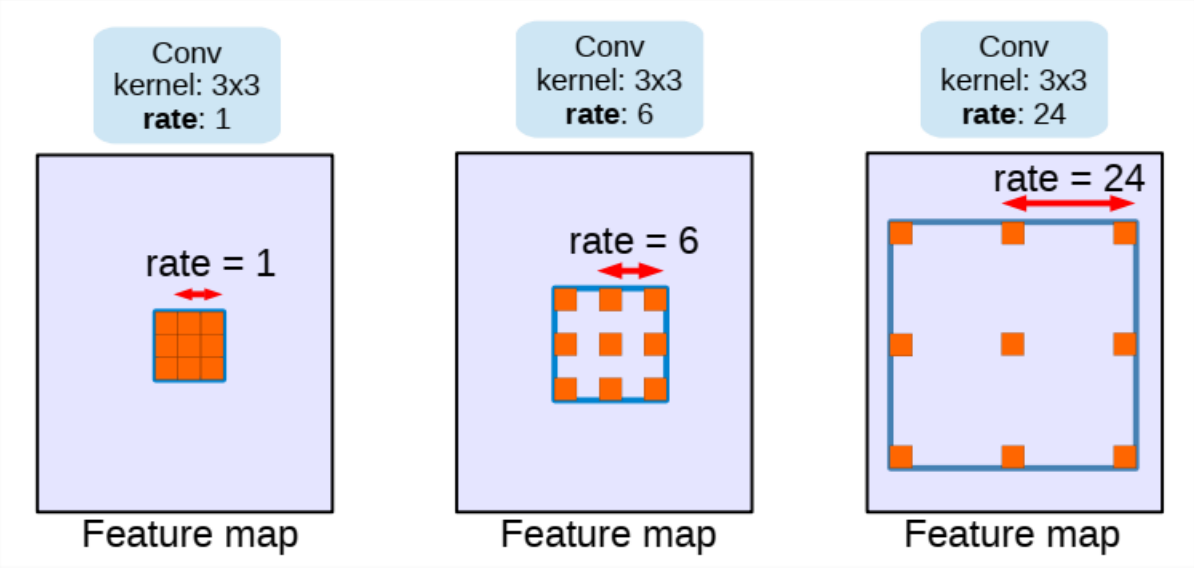


Figure 3.1: Atrous convolution as used in the DeeplabV3 model. The field of view (FoV) can be adjusted by the *rate*, which defines the spaces between the filter parameters [57].

the problem of vanishing or exploding gradients [59]. For the sake of simplicity, an activation function $g(z) = a$ and a biases $\mathbf{B}^{[l]} = 0$ are assumed. In very deep NNs, the vector of all estimated outputs of the network $\hat{\mathbf{Y}}$ can be described by

$$\hat{\mathbf{Y}} = \mathbf{W}^{[l]}\mathbf{W}^{[l-1]}\mathbf{W}^{[l-2]}\dots\mathbf{W}^{[3]}\mathbf{W}^{[2]}\mathbf{W}^{[1]}\mathbf{X} \tag{3.1}$$

Each matrix $\mathbf{W}^{[l]}$ is represented by its p_W parameters, as

$$\mathbf{W}^{[l]} = \begin{bmatrix} p_W & 0 \\ 0 & p_W \end{bmatrix} \tag{3.2}$$

in each of the K layers.
By neglecting the dimensions of the last parameter $\mathbf{W}^{[1]}$, the estimated output can be rewritten as

$$\hat{\mathbf{Y}} = \mathbf{W}^{[1]} \begin{bmatrix} p_W & 0 \\ 0 & p_W \end{bmatrix}^{[K-1]} \mathbf{X} \tag{3.3}$$

and therefore

$$\hat{\mathbf{Y}} = p_W^{[K-1]}\mathbf{X}. \tag{3.4}$$

The number of layers K tend to be very large in deep NNs. This leads to the effect, that the Eq. (3.4) will exhibit very large values if $p_W > 1$ (“exploding gradients”) or very

small values if $p_W < 1$ ("vanishing gradients"). As the activation values increase or decrease exponentially as a function of K, the computation of the gradients for gradient descent is affected in a similar way. In particular, vanishing gradients slow down training substantially [59]. The problem of vanishing / exploding gradients is addressed in the implementation of ResNet [58] by the application of skip-connections. As depicted

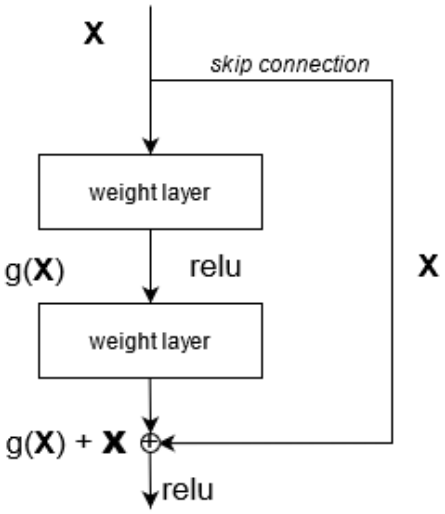


Figure 3.2: Residual block. The skip connection counteracts vanishing / exploding gradients.

in Fig. 3.2, the activation function of the subsequent layer $a^{[l+2]} = g(\mathbf{Z}^{[l+2]})$ is adapted to $a^{[l+2]} = g(\mathbf{Z}^{[l+2]} + \mathbf{A}^{[l]})$. To account for dimensions mismatch, $\mathbf{A}^{[l]}$ may be adjusted, so that $(\mathbf{W}^{[l]}\mathbf{A}^{[l]}) \in \mathbb{R}^{(n \times n)}$, whereas $\mathbf{A}^{[l+2]} \in \mathbb{R}^{(n \times n)}$.

ResNet offers the option to use a pre-trained model. The authors of [56] use a ResNet50 as well as a ResNet101 for comparison reasons. Both models have been pre-trained on ImageNet [60], that serves as a benchmark in object detection, image classification and single-object localization. The dataset consists of hundreds of object categories and millions of images and is therefore a good possibility for models that shall be pre-trained for a broad use-case.

Atrous Spatial Pyramid Pooling (ASPP) [56] addresses the issue, that the repeating combination of convolutional and max-pooling layers significantly reduces the resolution of the image. So called "deconvolutional" layers may counteract this effect, but with increased computational effort. ASPP applies several atrous convolutions with increasing rate subsequently to each other [56]. By doing so, the FoV can be arbitrarily enlarged while the resolution of the output feature map is kept stable.

All modules added by the authors in addition to ResNet include batch normalization. The authors of [56] demonstrate the superior performance of DeeplabV3, when compared to other state-of-the art NNs. The learning rate is set dynamically to $(1 - \frac{iter}{max\_iter})^{0.9}$, where *iter* and *max_iter* represent the current and total number of iterations, respectively. The initial learning rate is 0.007. After the first 30.000 iterations, the batch normalization is frozen and another 30.000 iterations are carried out with a smaller base learning rate $\alpha = 0.001$. Throughout the training process, the training data is augmented by randomly scaling the images and random horizontal flipping. The authors highlight the performance of the model: its mean intersection over union (IoU) amounts to 85.7 % [56]. Further details about the mean IoU metric for segmentation problems can be found in Section 3.4.

3.2 Training Procedure

As discussed in Section 2.1, a number of datasets have been generated for machine learning applications in the automotive domain. However, for the application to IWT, the variety of datasets is extremely limited. Indeed, the datasets collected by [31] and [32] provide high resolution RGB images collected from a nordic lake environment and various inland waterway scenes, respectively. However, the aim of both datasets is limited to water segmentation only: The ground-truth segmentation masks contain only the two classes “water” and “not water”.

MaSTr1325

The only training dataset suitable for the application to this work is therefore the MaSTr1325 dataset [33]. It was collected in the coastal water of Koper (Slowenia), to enforce the development of small USVs, operating in coastal waters. From the 50 hours of data, collected over a span of two years, 1325 images were hand-picked while ensuring to cover various weather conditions and a great variety of semantic scenes. The images are labelled pixel-wise to the categories “water”, “sky”, “obstacle” and “void”. To address the problem of labelling uncertainty at the edges between neighbouring classes, these pixels are labelled as “uncertain”. Being collected in a coastal, marine environment, the MaSTr1325 does not perfectly match the aimed application to

IWT semantic scene understanding. Still, the dataset covers a variety of weather conditions and different scenes and is labelled extremely carefully. It is therefore chosen for the training process in the application under discussion, even though the encountered scenes may differ slightly.
The authors of [33] augment their data by horizontal flipping and central rotation by 5° to 15° of the images. Furthermore, following the approach of [61], a small set of descriptive images is selected from the target domain and used for colour transfer. Indeed, data augmentation is a widely used approach to generate more representative data, which reduces the risk of overfitting. Also for the application to IWT, data augmentation is a valuable approach. However, as inland vessels barely tilt along their main axis (rolling), central rotation of the images might be less beneficial, compared to the application in coastal waters. Fig. 3.3 depicts the different modes of data augmen-

Figure 3.3: The dataset generated by [33] was augmented by seven colour augmentations and four rotations augmentations (last row).

tation. Together with the unaugmented images, a total of 53000 images are available for training [33]. With the dataset in place, the authors train, among other models, a DeepLabV2 [56] on a NVidia GTX1080 Ti GPU. For single image segmentation, the

DeepLab architecture achieves a mean IoU of up to 97.49 %, which is the best result compared to other networks in use. Therefore, the DeepLab architecture has been chosen for the IWT segmentation solution proposed in this work.
To first approach the segmentation problem, a model was trained on the unaugmented MaSTr1325 dataset for only 20 epochs. Even though the results were evolving in a promising way, it was clear that the trained model could not serve the task of thorough inland waterway scene understanding. While objects, water and sky could be recognized with reasonable accuracy, no distinction could be made between objects, bridges, vessels and other objects belonging to IWT infrastructures. The need for a dataset with a larger class variety arose.

3.3 Generation of the BerlinIWT Dataset

As described previously, the availability of training data for IWT applications is extremely limited. Even though the MaSTr1325 dataset is a valuable assistance for the desired use-case, a need arises for finer labelled IWT application-based datasets. For the purpose of further data acquisition, a measurement campaign was carried out, that shall be described in detail in this section.
Even though the number of training examples inside the dataset is limited when being compared to other datasets used in many ML problems, it is, to best of my knowledge, the first dataset for multi-class semantic segmentation for IWT applications.

3.3.1 Measurement Campaign

The main goal of the measurement campaign is data acquisition for the generation of an appropriate dataset. Bearing in mind the further scope of the application, the detected infrastructure shall be integrated by a SLAM algorithm into a global reference system. Therefore, also the GNSS position needs to be recorded.
Fig. 3.4 depicts the acquisition platform: a *Quicksilver 675 Pilothouse* motorboat. The *Aurora* is a survey boat, owned by the Department *Nautical Systems* of the Institute of Communication and Navigation, German Aerospace Center (DLR) Neustrelitz. With its 7 m length over all, 2.6 m maximum beam and less than 3 t of fully loaded mass, the boat can easily be moved to the desired location easily using a trailer. With the

112 kW outboard engine (and an additional bow thruster), the boat is easy to steer. A maximum draught of only 0.9 m allows to navigate even shallow waters. Carrying a wide range of valuable on-board equipment, such as an electronic chart plotter, depth sounder, GNSS positioning, very high frequency (VHF) radio and radar sensor, the boat is suitable for both, inland waterway and coastal navigation. These on-board sensors are part of the standard equipment provided and powered by the outboard engine. For the operation of all measuring hardware, 230 V power is provided by an additional alternator. The Aurora can host up to seven persons, even though space is rather limited, making it undesirable to exhaust this maximum capacity. The measurement campaign was carried out with six persons on board, which is sufficient for the different tasks on board, such as safe navigation, hardware surveillance and minuting the measurement process.

3.3.1.1 Hardware in Use

For the acquisition of optical sensor data, a LiDAR device and a stereo camera are used. The LiDAR employed is a Sick MRS6000. It emits pulsed laser beams at a wavelength of 870 nm every 10.5 ns. More information about the working principle of LiDAR devices can be found in Chapter 4. The device has a horizontal FoV of 120 ° with an angular resolution of 0.13 °. The vertical FoV amounts to 15 ° with an angular resolution of 0.625 ° , which is recorded within four different horizontal planes. The laser scanner is designated for outdoor usage, with a nominal working range of 0.5 to 200 m. However, reaching the maximum range requires perfectly reflecting objects, that can hardly be found in a typical outdoor scene. From the specifications of the LiDAR device manual, a maximum working range of 90 m seems more realistic [62]. Depending on the conditions, some laser beams might be reflected earlier than others, which could occur in case of rain, fog or transparent surfaces in the propagation path. For appropriate environment perception under these conditions, four different echoes can be evaluated by the device. More detailed information about laser devices can be found in Chapter 4.

The RGB images are recorded by a Stereolabs ZED 2i stereo camera. The baseline between both lenses amounts to 12 cm, allowing depth estimation within distances of up to 20 m. With a horizontal FoV of 120 °, the camera is rather wide-angled. Besides the visual perception sensors, the Aurora is equipped with multiple GNSS antennas

Figure 3.4: The motorboat *Aurora* was used for the measurement campaign. The optical sensors, IMU and one GNSS antenna were mounted on the bow of the boat and two further GNSS antennas on the roof.

and receivers, a couple of MEMS IMUs, a barometer and a magnetometer.
For the recording and storing of both, LiDAR and camera data, an Ubuntu 20.04 laptop equipped with the robotic operating system (ROS) is used. This middleware provides, among other functionalities, low-effort sensor synchronization, data recording and replying. Furthermore, ROS allows to view the recorded data in real-time and is equipped with straight-forward methods for playing back the recorded files.
Precise localisation and absolute attitude determination are provided by three different GNSS antennas and an IMU. All position data are saved locally on a designed computational platform, which is referred to as the PNT (position, navigation and timing) unit. For synchronization, all positioning sensors and the laptop recording visual sensor data are connected to the PNT unit.
The set-up of all sensors in use is depicted in Fig. 3.4. The combination of the optical sensors is mounted between the rails at the bow of the boat as depicted in Fig. 3.5. The

Figure 3.5: The sensor combination at the bow of the boat consists of a GNSS antenna, an IMU, the LiDAR sensor and a stereo camera (top-down).

stereo camera is installed below the LiDAR sensor and pointing in the exact same direction, which reduces the effort for calibration of the optical sensors. Special attention is required by the camera as the FoV can be obstructed by the vessel's anchor and its bracket. Therefore, the whole combination is tilted slightly upwards. This also eases the recognition of bridges with the LiDAR sensor, as its low vertical FoV of only 15 ° makes it difficult to capture the shapes of the river bank and the structures of higher bridges at the same time. Due to the focus on bridge detection, priority was given here to the recording of structures, which are higher above the water level. On top of the optical sensors, the external IMU and one GNSS antenna are installed. Two further GNSS antennas are placed on the equipment carrier on the cabin of the boat. The water-proof cables of all sensors are led through a side-hatch of the cabin and connected to the power plugs and processing hardware. Among others, the PNT unit (Fig. 3.6) consists of three GNSS receivers, that are each connected to a distinct antenna. Position data is saved with the *Precise Point Positioning* (PPP) standard and therefore centimetre-level accuracy in the *receiver independent exchange* (RINEX) format. The recording of positioning data can be monitored by a laptop connected to the PNT unit. In this case, the same laptop is used for recording optical sensor data, accessing the PNT unit and writing the minutes. For later traceability, the passage of other vessels, bridges and waterway locks are logged in addition to the start and end of the recording as well as the sensor behaviour.

Figure 3.6: The PNT unit is stored safely on a mounting structure inside the boat. It carries all GNSS receivers and is used for synchronization of all sensors.

3.3.1.2 Covered Trajectory

As described previously, the main focus of the measurement campaign is the acquisition of optical sensor data for the application to IWT. Special priority is given to the bridge detection in a first stage and to bridge surveying as one of the subsequent steps. As stated by the project [63], first developments of the self-updating inland waterway chart focus on the *Spree-Oder-Wasserstraße* (SOW), that connects a big crossing of inland waterways in Berlin further eastbound. Indeed, DLR contributes to the development of a digital test bed for cross-linked and highly automated IWT on the SOW [64]. It is therefore desirable to carry out this measurement campaign in an environment that is developing towards a future test bed. Furthermore, in the urban area of Berlin a considerable amount of bridges are located for traversing various channels and the SOW. The environment is therefore suitable for the acquisition of optical sensor data of a large variety of different bridges. Finally, the company operating Berlin's ports, *Behala* [65], as a valuable partner in the project consortium, supported the measurement campaign by providing their crane for trailer - water transfer of the boat as well as allocating a berth in Berlin Westhafen.

Berlin's urban waterways are governed by a number of restrictions with respect to the accessibility of pleasure crafts. Namely, the SOW in the center of Berlin between 10:30 a.m. and 7 p.m. can only by navigated by skippers having an additional VHF certificate. The measurement campaign was carried out on two different days. After

Figure 3.7: First part of the measurement campaign on day 1. After leaving the port, the boat proceeded westbound through the *Westhafenkanal* and then southbound to the SOW. The return point is the beginning of a restricted area at the bridge *Lessingbrücke* (km 12.01 of the SOW).

the boat had arrived on the trailer on the first day, it was transferred into the water by a crane. Most of the equipment has been already in place, requiring only minor additional checks on site. A first part of the trajectory was carried out from the port *Westhafen* westbound through the canals *Westhafenkanal*, *Schleusenkanal* and the SOW until km 12.01. The bridge *Lessingbrücke* indicates the border to the restricted area. This trajectory can be viewed in Fig. 3.7. The weather conditions were cloudy with several rain showers. Including the way back to the port this first trajectory on the first day amounts to 14 km. Additional measurements were performed by passing one single bridge several times while adjusting the mounting angle of the combination of optical

sensors. The circles around a small island next to the port are visible in Fig. 3.7. The goal of this manoeuvre was to find the optimal mounting angle, at which the LiDAR sensor can capture the largest part of the scene regardless of its limited vertical FoV.

In order to enter the restricted area in the city center, another section of the measure-

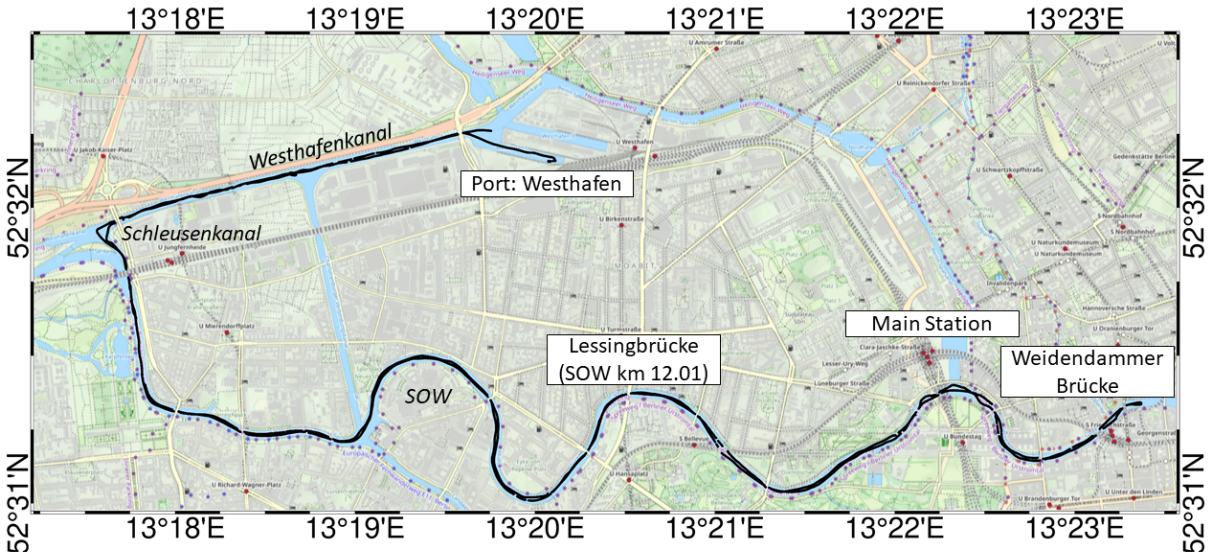


Figure 3.8: Second part of the measurement campaign on day1. The SOW was sailed until the bridge *Weidendammer Brücke* inside Berlin's restricted urban waterways.

ment campaign was carried out from 6:30 p.m. until 8:30 p.m. After leaving the port, the boat navigated on the SOW beyond the bridge *Lessingbrücke* and passed by the restricted waterway next to the main station. The boat turned around and began heading back to the port in vicinity to the bridge *Weidendammer Brücke*. The associated trajectory is depicted in Fig. 3.8. Lighting conditions during this second part of the first measurement day were governed by dusk in sunny and cloudy, but mostly dry conditions.

The trajectory carried out on the second day of the measurement campaign is depicted in Fig. 3.9. With the restriction to pass km 17.8 of the SOW by 10:30 a.m., the boat left the port at 07:45 a.m. on the second day of the measurement campaign and proceeded on the SOW beyond km 12.01. The waterway lock *Mühlendammschleuse* at km 17.8 of the SOW indicates the end of the restricted area and was passed by 10:00 a.m. From this location, the boat sailed the SOW further to the south-east and entered the narrow

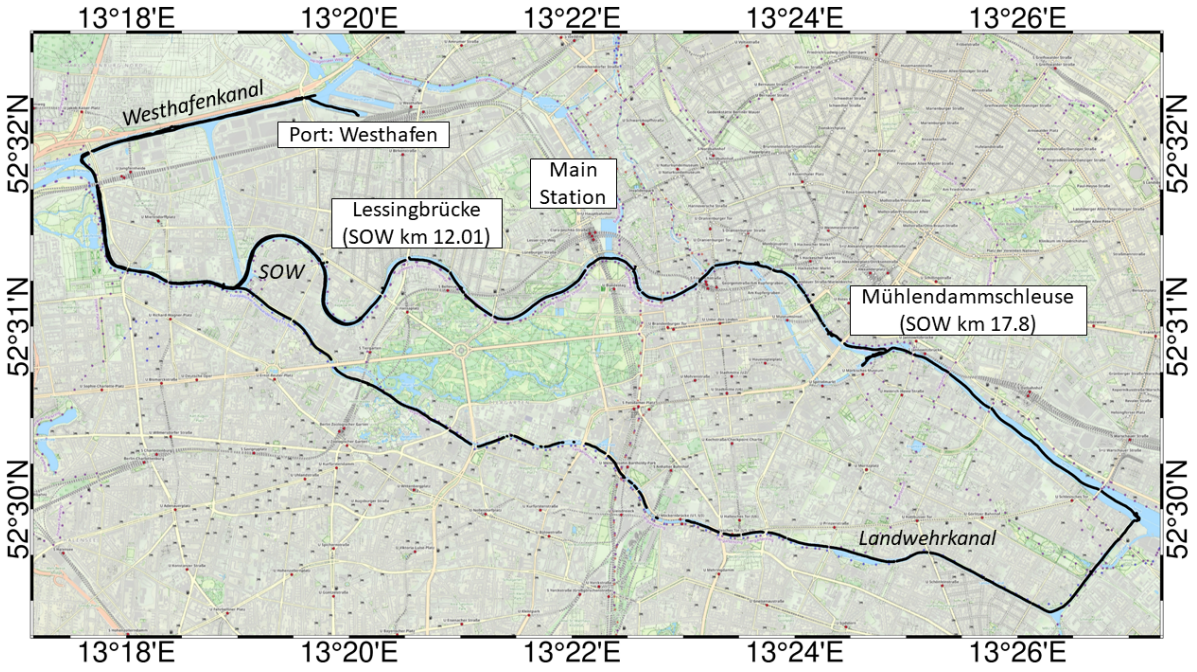


Figure 3.9: Second day of the measurement campaign: After leaving the port westbound, the boat proceeded through the SOW eastbound through the waterway lock *Mühlendammschleuse* and back through the narrow channel *Landwehrkanal*.

channel *Landwehrkanal*. The entire trajectory carried out on the second day is visible in Fig. 3.9. Even though the first parts of data were recorded well after sunrise, the low elevation of the sun and its reflection in the water abnormally compromised the correct recording of RGB images. With the sun rising up, this rare problem vanished completely and bright sunlight as well as fair winds provided perfect conditions: the scene is fully illuminated and small waves inside the water prevent the bridge from being reflected on the water surface. Looking at the trajectory depicted in Fig. 3.9, small outages in the position solution can be observed when following the *Landwehrkanal*. This is the result of small hardware outages in the associated time period. Regardless of these small issues, the complete measurement campaign was carried out without any problems.

3.3.1.3 Collected Data

With the varying weather conditions on the first and second day, a great variety in the data collection can be achieved with only these two days of measurement campaign. Even though both camera lenses were cleaned in regular intervals during the rain showers on the first day, several images are compromised by raindrops, as in Fig. 3.10. However, as real-world applications might face the exact same problem, also some of these images were chosen for the annotation process.
In addition to the single images, that are used for the generation of the dataset, also stereo images have been recorded by the stereo camera in order to keep the possibility of depth estimation using RGB images only. Depth estimation with stereo images can be performed at a maximum range of 20 m. This limitation is outperformed by the LiDAR sensor. Furthermore, depth estimation on stereo images would not be as accurate as the spatial mapping information provided by LiDAR. During the measurement campaign, the LiDAR device is cleaned regularly during the rainy periods, even though rain drops do not seem to affect the performance of the device. All optical sensor data is recorded with a sampling rate of 10 Hz.
The GNSS position of the vessel can be viewed in Fig. 3.7, 3.8 and Fig. 3.9 for both sections of the first day and the second day, respectively. Most of the time, the GNSS position is acquired with centimetre accuracy using the PPP standard. In case, the PPP standard is not available, the RTK (real-time kinematics) solution is used as a reference.

3.3.2 Annotation

Of the 11 hours of recorded data, the most relevant sections are extracted. Time frames, in which many different bridges or other vessels were passed are considered as relevant sections. Even though, commercial vessels can only operate on wider waterways, such as the SOW, also data from narrow channels (e.g. the *Landwehrkanal*) are used to increase the variety of the dataset. From the file recorded in the ROS format *rosbag*, RGB images for the most relevant time sections of the measurement campaign are extracted. In total, 200 images were chosen from the sections that have been considered as relevant. For the variety in the training dataset, images of different bridges, recorded at

Figure 3.10: Image recorded by the stereo camera, compromised by raindrops. The low quality was chosen for reasonable memory usage during the recording process.

varying distances, are selected. With lesser frequency, also various images of the same bridge under varying conditions (rain, sun) and varying distances, as well as pictures of under-bridge passages are selected. Within the possibilities, data recorded inside waterway-locks is additionally selected for annotation.
It was initially planned to perform the annotation process using the web-interface of the *Computer Vision Annotation Tool (CVAT)* [66]. The advantage of a web-interface based solution is that barely any additional software is needed for the annotators to fulfil their task. Beside the user-friendly handling, CVAT offers the possibility to create distinct projects. Another valuable functionality are the different stages of the annotation process, that, among others, also dedicate a specific stage to quality control of the masks. However, at the time requested, the CVAT server was not available. With the time constraints, the need for a quickly availably alternative solution arose.
Therefore, the annotation process is carried out with the web-interface *MakeSense* [67]. This tool allows annotation for image recognition or object detection. A text-file containing the labels can be loaded directly to the web-interface. The four in-house an-

notators are briefed with respect to their task and the software in use. Depending on the complexity of the image, the annotation of one single image takes between 10 and 20 minutes. In order not to constrain the usage of the dataset by annotating too few classes, a great variety of 14 classes is provided, including the classes

- Sky.
- Water.
- Bridge: all infrastructure, that fully traverses the waterway, including pipes, except power cables.
- Vessel: any kind of boat or ship.
- Floating object: objects, that float inside the water, but are not steered. Berlin's waterways carry a considerable amount of waste bottles or woods, but also water birds.
- Vegetation: Any kind of plants visible in the picture, especially trees and bushes. Such plant covers can be observed frequently along the banks of inland waterways.
- Moving object: any measure of transport, e.g. trains, cars or bicycles, but also any kind of animal. The main goal of this class is to filter out objects that are not part of a permanently installed infrastructure, but rather dedicated to displacement.
- Person. The differentiation to the class moving object might not be strongly necessary, but beneficial for future implementations.
- River bank: where ever the waterway is not directly adjacent to a building, the bank of the river is surveyed as part of the IWT infrastructure.
- Water mark. Signs for mariners are surveyed as part of the IWT infrastructure. Buoys are distinguished from other floating objects.
- Waterway lock: all gates and chamber walls that are part of a waterway lock.
- Other object: permanently installed objects, that do not comply with one of the above mentioned classes. This applies especially to any kind of building.

- Default: image structures, that cannot be recognized.
- Background. Standard class provided by *MakeSense*, contains all pixels not being assigned to any other class. Especially used whenever border lines between objects cannot be distinguished certainly.

Indeed, distinguishing between all of the mentioned classes is not always straight forward. Even though all of the in-house annotators were well briefed with respect to this issue, confusions arose throughout the annotation process. Furthermore, the diversity of the classes is prone to slow down the annotation of complex images. In fact, not all of the 14 classes may even be required for the application under discussion. However, the advantage of larger flexibility justifies the effort of using all of the classes for the annotation.
Once annotated, the annotation files can be derived from the *MakeSense* web-interface in *COCO-JSON* [68] format. With the conversion to the *labelme* format [69], the masks can be saved as *.png* file. To adapt the dataset to the standards of MaSTr1325 [33], the RGB values of the files are normalized to values between 0 and 14. In total, 190 images

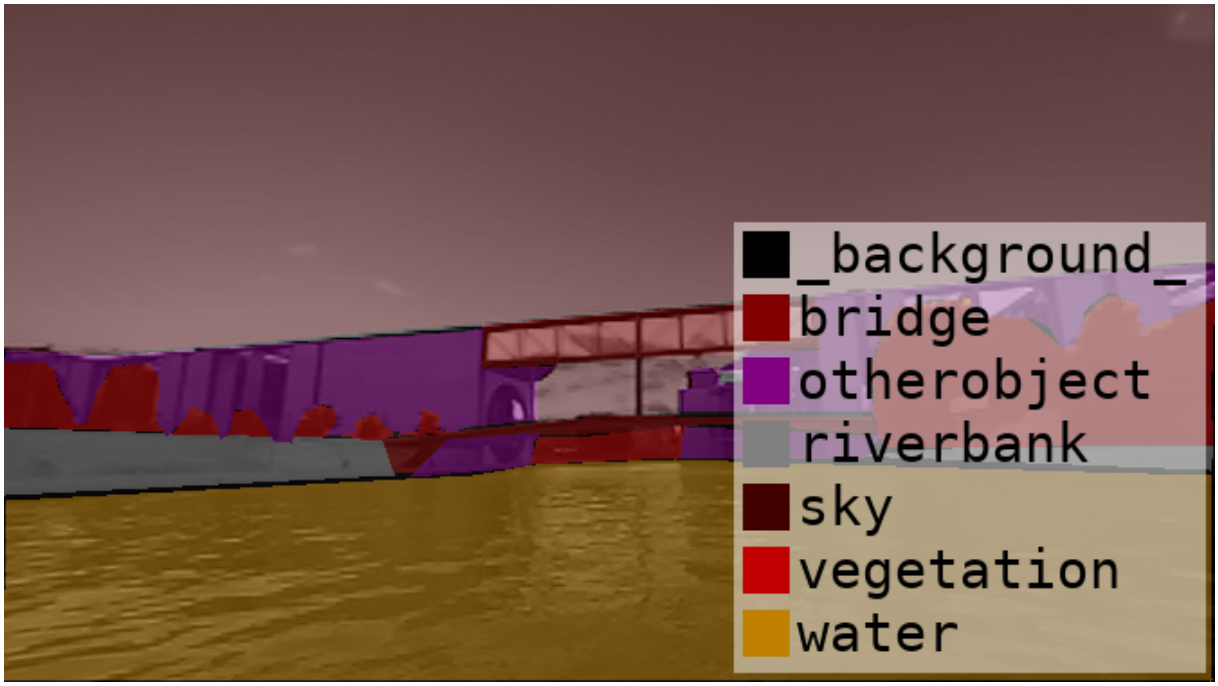


Figure 3.11: Hand-labelled annotation example. Not all 14 classes are present in this picture.

are annotated, of which 29 masks are compromised in a way, that they are not suitable for training. Also within the remaining 171 images, large differences can be observed in the quality of the pixel-level annotations.
In case the diversity of 14 different classes is not needed for a specific training purpose, the generated mask can easily be adapted by merging the RGB values of different classes together. This additional step in pre-processing could be executed at relatively low effort. On the other hand, the sudden need for an additional class would require to restart the labelling process, which would be a large drawback.

3.4 Training Environment

The NN in use is a DeepLabV3, as described in Section 3.1, that is pre-trained on the ImageNet dataset. To allow training on a *Graphical Processing Unit (GPU)*, a *Nvidia Jetson Xavier AGX developer kit* is used. The NN is implemented in pyhton's Tensorflow library 2.9.1, running over a Jupyter Notebook. Even though, the Jetson Xavier AGX developer kit does not offer extraordinary computational resources, the usage of a GPU is far more suitable to the ML-problem, than applying a conventional *Central Processing Unit*, which may not have all functionalities required by machine-learning libraries.
To find a reasonable trade-off between resolution of the images and masks processed by the NN and computational time, an image size of 256 pixels is chosen for both, height and width. For faster training, the dataset is divided into mini-batches of 8 examples each. The loss function is optimized with the Adam optimizer, using the default momentum terms of $\beta_1 = 0.9, \beta_2 = 0.99$ and $\epsilon = 10^{-8}$. Learning rate decay is not applied. However, a *Learning Rate Scheduler* adapts the learning rate if the evolution of a certain evaluation metric does not exhibit significant changes after a certain number of epochs. This number of epochs is also referred to as *patience*. For the training configuration in this work, the *validation mean IoU* is monitored with a patience of 35 epochs. The IoU is chosen in this case over the accuracy metric, because high accuracy values are very easy to achieve in semantic segmentation problems. Due to the high number of pixels per mask, accuracy values of 90 % may even be achieved after only one epoch of training. However, the metric may neglect small, but significant patterns by solely

comparing the overall pixels classification in true and predicted mask. This is where the advantages of the mean IoU metric take effect: the mean overlap of both, true and predicted, masks is computed with the respect to the number of classes. In general, the IoU can be thought of as a relation between the area of intersection of detected patterns and the area of union [70].

$$IoU = \frac{\hat{y}^{(i)} \cap y^{(i)}}{\hat{y}^{(i)} \cup y^{(i)}} \tag{3.5}$$

To achieve best possible results, the choice of hyperparameters and the training environment has been carefully adapted throughout several different configurations.

3.4.1 Results of a Five-class Dataset

The BerlinIWT dataset is explicitly developed for semantic segmentation of infrastructure in an inland waterway scenario. With its 14 classes, the variety of classified infrastructure is large, making the dataset applicable for a number of different use-cases. However, due to time constraints, the dataset consists only of 171 examples, that can be used for training and evaluating a CNN. Indeed, not all of the initially defined 14 classes are necessary for the application in discussion here: the overall goal of this work is the detection of bridges along inland waterways from optical sensor data. The large variety of classes is therefore merged down to only 5 different classes. Beside the class *bridge*, the classes *sea, sky, object* and *other* are defined. The class *object* contains any kind of object, while the class *other* contains all pixels that were assigned to the classes *background* and *default* as explained in Section 3.3.2. This offers the possibility to enlarge the dataset with the MaSTr1325 dataset [33], which contains the last four classes *sea, sky, object* and *other*. Furthermore, the lower number of different classes is expected to counteract the flaw of the low number of training examples. The strategy of pre-defining more classes than possibly needed and merging them accordingly to the use-case is a straight-forward way for efficient use of recorded datasets. Generally, this also works flawlessly for this particular application. The only drawback is introduced by re-labelling pixels of the class *water-mark* as *object*. As depicted in Fig. 3.12, water-marks are often attached to bridges. As a result of re-labelling the associated class as *object*, certain areas of the bridge are labelled as *object*, that should be labelled

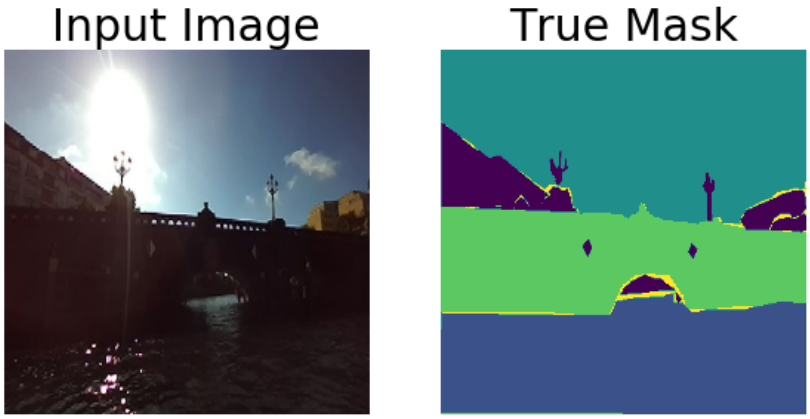


Figure 3.12: Input image x and true mask y as processed by the network. The example is part of the BerlinIWT dataset. The re-labelling process results in certain areas of the bridge being labelled as object.

as *bridge*. Still, re-labelling the class *water-mark* as *bridge* is a worse alternative to this problem, as many floating buoys are labelled as *water-mark*.
Even though the environment of the examples in both, the MaSTr1325 and the BerlinIWT dataset is very different, both datasets are combined for the training process. Fig 3.13 shows an example from the MaSTr1325 dataset [33]. While the class *bridge* clearly remains underrepresented, the training process strongly benefits from the increased number of examples accounting for *sky - water - object* segmentation.
For training, validation (development) and testing, 70 %, 20 % and 10 % of the dataset

Figure 3.13: The exmaples of the MaSTr1325 dataset are used additionally for training.

are used, respectively. Most examples in use show completely different bridges. How-

ever, rarely also various pictures of the same bridge occur in the dataset. In some extremely seldom cases, even the conditions under which the picture of the bridge was taken (daylight, weather, distance to bridge) are similar. To show repeating information in different (training and validation) sets, it is tried to ensure, that examples depicting the same bridge are in the same, either training or validation set. Table 3.1

	MaSTr1325	**BerlinIWT**	**Total**	**of which BerlinIWT**
whole dataset	1325	171	1496	11 %
training set	931	116	1047	11 %
validation set	263	37	300	12 %
test set	131	18	149	12 %

Table 3.1: Number of examples in training, validation and test set. The percentage of BerlinIWT examples is sufficiently constant among the sets.

shows the number of examples retrieved from either the MaSTr1325 or the BerlinIWT dataset, used for training, validation and testing. For performance comparison, it is important that the class imbalance (underrepresented BerlinIWT examples) remains approximately constant among training, validation and test set. The combination of both datasets leads to a total number of examples of 1469, with 11 % of BerlinIWT examples. This trend is kept throughout all of the other subsets, that are split up as described previously.

The 1047 examples in the training set are augmented with random values for brightness, contrast and saturation. Furthermore, the images are flipped horizontally, resulting in a total number of (augmented and unaugmented) 5235 examples used for training. Fig. 3.14 depicts the result of this data augmentation process for one example from the training set that belongs to the BerlinIWT dataset. The images shown in Fig. 3.14 are not (yet) resized for the processing by the NN. The mask depicted in the bottom right corner of the figure is flipped horizontally in order to match the horizontally flipped image. Training the DeepLabV3 for 100 epochs on the augmented dataset requires approximately 20 hours on the NVidia Jetson Xavier AGX. The learning rate $\alpha = 0.001$ remains constant throughout the whole training process, as no learning rate decay is applied and the validation mean IoU does not settle for the threshold of 35

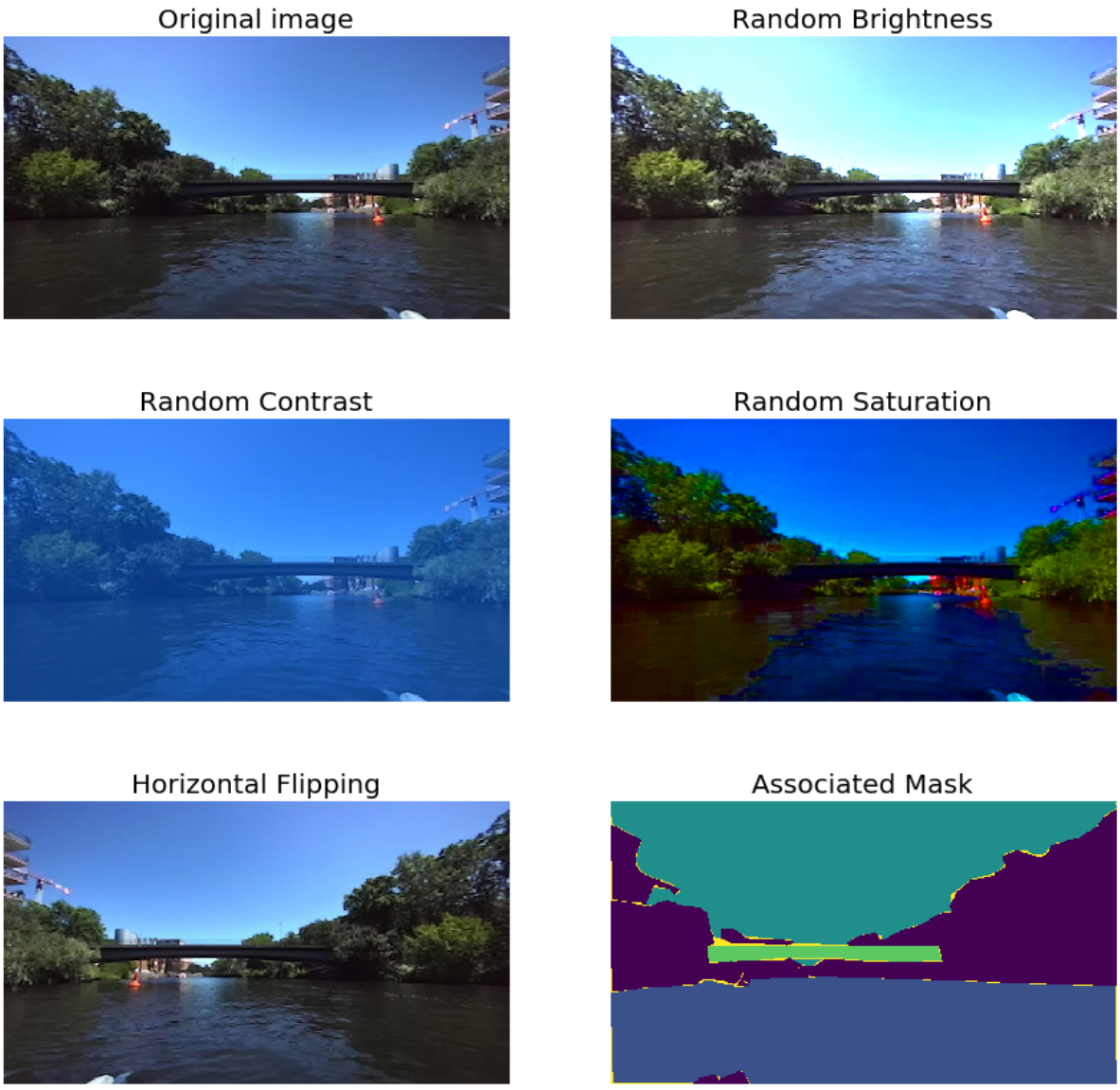


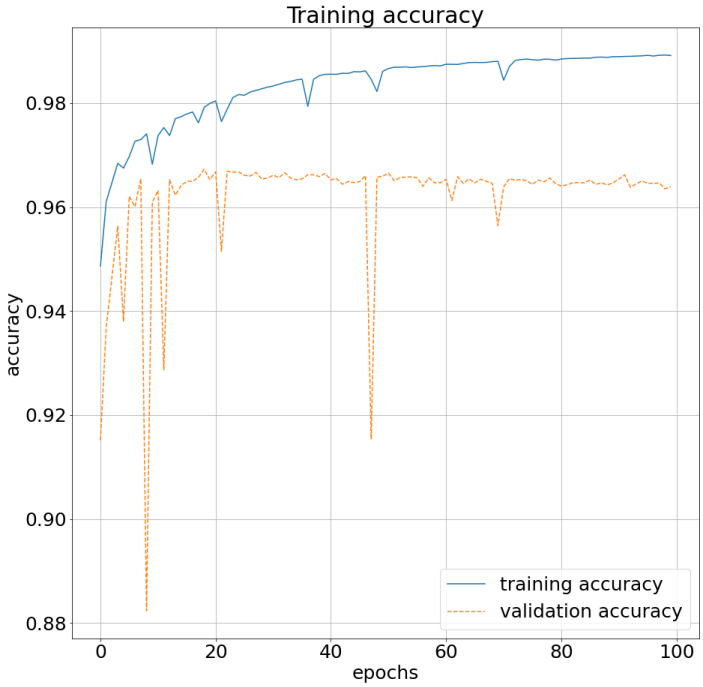
Figure 3.14: All examples in the training set are augmented by horizontal flipping as well as random brightness, contrast and saturation.

epochs. Table 3.2 provides an overview of the hyperparameter settings used in this training scenario. Fig 3.15a depicts the accuracy achieved during the training pro-

hyperparamter	**value**
number of epochs	100
α	0.001
β_1	0.99
β_2	0.9
ϵ	10^{-8}

Table 3.2: Overview of hyperparamter settings.

cess over the 100 epochs performed. As described initially, good accuracy values are easy to achieve in semantic segmentation problems, because even moderate prediction masks are recognized as good estimates. Small, but significant deviations between true and predicted mask can hardly be viewed by looking at the accuracy values only. Obviously, it is expected that the validation accuracy is significantly smaller than the training accuracy. Still, evaluating the performance of the network on data that are not used by the optimization algorithm for parameter adjustments is an important metric for hyperparameter tuning and also assists in overfitting prevention. Furthermore, it is noted that the validation accuracy exhibits uneven behaviour than the training accuracy. This effect is caused by the different size of the two sets. In the smaller validation set, even small parameter changes might affect the accuracy value significantly. In the larger training set, small parameter changes have a less severe effects on the accuracy. After the initial increase of both accuracy values, a slight decrease in the validation accuracy can be observed. This behaviour indicates first, slight overfitting effects: the parameters of the model are well adapted for the examples inside the training set. Indeed, an increasing training accuracy can also be observed from the plot. However, the model slightly lacks sufficient generalization capabilities and therefore does not perform as well on data, that has not been used for parameter adaptation. For practical use-cases, it is extremely important, that the model is able to generalize the inputs well, as real-world applications need to deal with unseen data. After training for 100 epochs, a training accuracy value of 98.91 % and a validation accuracy of 96.39 % can be



(a) Accuracy achieved during training on the training and validation set.

(b) Mean IoU achieved during the training process on training and validation set.

Figure 3.15: Accuracy and mean IoU on training and validation set.

observed. The accuracy is also evaluated on the test set, where it amounts to 96.81 %. The test accuracy being slightly higher than the validation accuracy is a coincidence, that indicates room for improvement with respect to the hyperparameter tuning. However, due to the large computational time of 20 hours for only one run of 100 epochs, hyperparameter adjustment has already been a time-intense process until this point. Further hyperparamter tuning is, given the limited time and computational resources, beyond the scope of this work.

As mentioned previously, the accuracy might not be the most meaningful metric for the evaluation of a semantic segmentation model. Fig. 3.15b depicts the evolution of the mean IoU assessed on the training and validation set throughout the 100 epochs of training. Both, the training and validation mean IoU start at a value of 40 %, when training is first initialized. The largest evolution of the metric can be observed until epoch 40, where the mean IoU for training and validation reaches 90 %. Again, the bumpy behaviour of the validation mean IoU with respect to the corresponding value assessed on the training set can be noticed. During the first 40 epochs, the validation mean IoU can even be observed surpassing the training mean IoU. This effect can happen by coincidence, especially when the training and validation sets suffer from redundant information. However, thanks to the carefully executed pre-processing steps of the data, training and validation mean IoU tend to be levelled at a decent value above 90 %. In contrast to the accuracy plots, no overfitting behaviour can be observed from the mean IoU values: both of the curves tend to increase until the last of the 100 training epochs. Best performance could be achieved by a model, that is optimized until the very last epoch before the overfitting starts. It would be therefore interesting to perform longer training and observe the overfitting behaviour. However, such experimentation will be considered as future work due to the limited computational resources. The training mean IoU after 100 epochs amounts to 94.36 % and the validation mean IoU to 92.52 %. Again, the associated value computed for the test set is with 93.12 % slightly higher than the validation mean IoU indicating the need for further hyperparameter optimization.

Fig 3.16 shows how the prediction of the model evolves over the training epochs. For the assessment of the model evolution, a validation image from the BerlinIWT dataset is chosen. The notable adverse lighting conditions of the input image make this exam-

(a) Prediction after epoch 1.

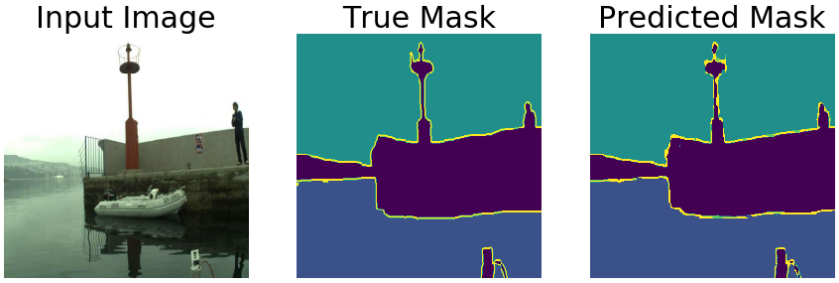
(b) Prediction after epoch 7.

(c) Prediction after epoch 72.

Figure 3.16: Evolution of the masks predicted by the model. The shown mask is part of the validation set.

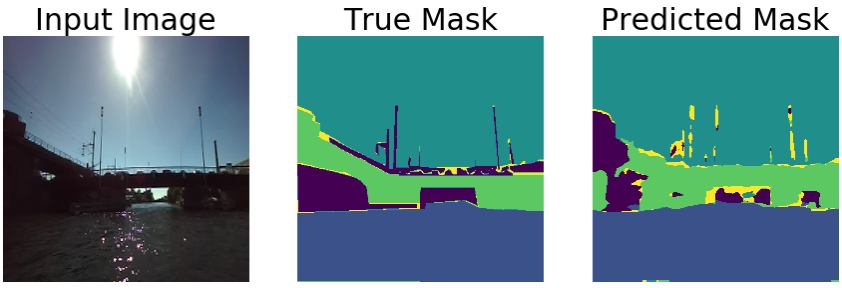
ple a challenging one. The true mask depicts a rather complicated scene. Furthermore, the pixel-wise annotation could be more precise, but is still suitable for the purpose. A considerable amount of pixels are labelled as uncertain and the fine-grained bridge structures are not labelled in every detail. Still, the model exhibits decent performance already in the estimate after epoch 7. Even after epoch 1, significant progress can be noticed when looking at the predicted mask. Looking at Fig 3.15b and Fig. 3.16, a decent model evolution over the first epochs is visible. This behaviour is expected to be caused by the rather high learning rate of 0.001. However, it is a challenging task to determine the correct point and time where the learning rate should be decreased in order to close the small, but notable performance gap remaining at a mean IoU of 94.36 %. Still, a validation mean IoU of 92.52 % is an excellent result, especially bearing in mind the limitations of the available training data and computational resources.
It has been discussed, that the mean IoU is an extremely relevant metric for segmentation problems. Therefore, the validation mean IoU is monitored for possible adjustments of the learning rate. Generally, it would also be possible to use the mean IoU as cost function and directly optimize on the most desirable metric. However, this prevents the model from learning the actual classification-related segmentation. The mask prediction would be adapted nicely towards decent mean IoU values, whereas the classification performance would suffer severely. For a segmentation problem containing more than two classes, this leads to a model that exhibits decent ability in shape detection, but performs poorly when it comes to classifying the detected object. This behaviour is not feasible. In addition, the flatter derivation of the mean IoU with respect to the cross-entropy loss function is prone to slow down the training process as the optimizer would converge slower towards a minimum. Therefore, the standard cross-entropy function is used as loss function. In Fig. 3.17, a selection of masks from the validation set is depicted with the associated predicted mask by the fully trained model. The two upper examples belong to the MaSTr1325 dataset. It can be noticed, that even subtle structures, such as the crane structures in Fig. 3.17a, are predicted by the model with sufficient accuracy. Bigger shapes, such as quay walls and coastlines, as in Fig. 3.17b, are recognized without any difficulties. For the examples from the BerlinIWT dataset, the lack of training data visibly affects the performance in the associated examples. Still, the overall bridge structures are recognized in Fig. 3.17c and

(a) Example from the MaSTr1325 dataset.



(b) Example from the MaSTr1325 dataset.

(c) Example from the BerlinIWT dataset.



(d) Example from the BerlinIWT dataset.

Figure 3.17: Selected masks from the validation dataset with predictions of the fully trained model.

3.17d. Uncertainties arise mostly concerning the superstructures of the bridge, such as the lamp post on the bridge in Fig. 3.17c. These structures are labelled as part of the bridge and classified as object by the model. However, this uncertainty is expected not to compromise the performance of the algorithm with respect to a real-world use-case. For the application to inland waterway charts, the clearing heights and widths below the bridge are of importance, whereas possible superstructures could be neglected completely. Furthermore, any kind of bridge superstructure could arguably also be classified as object, instead of being treated as part of the bridge. The model faces further challenges when looking at the example depicted in Fig. 3.17d. In most of the training examples considered, the depicted bridges cross the waterway transversely to the waterway direction. In the example depicted in Fig. 3.17d, an additional bridge is visible in the left part of the image, following the waterway in longitudinal direction. This structure is not detected perfectly.
The room for improvements on the hyperparameter tuning is clearly visible from the prediction masks depicted in Fig. 3.17. Very challenging examples from the validation set may even exhibit worse performance. In rare cases, the predicted mask might even show partially failed segmentation operations.
The trained model was tested successfully on an independent test set, where the test mean IoU amounts to 93.12 %. Despite further adaptations that could be made for further improvements on the results, this value for the mean IoU indicates an incredibly good performance. This is especially true when considering the small set of application-related training data and the limitations of computational resources.

3.4.2 Results for a Dual Class Dataset: Bridge Detection

For the intended use-case of bridge recognition, even a two-class segmentation problem might be sufficient. To simplify the training process, the remaining classes from the previous scenario *sky, water, object* and *other* are all merged into one single class. The result is a training dataset with only two classes: *bridge* and *other*. As the MaSTr1325 does not contain any relevant information for bridge detection, it will not be used in this example. Furthermore, all images not depicting a bridge are removed from the BerlinIWT dataset. The remaining 164 examples are split to 70 %/20 %/10 % for training, validation and testing, respectively. The training set therefore contains 116 examples, the

validation set 30 and 18 examples are kept for testing. Fig. 3.18 shows an example

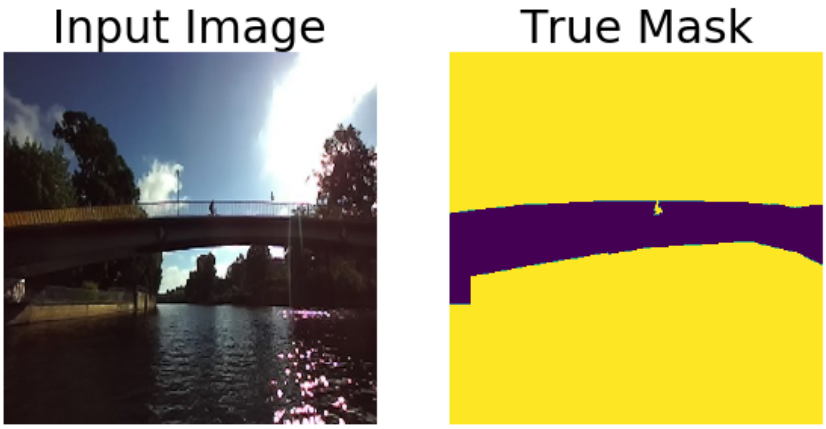


Figure 3.18: Example from the BerlinIWT dataset. Pixels are classified as *bridge* and *other*.

retrieved from the validation set with the associated 2-class-mask. It can be noticed, that pedestrians crossing the bridge are well annotated as *person*, as described in Section 3.3. Using only two classes for the segmentation task, the class *person* is merged into the class *other*, causing a small inconsistency in the shape of the bridge. The effect observed here was already described in Section 3.4.1, were watermarks attached to the bridge compromised the correct modelling of the shape. Even though slight uncertainties in the generation of the true mask can be noticed when various pre-defined classes are merged into one, the overall shape of the objects, namely bridges, is modelled correctly. The result of the segmentation problem is therefore expected not to be compromised significantly by these labelling inaccuracies.

The 116 training examples are augmented with horizontal flipping and random values for brightness, saturation and contrast resulting in a total of 580 training images. An example is depicted in Fig 3.19. In the example depicted, the camera points towards the sun, causing low illumination of the rest of the scene. The challenging conditions are even aggravated by the lower illumination of the example augmented with random brightness values in the upper right corner. The mask depicted in the lower right corner of Fig. 3.19 is associated with the horizontally flipped image. The fine structures of the steel bridge depicted are not labelled in the detail of every pixel. This might be acceptable as not all fine structures of the bridge may be recognized at this distance. Still, this example is another indication for the varying quality of the pixel-wise anno-

Figure 3.19: Data augmentation is applied by horizontal flipping, as well as random colours for brightness, saturation and contrast.

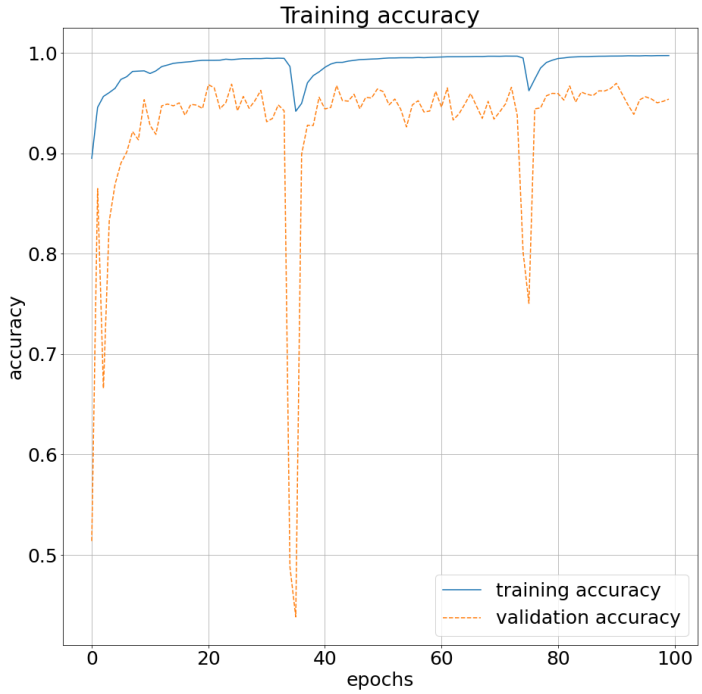
tations inside the dataset.
Considering the aforementioned hardware , the training on the described dataset for 100 epochs requires only 3 hours. Generally, this would allow training for many more epochs while still achieving reasonable computational time. In order to be comparable to the training scenario described in Section 3.4.1, the training process is limited to 100 epochs also in this scenario. In contrast to the 5 class scenario, the loss function is adapted to the training goal. As described, in Section 3.4.1, the mean IoU is again used as evaluation metric for the segmentation problem. Using it as a loss function might not be appropriate for multi-class segmentation. However, for two-class segmentation the recognition of the shapes in the masks might be sufficient. Therefore, the cross-entropy loss function is replaced by the mean IoU in this training scenario. The learning rate is initially chosen to $\alpha = 0.001$ and adapted by a learning rate scheduler monitoring the validation mean IoU with a patience of 35 epochs. With the validation mean IoU not settling for 35 epochs throughout the training process, the learning rate remains constant for all 100 training epochs. The relatively high learning rate turned out to demonstrate best performance among all hyperparameter settings that were tried. The training examples are sub-divided into batches of 8 examples per batch. The Adam-optimization algorithm is applied with the default parameters $\beta_1 = 0.99, \beta_2 = 0.9$ and $\epsilon = 10^{-8}$. The model in use is a DeeplabV3, pre-trained on the ImageNet dataset. An overview of the hyperparameter configuration is provided by Table 3.3.
Fig. 3.20a shows the accuracy evaluated on the training and validation set over all 100

hyperparamter	**value**
number of epochs	100
α	0.001
β_1	0.99
β_2	0.9
ϵ	10^{-8}

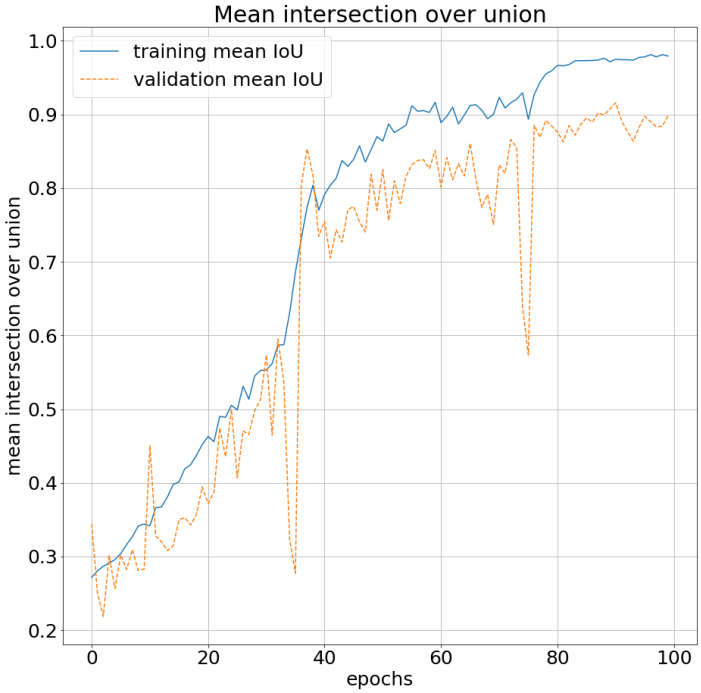
Table 3.3: Overview of hyperparamter settings.

epochs. As described previously, good accuracy values are relatively easy to achieve

in segmentation problems, because small, but significant deviations in the predicted mask barely influence this metric. It can be noticed, that the validation accuracy values develop with a higher noise level than the training accuracy. This is caused by the lower number of examples inside the validation set when compared to the training set. As the performance can be evaluated on only 30 examples, even small developments of the model can cause remarkable changes of the validation accuracy. A similar effect is expected to introduce the spikes in the training and validation accuracy at epoch 35 and 75. With the limited dataset, even small parameter adjustments can lead to significant accuracy changes. The larger training set is less affected by this development and shows smaller spikes than the smaller validation set. This is another indication that the spikes are caused by the limited dataset. As the learning rate influences the magnitude of parameter adjustments, a smaller learning rate could possibly reduce the spike effects. Another option would be to decrease the patience of the learning rate scheduler and adapt the learning rate during the training process. Finding the correct value for the patience of the learning rate scheduler is a challenging task. If the value is set too low, the learning rate is decreased rapidly causing the parameters adjustments to become extremely low, which in fact prevents relevant developments towards a minimum of the cost function.
Disregarding the spikes, both accuracy values show decent development, even after the convergence around epoch 15. With the validation accuracy remaining constant, no overfitting effects can be observed. After training for 100 epochs, the training accuracy amounts to 99.72 % and the validation accuracy to 95.39 %. On the independent test set, an accuracy of 96.17 % is achieved. With the test accuracy slightly outperforming the validation accuracy, the need for further hyperparameter adjustments is notable. Further hyperparameter optimization will be considered as future research. While the accuracy indicates good performance of the model, the limited significance of the metric needs to be kept in mind.
Fig. 3.20b depicts the mean IoU evaluated on the training and validation set, respectively. Again, the noisier behaviour of the validation mean IoU with respect to the training mean IoU is notable. At times, the validation mean IoU even surpasses the associated training value by chance. It is interesting to note the significant spikes in the validation mean IoU at epochs 35 and 75, that are also visible for the accuracy metric



(a) Accuracy over 100 epochs. The introduced spikes might be the effect of the small dataset.



(b) Mean IoU over 100 epochs, converging comparably late around epoch 80.

Figure 3.20: Accuracy and mean IoU on training and validation set.

in Fig. 3.20a. The combination of both metrics indeed indicates poor development of the parameters around these two epochs. However, the overall development of training and validation accuracy shows that the model parameters evolve in a favourable way. The metrics show first convergence behaviour around epoch 60, but experience another improvement at epoch 80. With the spikes additionally influencing the validation mean IoU, it is clear that this metric has not settled for 35 epochs at any point in the training process. It is therefore plausible, that the learning rate is not decreased by the learning rate scheduler. After a total of 100 epochs, the training mean IoU amounts to 97.93 % and the associated validation value to 89.86 %. The test mean IoU of 91.50 % indicates room for further hyperparameter improvements. Even though these further improvements are beyond the scope of this work, the mean IoU values for both, validation and testing indicate surprisingly good performance of the model. Mean IoU values around 90 % are extremely good results, especially when considering the limitations of the dataset and the computational resources.

Fig. 3.21 depicts the evolution of a predicted mask in different epochs. The lighting conditions in the example depicted here can be considered challenging. Furthermore, the scene contains a variety of objects, further complicating the recognition of the bridge. The depicted example is part of the validation set. It is remarkable, that already after epoch 1 (Fig. 3.21a), significant segmentation progress is visible. At epoch 12, first basic bridge structures are visible from the predicted mask. Until epoch 67, the shape of the bridge is modelled more accurately with respect to the fine structures. By this epoch, the validation mean IoU amounts to approximately 85 % and indicates first convergence trends. Still, remarkable uncertainties in the finer structures of the bridge keep persisting. Apart from that, a small amount of false positive pixels labelled as bridge can be recognized in the left part of the predicted mask in Fig. 3.21c.

Fig. 3.22 shows a selection of examples with their true masks and the prediction made by the fully trained model. All depicted examples are part of the validation dataset. It can be noted, that the overall structure of the bridge is predicted with an acceptable certainty by the model. Following the validation mean IoU of 89.86 %, the predicted mask shows significant overlap with the true mask. Even though 100 % of validation mean IoU is practically impossible to achieve, the predicted masks illustrate that even this very decent value of almost 90 % still indicates a remarkable performance gap. The

(a) Prediction after epoch 1.

(b) Prediction after epoch 12.

(c) Prediction after epoch 67.

Figure 3.21: Evolution of the masks predicted by the model. The shown mask is part of the validation set.

(a) Example from the BerlinIWT dataset.

(b) Example from the BerlinIWT dataset.

(c) Example from the BerlinIWT dataset.

Figure 3.22: Selected masks from the validation dataset with predictions of the fully trained model.

remaining 10 % towards a truly perfect predicted mask could only be achieved by significant improvements in the finer structures of the recognized shapes. The labelling uncertainties in the superstructures of the bridge depicted in Fig. 3.22b are hardly relevant for a real-world application in the IWT domain. Nevertheless, in the masks of Fig. 3.22a and 3.22c, the uncertainties also affect the estimate of the delicate underbridge passages, which is indeed prone to compromise an application that seeks to provide navigation assistance, especially for safety-critical uses. These problems may be addressed by further hyperparameter optimization and longer training.

3.4.3 Comparative Results

An overview over all training scenarios and the results presented is given in Table 3.4. The accuracy and mean IoU evaluated on the training, validation and test set are given in percent. The values listed under *5 class scenario* are the results from the training scenario presented in Section 3.4.1, where the model has been trained using a classical cross-entropy loss function on a combination of the MaSTr1325 and the BerlinIWT dataset, that consists of five different classes. *Dual class scenario* indicates the training scenario presented in Section 3.4.2. In this scenario, the model has been trained on a the BerlinIWT dataset consisting of only two different classes with the mean IoU as loss function.

Segmentation problems can easily achieve good accuracy values. As the mean IoU

	Accuracy [%]			mean IoU [%]		
	Training	Validation	Testing	Training	Validation	Testing
5 class scenario	98.91	96.39	96.81	94.36	92.52	93.12
Dual class scenario	99.72	95.39	96.17	97.93	89.86	91.50

Table 3.4: Overview of all metrics for both training scenarios.

is a more suitable metric for the evaluation of segmentation tasks, it is hardly surprising, that all of the accuracy values are higher than the corresponding mean IoU values. Furthermore, it is expected that the training accuracy and mean IoU surpasses the associated mean IoU in all of the scenarios. Both evaluated metrics exhibit better

performance on the test set than on the validation set. This is true for both training scenarios in discussion and shows that further hyperparameter optimization should be applied to both evaluated scenarios. However, detailed hyperparameter optimization is beyond the scope of this work due to limited availability of computational resources and time.

Another interesting development can be viewed when comparing the performance of both training scenarios directly to each other. The dual class scenario achieved slightly higher training accuracy, but lower validation accuracy. The test accuracy in both scenarios is comparable. The five class scenario might be more challenging for training than the two class scenario, as five different classes may be more difficult to distinguish than two. A relevant note is, however, that the dataset for bridge detection is significantly smaller (nine times less), which hinders the learning process. With regard to the testing accuracy these two effects seem to cancel out each other.

The value of the training mean IoU is remarkably higher for the two class scenario than the one for the five class scenario. This might be an effect caused by the mean IoU used as loss function for the two class scenario. However, looking at the mean IoU for validation and testing, the five class dataset with classical cross-entropy loss function shows better performance. This indicates that the larger training dataset offers better conditions than the adapted loss function. The effect does not seem to be compromised by higher number of classes.

Overall the differences in the evaluated metrics are very small. Both metrics evaluated indicate extremely good performance for both scenarios.

4. LiDAR Assisted Spatial Mapping

The process of generating a precise model of the surrounding by using sensor data is called spatial mapping. It is a crucial technique for the generation of a self-updating inland waterway chart, as the model of the environment needs to be set up and defined clearly before it can be mapped into a global frame. High quality spatial mapping information requires semantic scene understanding, but also precise range information to allow for the localisation of objects in the surrounding. Finally, the detected features need to be tracked over time to develop a model of the larger surrounding.
As described initially, a perception system contains different information levels, such as physical and semantic description and intention prediction [71]. Semantic information can be provided using single RGB images as shown in Chapter 3 at good accuracy with comparably low effort [71]. However, single images do not provide any depth information. Even stereo cameras with a decent baseline offer poor distance estimation in measures of maximum range and accuracy, whereas LiDAR sensors provide more precise range information at higher maximum distances. In general, this sensor type is a decent choice for physical description. However, looking at LiDAR point clouds, semantic scene understanding can quickly become a very complicated task. Indeed, when lacking relevant information for semantic description, LiDAR sensors may be inadequate for this task [71].
To exploit the benefits of both sensors, it is a common practice to combine the accurate distance measurements of LiDAR devices with the semantic description achieved by cameras [11], [71]. Due to its valuable contribution on precise range measurements, outperforming camera and even radar systems [11], the LiDAR sensor is subject of further investigation in this work.

4.1 Related Work

Combining LiDAR point clouds and RGB images for semantic segmentation as well as precise range information is a popular approach in order to exploit the maximum capabilities of both sensors.
The goal of *DeepI2P* [72] is to align LiDAR point clouds and RGB data that have been recorded from different points of view or at different angles. The LiDAR point clouds are projected onto the plane of the RGB image. Points outside the camera's FoV are therefore neglected. For the training process, labelled point cloud data is needed, as it would be available from the KITTI [19] dataset [72]. DeepI2P is a robust method for pixel-wise alignment.
Perception-aware multi-sensor fusion (PMF) is proposed by [73] as a method for sensor fusion for 3D LiDAR semantic segmentation. Similar to DeepI2P, also this algorithm projects each point of the point cloud to the camera coordinates: each 2D pixel is assigned with a distance, x, y, z coordinates and an intensity value. A two-stream network, called *TSNet* [73] is proposed to extract perceptual features separately and thus account for possible poor lighting conditions in RGB images. Based on the sensor characteristics, a perception-aware loss is introduced into the network: LiDAR point clouds are considered for local feature detection as shape estimation with this data leads to comparably poor performance [73]. RGB images yield unprecise edge detection, but good shape estimation.
The authors of [74] present an approach that labels GNSS-registered LiDAR point clouds and street-view images by combining rule-based parsing and learning-based labelling. Point clouds and photometric features are combined. First, rule-based detection is applied for the recognition of large surfaces, such as roads and large buildings. These structures constitute almost 75 % of a point cloud in an urban environment [74]. The remaining point cloud is processed using a learning-based approach. The labels can also be projected back into the 2D surface, resulting in a fully labelled 3D point cloud as well as a 2D image [74].
The authors of [75] propose S3CNet, a sparse CNN, that is able to predict the semantically completed scene from a unified LiDAR point cloud using 2D semantic scene completion. The approach achieves state-of-the-art results on the KITTI [19] bench-

mark.
In [76], a method is proposed, to align stereo camera point clouds and sparse LiDAR point clouds without the need for object labelling in 2D images. The algorithm uses a bird's eye view (BEV) representation of the feature map and projects the LiDAR 3D points onto the image feature map. The method is enabled with multi-task learning and surpasses state-of-the-art on the KITTI [19] benchmark [76].
The authors of [77] propose an approach to line up fully segmented RGB images to LiDAR point clouds. The segmentation task is performed on the 2D image, that offers precise semantic description at lower complexity than a 3D point cloud. The LiDAR point cloud is projected into the image plain for the ease of alignment. Finally, the labelled point cloud is generated mapping the 2D image pixel-wise to the 3D point cloud. As no additional calibration for sensor orientation is performed, the two sensor outputs are considered as aligned. The approach does not require LiDAR point clouds that have been annotated in advance, but only 2D images as training data. The library *LDLS* is available online and can be installed on a machine that is equipped with an NVidia GPU.
Especially in the automotive domain, LiDAR sensors are widely used for object detection and even classification. The following two approaches perform semantic segmentation on LiDAR point clouds only and are mentioned here for the sake of completeness.
The authors of BirdNet [78] propose a method for 3D object detection and classification using a LiDAR point cloud that is encoded as a BEV image. The 3-channel image contains information about the height of the highest point in a predefined cell, the mean intensity of all points in that cell and the density of the points inside the cell. The density information is necessary for normalization purposes, which allows the usage of different LiDAR sensors with different resolutions [78]. The sensor orientation is estimated using a Faster R-CNN [79]. Feature extraction can be improved by using RGB-pre-trained weights [78].
RangeNet++ is an algorithm to perform semantic segmentation for point clouds recorded by a rotating LiDAR sensor [80]. The 3D point clouds are mapped to a 2D representation using spherical coordinates. Semantic segmentation is performed on the 2D representation with a CNN. Transformation back to a 3D representation leads to a three-

dimensional, fully labelled point cloud. As the segmentation process is carried out on a 2D representation, it can be obtained at comparably low computational costs.
The athors of [81] propose an approach to identify docking locations for USV using LiDAR sensors only. The problem of a very sparse point cloud is counteracted by overlapping consecutive scans while accounting for the relative motion of the USV. For identifying the dock location, geometric features are exploited. The algorithm is tested successfully on a small dataset. However, the approach remains limited to docks of certain shape and may be compromised by unforeseen movement of the vessel [81].

4.2 Spatial Mapping Information from LiDAR Sensors

As indicated earlier, LiDAR sensors are gaining increasing interest in the automotive domain, mostly applied to self-driving cars. The LiDAR perception system can be used for object detection, classification, tracking and intention prediction. Especially the physical information provided by such sensors is highly reliable. Even though semantic segmentation in LiDAR point clouds is still an extremely challenging task, the sensor offers valuable abilities for object recognition [11].
In general, LiDAR sensors scan their FoV using at least one laser beam and an associated beam-steering system [11]. Many recent devices operate at 1550 nm wavelength. A higher output power (and range) is possible while still meeting eye-safety requirements. However, at this wavelength, also atmospheric water absorption increases, so wavelengths of 850 to 950 nm in the near-infrared (NIR) area are a common usage [11].
The system of a LiDAR sensor can be sub-divided into a laser-rangefinder and the scanning system. The laser-rangefinder consists of the following parts [11]:

- The laser transmitter illuminates the FoV with a modulated wave.
- The photodetector generates an electronic signal from the reflected photons.
- The optics are used to adjust the emitted laser beam and focus the reflected signal on the photodetector.
- Finally, signal processing electronics are used to estimate the distance to the reflecting surface.

An overview of this principle is illustrated in Fig. 4.1 using the example of a direct-detection laser-rangefinder. Direct-detection laser-rangefinders measure the ToF of a

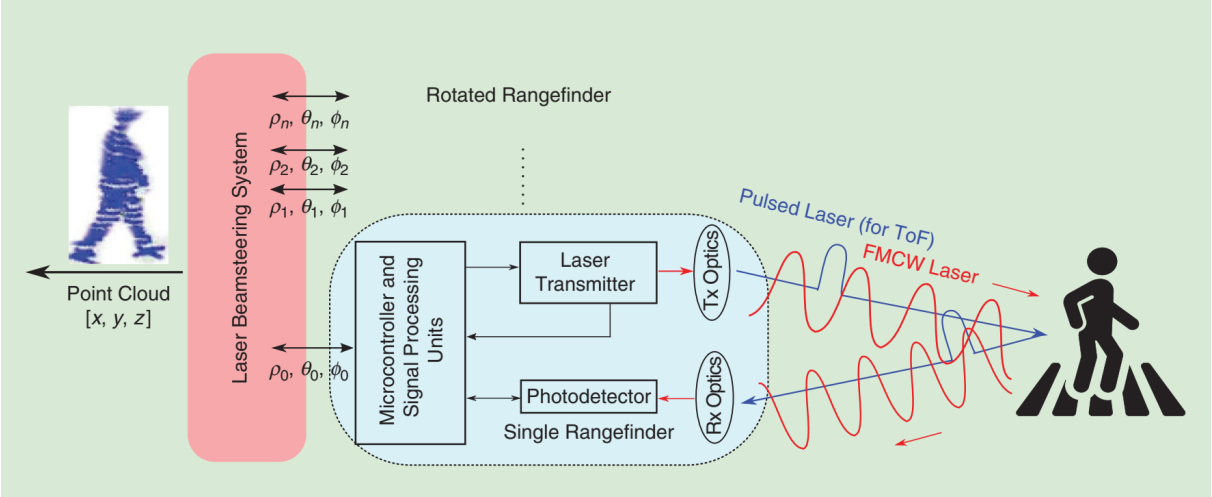


Figure 4.1: Working principle illustrated on a ToF laser example [11].

pulsed laser singal. Coherent-detection laser-rangefinders indirectly measure the distance and velocity from the Doppler effect. For this technique, the laser signal is a frequency-modulated continuous wave (FMCW) [11].
After transmission, the laser is attenuated by the transmission medium. During the reflection at the medium the laser gets diffused, so only a part of it can be captured by the receiving optics. Finally, the received signal is transformed by a photo-detector to an electrical signal. The received power P_r can be modelled as follows [82].

$$P_r = E_p \frac{c\eta A_r}{2r^2} \cdot \beta_r \cdot T_r \tag{4.1}$$

with the distance to target r, received signal power P_r, total energy of a transmitted pulse laser E_p, the speed of light c_r, the area of receiving aperture (Blende) at range r A_r, the overall system efficiency η, the reflectance of the target's surface β_r and the transmission loss through the transmission medium T_r. The reflectance of the targets surface β_r depends on the surface properties and the incident angle. For the simple case of a Lambertian reflection with a reflectivity of $0 \leq \Gamma \leq 1$, it applies $\beta_r = \Gamma/\pi$. This is usually assumed on matte or most rough, non-shiny surfaces, but also a good approximation if the medium is unknown.

A direct-detection laser-rangefinder, using ToF measurements with a pulsed laser signal, as depicted in Fig. 4.1 determines the range r following the equation [11]:

$$r = \frac{1}{2n_r} c_r \triangle t \tag{4.2}$$

with time difference between signal transmission and reception $\triangle t$ and the refraction index $n_r = 1$ used for propagation in air. The structure and signal processing techniques are comparably simple. However, since for ToF LiDARs no modulation is applied, the measured signal of these devices might accidentally interfere with other pulsed lasers. Also, interference from strong sunlight may accidentally be treated as a received signal. The range of this measurement technique is limited due to eye-safety limitations on the transmitted power.

A frequency-modelled signal is emitted constantly over time while a constant reference signal ("local oscillator") is kept. After being reflected, the signal can then be demodulated by mixing the received signal with the carrier signal from the local oscillator. The FMCW constantly illuminates its FoV, so less power needs to be emitted for this purpose. Therefore, eye-safety requirements can be met still by increasing emitted power and FoV.

Figure 4.2 shows the signals for coherent detection. The intermediate frequency IF (in red) is obtained by mixing the received signal (in blue) with the local oscillator (in green). As the laser source and the tracked object may move, the frequency of the received signal is expected to be higher than the reference frequency (Doppler). However, the Doppler frequency shift f_d is assumed to be less than f_{if}, so the following formula applies [83]:

$$f_{if} = \frac{4rB_r}{c_r t_w} = \frac{f_{if}^+ + f_{if}^-}{2}, f_d = \frac{f_{if}^+ + f_{if}^-}{2} \tag{4.3}$$

The velocity is obtained from

$$v = \frac{f_d \lambda}{2} \tag{4.4}$$

with the modulation bandwidth B_r, the waveform period t_w and the wavelength λ. Preferably, the LiDAR device does not contain any moving parts (solid state), is not too bulky and can therefore be fitted easily to an application-related platform.

Mechanical spinning steers the laser beam by moving a mirror or prism, controlled by

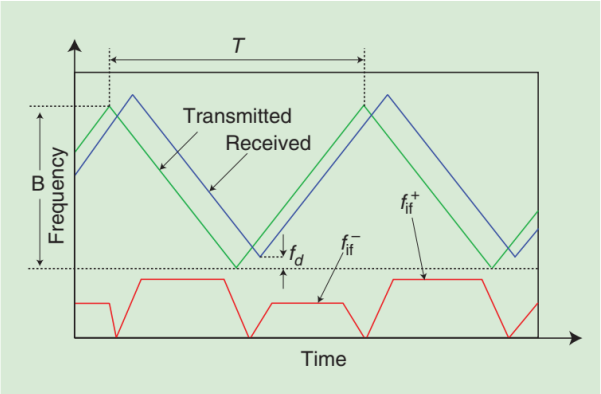


Figure 4.2: Principle of laser FMCW [11]. The intermediate frequency (red line) provides the distance estimate and is generated from the transmitted and received light waves [83].

a motor in order to get a larger FoV. Current LiDAR devices use multiple beams to reduce moving parts. They provide a high signal to noise ratio (SNR) and wide FoV, however the rotating mechanism is bulky and fragile [11].
Microelectromechanical systems microscanning (MEMS) is a tiny mirror embedded directly on the chip, so only very small mirrors need to be rotated if the beam shall be adapted. The *AEye* LiDAR is a MEMS based device, that can adapt its FoV dynamically. Even though this technique still contains moving parts, it can be seen as near solid-state [11].
Flash LiDARs don't contain moving parts, so they are true solid-state. A single laser beam is spread by an optical diffuser to scan the whole area. A 2D array of laser diodes is processed into a 3D point cloud. The pixels measure all ranges simultaneously. The range of this low-cost device is small (100 m), because the whole FoV needs to be scanned with one single laser source, limited in its power emission by eye-safety constraints [11].
Optical phased arrays are solid state devices. Beam adaptation is achieved by a changing the speed of light through optical phase modulators. Even though this technology seems to be promising, it is not yet widely available on the market [11].
As described in Section 3.3.1.1, the LiDAR device used for this work is a *Sick MRS6000*. It emits laser beams following the ToF principle, that are steered by four internally rotating mirrors [62]. Even though the sensor is not true solid state, it is easy to handle. Fig. 4.3 depicts the device with its vertical FoV of 15° and the horizontal FoV of 120°. The working range of 200 m can hardly be reached under real-world condi-

tions as it would require a perfectly reflecting surface in order to achieve a *remission* of 100 % [62]. However, for a remission of 90 %, the scanning range decreases to 90 m, which is important to take in account when using the sensor. In order to view, record

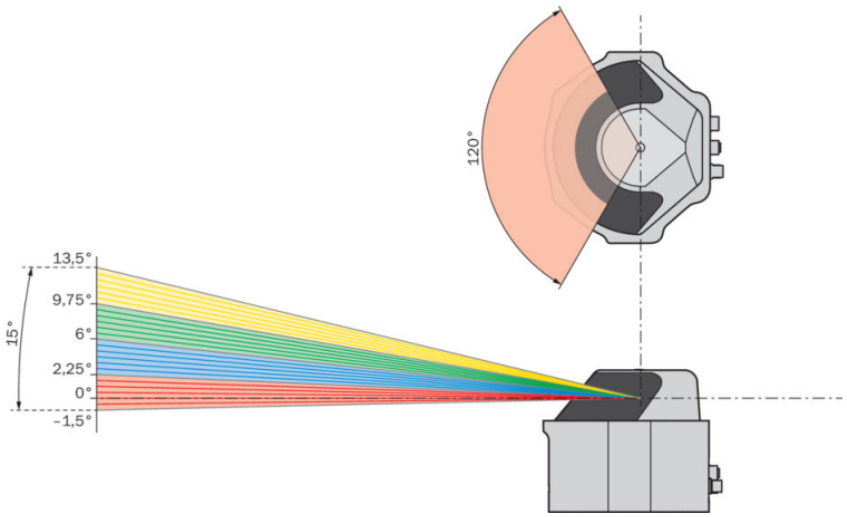


Figure 4.3: The ToF LiDAR Sick MRS6000 provides a horizontal FoV of 120° and a vertical FoV of 15° [62].

and play back point clouds registered by the *Sick MRS6000*, the middleware ROS is used. Fig. 4.4 depicts a point cloud registered by the device during the measurement campaign described in Section 3.3.1. The colors encode the intensity of the reflected points. As can be interpreted by the squared grid coordinate system in the middle of the figure, the view angle has been adapted slightly to demonstrate the full capacity of a 3D point cloud. The point cloud was recorded on the relatively wide waterway *SOW*. Even though the bank of the river on the left-hand side is not visible , the bridge can be distinguished clearly as well as the building block on the right hand side. The accurate resolution of the sensor can be emphasized. However, the limited horizontal FoV is prone to constrain the registration of points either to the entire bridge (as depicted in Fig. 4.4) or to the lower structures of the bridge if the bank of the river shall be depicted additionally. Furthermore, in contrast to the application of LiDAR sensors in the automotive domain, where solid road surfaces generate echoes that can be viewed in the point cloud, the water surface is not visible for the LiDAR sensor. On the one hand, the navigable area can thus be determined relatively straight-forward by the area between two river-banks. Still, this process might be rather complicated if at

least one of the river-banks is located outside the FoV of the device. As pointed out in

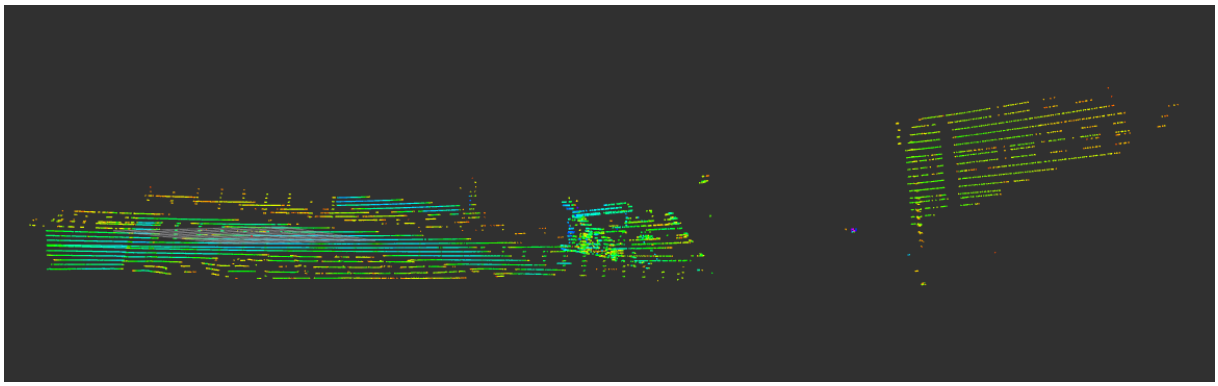


Figure 4.4: Point cloud registered by the *Sick MRS6000*. The colors encode the intensity of the reflected signal.

Section 4.1, the problem of aligning LiDAR point cloud data to the associated image is generally addressed by projecting the point cloud into the image plane [84], [85]. Following the coordinate system depicted in Fig. 4.5, the image plane can be described be the $y_c - z_c$ plane. The direction described by the x_c-axis is the direction of view of both optical sensors. For three-dimensional spatial mapping, it is desirable to orientate the sensors in a way, that the z_c-axis is perpendicular to the local ground plane. However, it was necessary to install the optical sensor combination in a way that it was slightly pointing up into the vertical direction. One important factor for this installation is the limited vertical FoV of the LiDAR device: The sensor had to be installed pointing up in order to capture all the superstructures of the bridges. Another benefit of this orientation is to ensure the thorough scans of the bridge's lower surface while passing under it. This is especially beneficial for the application in discussion, as the clearing height below a bridge is a crucial factor in overall navigation and particularly in inland waterway navigation. Furthermore, the FoV was constrained below the optical sensor combination by an installation of the anchor of the boat, that could not be removed. However, it is important to bear the orientation of the sensors in mind when considering the projections of the point cloud.
The points obtained form the sensor message of the LiDAR are represented in local cartesian coordinates x_c, y_c, z_c. Their projection into spherical coordinates can be ob-

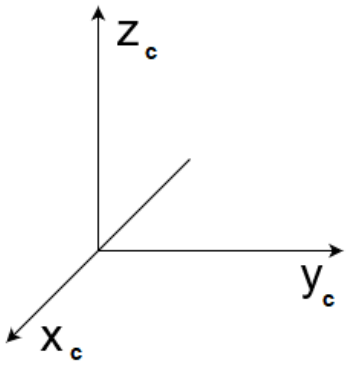


Figure 4.5: Coordinate system used for optical sensor data processing.

tained by [86]:

$$\begin{aligned} d_c &= \sqrt{x_c^2 + y_c^2 + z_c^2} \\ \theta_c &= \arcsin \frac{z_c}{\sqrt{x_c^2 + y_c^2 + z_c^2}} \\ \phi_c &= \arcsin \frac{y_c}{\sqrt{x_c^2 + y_c^2}} \end{aligned} \tag{4.5}$$

The calculation of the euclidean distance d_c is straight-forward. The angels ϕ_c and θ_c are measured into horizontal and vertical direction, respectively. Accounting for the horizontal and vertical FoV, as well as the associated resolution of the sensor leads to accurate visual representation. The result can be viewed in Fig. 4.6. The plots depict the projection of the point cloud shown in Fig. 4.4. It is important to notice that the angle of view might differ between both representations. The colour scale of the diagrams represents the distance and the intensity of the detected points, respectively. In vertical and horizontal direction, a maximum of 24 and 924 points can be detected by the device, respectively [62]. These limitations constitute the axis for height and width of the projection in Fig. 4.6. For orientation, the vertical line in both of the plots represents the middle of the horizontal FoV of the sensor.

The distance representation depicted in the upper part of the figure shows the projected point cloud with the colours encoding euclidean range between the sensor and the detected point. To match the LiDAR device performance under real-world conditions, the range scale is normalized to a maximum of 100 m, which is suitable considering a maximum scanning range of 90 m at 90 % remission [62]. On the left side of the projection, bridge structures are visible at a distance of approximately 40 m. The building structures in the lower right corner are detected at a lower distance of roughly

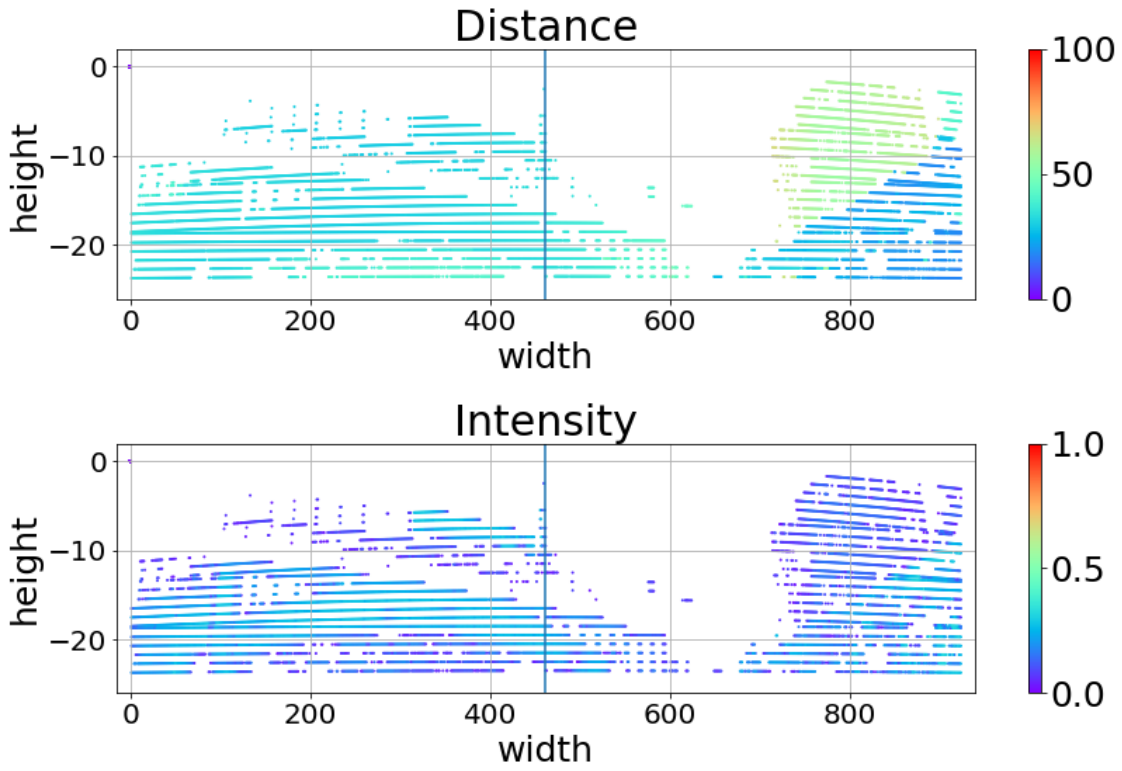


Figure 4.6: Projections of the LiDAR point cloud into the image plane.

10 m. In the upper right corner, the structures of another, larger building are detected with a range of 60 m. The point cloud projection with a colour encoding of the intensity values is visible in the lower part of Fig. 4.6. The intensity values are normalized to a maximum intensity of 1, which would represent perfect reflection of the LiDAR signal. However, the maximum intensity is rarely reached. Slightly increased intensity can be observed from the reflection of some bridge and lower buildings structures. Throughout the whole representation, the detected intensity does not surpass 50 % of the maximum intensity.

4.3 Pixel-wise LiDAR to RGB Alignment

As discussed initially in Section 4.1, three-dimensional point clouds are not as straightforward to label by using semantic segmentation as RGB images. One workaround to

this challenge is to perform semantic segmentation on the 2D RGB images and to align the resulting mask pixel-wise to the associated LiDAR point clouds. A very similar approach, called *Label Diffusion LiDAR Segmentation (LDLS)*, has been implemented by [77].

The authors emphasize the ability of LiDAR sensors to provide spatial mapping with astonishing accuracy. It needs to be admitted, that unstructured 3D point clouds are complex to label and the generation of application-related training datasets can therefore be very complicated. 2D images, however, can easily be annotated by non-experts, as has also been described in Section 3.3 of this work. The challenge can be viewed in numbers when comparing two large datasets such as the KITTI dataset [19] and the MS COCO [68] dataset: MS COCO is an image-only dataset and consists of 200.000 images, whereas the LiDAR-camera combined KITTI is made of less than 8.000 examples [77]. LDLS combines the advantages of both, the LiDAR and 2D camera sensor in order to align the sensor data and finally generate a fully labelled 3D point cloud. The code of the tool is available at the associated GitHub repository [77].

The input required by LDLS is a LiDAR point cloud and an aligned RGB image. This sensor configuration is common for being applied to autonomous vehicles [77] and was also chosen for the measurement campaign described in ection 3.3 of this work. For simplification, only LiDAR points are considered, that lie within the FoV of the camera. Unlike the approach of supervised learning, presented in ection 2.2 of this work, LDLS applies semi-supervised learning, assuming that only a subset of all the available points is labelled. A graph $\mathbf{G}_l$ is constructed, that consists of nodes assigned for 2D images and 3D LiDAR points, respectively. The nodes are connected by two different types of connections: from a 2D pixel to a 3D point and between 3D points. Every 2D pixel and every 3D point is therefore a node within $\mathbf{G}_l$. While the 2D pixels are labelled according to a segmentation mask, all 3D points are unlabelled initially. Considering both sensors as aligned, every 3D point can be projected into 2D pixel coordinates. However, this would introduce high rates of labelling errors, especially around the boundaries defined in the 2D image [77]. This issue is driven by calibration errors between both sensors, as well as the fact that the 2D labelled mask does not take into account the depth information provided by the 3D point cloud. Therefore, a

subgraph is constructed, that combines 2D and 3D information.

$$\mathbf{G}_{l,ij}^{2D\to 3D} = \begin{cases} \lambda_l & \text{if } \mathbf{p}_{l,j} \in P(\mathbf{x}_{c,i}) \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

The set of image pixels $P(\mathbf{x}_{c,i})$ is near the projected 2D location of the LiDAR point $\mathbf{x}_{c,i}$. $\mathbf{p}_{l,j}$ describes the coordinates defined by a 2D pixel. The parameter λ_l is used to control the amount of information flowing from a pixel to its associated LiDAR point. The authors of LDLS use a small and constant value ($\lambda_l = 0.001$) to mitigate sensor calibration errors [77].

For encoding connections between 3D point clouds, an exponentially-weighted nearest neighbours graph is constructed. The set $KNN(\mathbf{x}_{c,i})$ is computed for each point $\mathbf{x}_{c,i}$ within the point cloud according to the euclidean distance.

$$\mathbf{G}_{l,ij}^{3D\to 3D} = \begin{cases} 1 & \text{if } i = j, \text{ else} \\ \exp(-\frac{||\mathbf{x}_{c,i}-\mathbf{x}_{c,j}}{\sigma_l}||_2^2\sigma) & \text{if } \mathbf{x}_{c,j} \in KNN(\mathbf{x}_{c,i}) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

During their experiments, the authors of [77] set $K = 10$ and $\sigma_l = 1$. Finally, the full graph combining all connections is defined by

$$\mathbf{G}_l = \begin{bmatrix} G_l^{3D\to 3D} & G_l^{2D\to 3D} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.8}$$

with the identity matrix $\mathbf{I}$ and $\mathbf{G}_l$ matching the shape of the RGB image. After constructing the graph according to Eq. (4.8), each row is normalized according to

$$\mathbf{G}_{l,ij} \leftarrow \frac{\mathbf{G}_{l,ij}}{\sum_{j'}, \mathbf{G}_{l,ij'}}. \tag{4.9}$$

For all non-zero elements in graph $\mathbf{G}_l$ a connection is indicated, where information in object instance labels should be diffused. Within this process, precise 3D segmentation is performed using both, 2D and 3D data. Including the background instance, a total number of $M_l + 1$ objects instances is assumed. The labels for these instances are stored inside the vector $\mathbf{z}_l^{(m)}$, which then contains one entry for every 3D point and 2D pixel. The computation

$$\mathbf{z}_l^{(m)} \leftarrow \mathbf{G}_l \times \mathbf{z}_l^{(m)} \tag{4.10}$$

is performed iteratively for all $M_l + 1$ instances in order to diffuse labels throughout the graph nodes. If a point $\mathbf{x}_{l,i}$ is unlabelled, but connected to a labelled pixel $\mathbf{p}_{l,j}$, the same label is likely to be applied to the point if $\mathbf{G}_{l,ij}^{2D\to 3D} > 0$. After a sufficient number of iterations is performed for this task (200 is a good benchmark [77]), each LiDAR point is assigned with the most likely label.
Correct label diffusion may fail if the projection or mask boundary is erroneous, which would result in a large number of LiDAR points being projected to the 2D segmentation mask. To account for this error source, an outlier removal step is introduced. The subgraph $\mathbf{G}_l^{(m)}$ of the graph $\mathbf{G}_l^{(3D\to 3D)}$ is defined by considering only LiDAR points labelled as objects. $C_l(\mathbf{G}_l^{(m)})$ is the largest connected component in $\mathbf{G}_l^{(m)}$. The LiDAR points are updated by

$$y_{l,i} \leftarrow \begin{cases} y_{l,i} & \text{if } \mathbf{x}_{l,i} \in C_l(\mathbf{G}_l^{(m_l)}) \\ 0 & \text{otherwise} \end{cases} \forall y_{l,i} \in \{y_l | y_l = m_l\} \tag{4.11}$$

for the current object instance m_l assigned to the pixel y_l of the mask. Finally, the 3D LiDAR point cloud is labelled, in a way that every point is either assigned as background or as one of the class labels [77]).
Indeed, LDLS is an extremely valuable assistance to the problem of labelling 3D point clouds. There is no labelled point cloud data needed, as the labels can be generated from the labelled 2D segmentation mask. These features perfectly fit to the conditions of the development discussed in this work. The code for LDLS is publicly available and requires a python environment and a *NVidia* GPU. These requirements can be met by the usage of the *Jetson Xavier AGX* developer kit. However, without any additionally installed solid state disk, the memory of this hardware in use is not sufficient for the installation of all dependencies required by LDLS. The problem could be easily addressed with the installation of an additional disk. Sadly, this possibility has been infeasible for this work as the availability of the hardware was not matching the time constraints.
The practical implementation of LDLS to the application discussed in this work is an extremely interesting problem which will be subject of future research.

4.4 Optical Flow Estimation

For retrieving precise spatial mapping information, the detected features need to be tracked over time. Therefore, optical flow estimation is another important tool assisting in the generation of a precise, self-updating inland waterway chart.
Since first emerging around the year 2000 [87], three dimensional scene-flow has made constant progress [88]. In general, optical flow describes relative, dense displacements between according pixels through a number of images, mostly consecutive frames of a video [89]. The principle can be used in various fields. The authors of [90] and [91] use optical flow estimation supported by a deep NN for video compression. Optical flow estimation is generally applied to RGB images and can be also used for action recognition, as in [92], [93] as well as for video denoising [94], [95]. Bearing in mind traffic-telematics related applications like the generation of a self-updating inland waterway chart using optical sensor data, the most interesting application appears to be object tracking, as it is implemented by [96], [97].
Indeed, real-time object tracking plays a remarkable role in a number of computer vision challenges, such as motion estimation, activity recognition, but also 3D reconstruction, vehicle navigation and traffic management systems [96]. In general, optical flow estimation is often applied to such surveillance problems. Enabling camera sensors on inland waterway vessels with flow estimation could offer several valuable functionalities for automated map generation. Optical flow estimation would ease the integration of visual odometry. Visual SLAM algorithms could benefit from motion estimation and object tracking assisting in precise relative positioning. Furthermore, the approach could assist in distinguishing moving objects, such as other vessels, from the rest of the surrounding scene while proceeding on inland waterways. Beyond the scope of the application in discussion, this ability could then be exploited for the implementation of systems assisting in collision avoidance or unmanned vessels. However, the placement of the sensor remains as a large difference to the traditional object tracking task. Generally, for object tracking the sensor is placed motionless in vicinity to the infrastructure used by the objects to be tracked. For the generation of a self-updating inland waterway chart, the optical sensors are installed on the vessel in order to recognize the infrastructure while it moves along. With the camera being no longer statically

placed, the relative motion of an arbitrarily chosen object does not describe necessarily the velocity of the vessel. The approach *DeepTAM (Deep Tracking and Mapping)*, presented by [97], applies Deep Learning for dense camera tracking for 3D mapping tasks. Existing learning-based approaches are extended towards full-scale SLAM solutions. The main novelty of DeepTAM is a learned tracking and mapping network, that generalizes well to new datasets [97].
To evaluate the benefit of optical flow estimation for the generation of a self-updating inland waterway chart, two main algorithms are discussed in the following.

4.4.1 Sparse Optical Flow Estimation: The Lucas-Kanade Method

The Lucas-Kanade method was presented as a computationally efficient technique for registering two images together, based on their geometrical features [98]. This concept can also be applied to the registration of subsequently following frames of a video. The method assumes, that the flow is constant in the neighbourhood around the pixel under consideration, which reduces the computational costs [99]. The basic optical flow equations in this neighbourhood are solved by a least squares criterion.
Assuming the frames of the video have been recorded at time t and $t + \delta t$, respectively, the optical flow method calculates the movement between the two frames. In the first image, one pixel $I_1(x_f, y_f, t)$ is considered, that moves the distance $\delta x_f, \delta y_f$ in a certain time δt_f until it arrives in the second frame $I_2(x_f + \delta x_f, y_f + \delta y_f, t + \delta t)$. By using the knowledge, that I_1 and I_2 are two images depicting the same scene within a very short time t, the relation

$$I_1(x_f, y_f, t) = I_2(x_f + \delta x_f, y_f + \delta y_f, t + \delta t) \tag{4.12}$$

can be adopted [98]. The translations tend to be small, as $\delta x_f, \delta y_f, \delta t$ are not too large. The similarity of the two functions can be obtained from the first-order Taylor series in Eq. (4.12). In general, the Taylor series for a value a is defined by

$$\sum_{n_t=0}^{\infty} \frac{f^{n_t}(a)}{n_t!} = f(a) + \frac{f^{'}(a)}{1!}(x-a) + \frac{f^{''}(a)}{2!}(x-a)^2 + \dots \tag{4.13}$$

The local approximations of Eq. (4.12) can be derived as differentials by

$$I(x_f, y_f, t) = I(x_f, y_f, t) + \frac{\partial I}{\partial x_f} dx_f + \frac{\partial I}{\partial y_f} dy_f + \frac{\partial I}{\partial t} dt \tag{4.14}$$

, which also applies to $I(x_f + \delta x_f, y_f + \delta y_f, t + \delta t)$. The relation between both images can be interpreted as velocity with the components $v_x = \frac{dx_f}{dt}$ and $v_y = \frac{dy_f}{dt}$. The equation can therefore be transformed to the *continuity equation*

$$\frac{\partial I}{\partial x_f} v_x + \frac{\partial I}{\partial y_f} v_y + \frac{\partial I}{\partial t} = 0 \tag{4.15}$$

with the image's differentials $I_{\{x_f, y_f, t\}} = \frac{\partial I}{\partial \{x_f, y_f, t\}}$. It indicates the conservation of the spatial intensity mapped to the velocity of the pixels (x_f, y_f) in the image at time t. The equation

$$I_{x,f} V_x + I_{y,f} V_y + I_t = 0 \tag{4.16}$$

describes the movement, which can be rewritten as

$$\nabla \vec{\mathbf{I}}^{\top} \cdot \vec{\mathbf{V}} = -I_t \tag{4.17}$$

with the spatial gradient of intensity $\nabla \vec{\mathbf{I}} = (I_{x,f} I_{y,f})$ and the image velocity (also called optical flow) $\vec{\mathbf{V}} = (V_{x,f}, V_{y,f})$ of the pixel (x_f, y_f) at time t. As this equation with two unknowns cannot be solved directly, another set of equations is needed.
With the assumption made by the Lucas-Kanade method, that the displacement between both images is small and approximately constant in the neighbourhood of the pixel p_f, it can also be assumed that the optical flow is equal for all pixels within a certain window centred around p_f.

$$\begin{aligned} I_{x,f}(q_1) V_{x,f} + I_{y,f}(q_1) V_{y,f} &= -I_t(q_1) \\ I_{x,f}(q_2) V_{x,f} + I_{y,f}(q_2) V_{y,f} &= -I_t(q_2) \\ &\vdots \\ I_{x,f}(q_n) V_{x,f} + I_{y,f}(q_n) V_{y,f} &= -I_t(q_n) \end{aligned} \tag{4.18}$$

with partial derivatives $I_{\{x_f, y_f, t_f\}}(q_n)$ of image I in position (x_f, y_f) at time t and pixel (or point) q_i. With the matrices

$$\mathbf{A}_f = \begin{bmatrix} I_{x,f}(q_1) & I_{y,f}(q_1) \\ I_{x,f}(q_2) & I_{y,f}(q_2) \\ \vdots & \\ I_{x,f}(q_n) & I_{y,f}(q_n) \end{bmatrix}; \vec{\mathbf{v}} = \begin{bmatrix} V_{x,f} \\ V_{y,f} \end{bmatrix}; \mathbf{b}_f = \begin{bmatrix} -I_{t,f}(q_1) \\ -I_{t,f}(q_2) \\ \vdots \\ -I_{t,f}(q_n) \end{bmatrix} \tag{4.19}$$

Eq. (4.18) can be written as

$$\mathbf{A}_f \vec{\mathbf{v}} = \mathbf{b}_f \tag{4.20}$$

, which can be solved to obtain the optical flow $\vec{\mathbf{v}}$:

$$\begin{bmatrix} V_{x,f} \\ V_{y,f} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} I_{x,f}^2(q_i)^2 & \sum_{i=1}^{n} I_{x,f} I_{y,f}(q_i)^2 \\ \sum_{i=1}^{n} I_{x,f} I_{y,f}(q_i)^2 & \sum_{i=1}^{n} I_{y,f}^2(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_{i=1}^{n} I_{x,f} I_t(q_i)^2 \\ -\sum_{i=1}^{n} I_{y,f} I_t(q_i)^2 \end{bmatrix}. \tag{4.21}$$

In order to asses whether the method would be suitable to assist in visual odometry and may be placed as part of a visual SLAM algorithm, the Lucas-Kanade method is implemented on a small subset of the images gained throughout the measurement campaign described in Section 3.3.1. Fig. 4.7 depicts the results of the implementation.

Figure 4.7: The Lucas-Kanade algorithm is implemented on the RGB images gained during the measurement campaign. The dots and lines indicate significant geometrical features recognized and tracked by the algorithm.

The sampling rate of the frames is 10 Hz. For a higher sampling rate, the difference between two consecutive frames (x_f, y_f) is smaller than for lower sampling rates, which is prone to ease the estimation of the optical flow between these frames. The dots and

lines in the picture (Fig. 4.7) represent the position of a point q_i that is chosen as a significant feature by the algorithm and therefore tracked for the optical flow estimation. Best results are achieved when the boat moves in a straight line forward. In the example depicted, the boat was moving forward at low speed to ensure the safe passage of the oncoming vessel on the left hand side. As can be seen from the figure, several features of the navigation bridge of the oncoming vessel are used for the flow estimation as well as a number of objects at the shore. Several features adjacent to the distant road bridge are tracked and registered as moving sideways. This is the result of small manoeuvring uncertainties of the boat: even in small turning manoeuvres, sidewards moving points are registered and used for the flow estimation.
From the first estimate of the picture, the Lucas-Kanade method is not considered as valuable contribution for visual odometry or SLAM algorithms. Tracking only a limited number of points instead of all pixels in the image might be suitable for saving computational resources. However, as the camera is not placed motionless, but on the moving boat, the flow estimation is prone to be compromised by turning manoeuvres. This is not suitable for applications like visual odometry or visual SLAM algorithms, that require precise relative motion estimation. Therefore, sparse optical flow estimation is not considered as the most suitable algorithm for the desired application.

4.4.2 Dense Optical Flow Estimation: The Farneback Method

While in sparse optical flow estimation, only a limited amount of pixels are tracked by the algorithm to maintain a reasonable usage of computational resources, dense optical flow estimation uses all of the pixels available from the frame. Depending on the movement of the objects taken into consideration, the output of such an algorithm might resemble semantic segmentation masks. Indeed, the authors of [100] apply the *Farneback method* [101] as dense optical flow estimation for assistance in semantic segmentation. The suggested algorithm is used for automated driving and achieves state-of-the-art results on the KITTI benchmark [100]. This approach is interesting to the application of inland waterway map generation, because the visual sensors are placed on the moving measuring platform, which is also the case in the application under discussion. Furthermore, dense optical flow estimation can support in improving the results of semantic segmentation [100].

The method suggested by *Farneback* shall be briefly presented in the following. The neighbourhood of each pixel is approximated with the second order polynomial [101]

$$f_{m,1}(x_m, y_m) \sim p_{m,1}(x_m, y_m) = \mathbf{x}_m^\top \mathbf{A}_{m,1} \mathbf{x}_m + \mathbf{b}_{m,1}^\top \mathbf{x} + c_{m,1} \tag{4.22}$$

with the matrices

$$\mathbf{x}_m = \begin{bmatrix} x_m \\ y_m \end{bmatrix}; \mathbf{A}_m = \begin{bmatrix} r_{m,4} & \frac{r_{m,6}}{2} \\ \frac{r_{m,6}}{2} & r_{m,5} \end{bmatrix}; \mathbf{b}_m = \begin{bmatrix} r_{m,2} \\ r_{m,3} \end{bmatrix}; c = r_{m,1}. \tag{4.23}$$

The coefficients $\mathbf{A}_m; \mathbf{b}_m;$ and c_m and consequently $r_{m,1}, \ldots, r_{m,6}$ are to be determined by fitting the signal f_m to the polynomial p_m using a weighted least squares method. The problem can be described by

$$argmin_{r_{m,1},\ldots,r_{m,6}} \sum_{x_m, y_m} (w_m(x_m, y_m)(f_m(x_m, y_m) - p_m(x_m, y_m)))^2 \tag{4.24}$$

with the weights

$$w_m(x_m, y_m) = \begin{cases} e^{\frac{x_m^2 + y_m^2}{2\sigma^2}}, & |x_m| \leq \frac{N_m - 1}{2}, |y_m| \leq \frac{N_m - 1}{2} \\ 0, & \text{otherwise} \end{cases} \tag{4.25}$$

where N_m is assumed to be odd and at least equal to 3. The weight function is used to assign equal importance to points in various directions within the neighbourhood. Furthermore it determines the size of the structures captured in the polynomial approximations [101]. As this computation needs to be performed over all pixels in both of the considered frames, it may become computationally intense. This problem is addressed by organizing the computations as a small number of convolutions [101]. With the local signal model constructed in a local coordinate frame by Eq. (4.22), also a new signal model can be constructed after the displacement $\mathbf{d}$ took place [102].

$$f_{m,2}(x_m, y_m) \sim p_{m,2}(x_m, y_m) = \mathbf{x}_m^\top \mathbf{A}_{m,2} \mathbf{x}_m + (\mathbf{b}_{m,1}^\top - 2\mathbf{A}_{m,1}\mathbf{d})^\top \mathbf{d}^\top \mathbf{A}_{m,1} \mathbf{d} - \mathbf{b}_{m,1}^\top \mathbf{d} + c_{m,1} \tag{4.26}$$

The second video frame can alternatively be represented by

$$f_{m,2}(x_m, y_m) \sim p_{m,2}(x_m, y_m) = \mathbf{x}_m^\top \mathbf{A}_{m,2} \mathbf{x}_m + \mathbf{b}_{m,2}^\top \mathbf{x} + c_{m,2} \tag{4.27}$$

with the coefficients

$$\mathbf{b}_{m,2} = \mathbf{b}_{m,1} - 2\mathbf{A}_{m,1}\mathbf{d} \tag{4.28}$$

$$2\mathbf{A}_{m,1}\mathbf{d} = -(\mathbf{b}_{m,2} - \mathbf{b}_{m,1}) \tag{4.29}$$

$$\mathbf{d} = -\frac{1}{2}\mathbf{A}_{m,1}^{-1}(\mathbf{b}_{m,2} - \mathbf{b}_{m,1}). \tag{4.30}$$

In addition to these basic equations of the Farneback optical flow estimation method, further refinements might be applied by parametrizing the displacement field with an appropriate motion model [102].
The Farneback method is applied to the same subset of the images gained from the measurement campaign as it was the case for the Lucas-Kanade method described in Section 4.4.1. Fig. 4.8 depicts the results of a first implementation of the Farneback

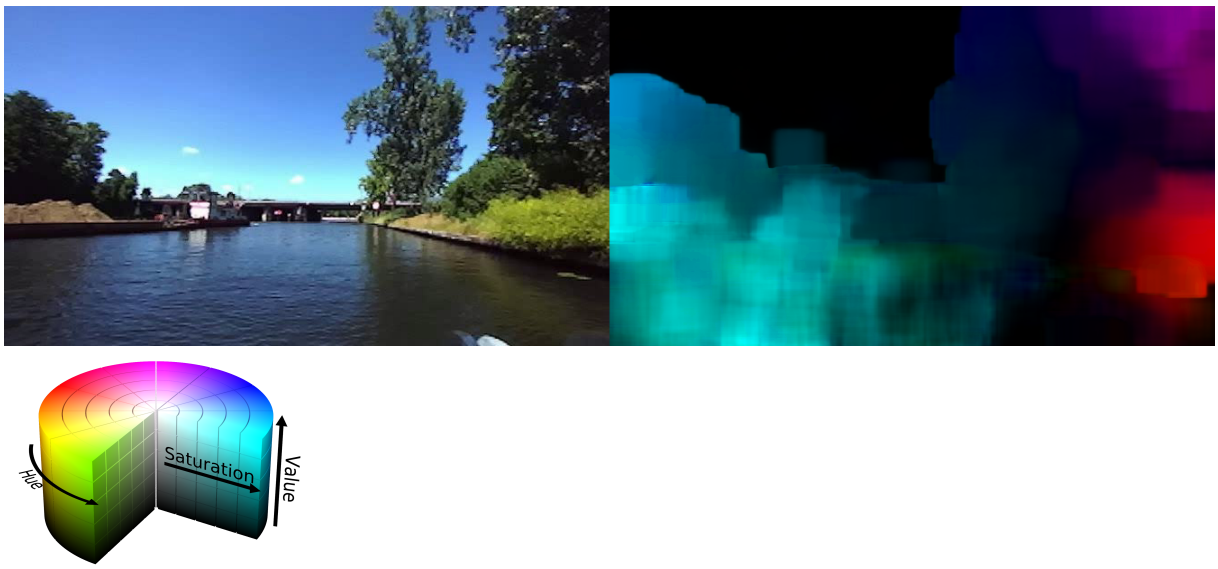


Figure 4.8: The Farneback method (right) is implemented on a subset of the RGB images gained during the measurement campaign. The colours encode the direction and magnitude of the moving objects as illustrated by the scale in the lower left corner. Besides the trees at the shoreline and the oncoming vessel, also smaller water waves are registered as features. The picture of the scene (left) is presented as reference.

algorithm on the RGB images gained during the measurement campaign. The frames are sampled with a rate of 10 Hz. The colors depicted in the right picture (Fig. 4.8) encode direction and magnitude of the moving objects. A colour scale illustrating the movements in HSV format is provided in the lower left corner of the figure. With the hue and the saturation, the defined colour encodes the direction of movement of the

tracked object. The value illustrates the magnitude of the movement. Starting on the right side of the camera FoV, the bank of the river can be recognized as tracked object. The red colour is the primary colour indicated by the colour scale. The trees above the river bank on the right hand side are marked in violet colour indicating longitudinal movement. Several trees on the left side of the flow estimation mask are marked in turquoise. As indicated by the colour scale, this colour illustrates a movement in the opposite direction compared to the red (primary) colour. The value of both tracked features on the left and right side of the mask indicates that the magnitude of the movement of these features is comparable. Also the superstructures of the oncoming vessel are registered for the flow estimation with slightly higher value when compared to the static objects. The higher magnitude of movement of the oncoming vessel is to be expected. However, the Farneback method classifies additionally certain small water waves in the middle of the waterway as objects, which appear in direct vicinity to the oncoming vessel. This is a limitation when being applied for spatial mapping purposes: in the current state, the application of the algorithm would be limited to conditions that exhibit an absolutely flat water surface.

The application of optical flow estimation is an extremely interesting field with respect to the use-case investigated in this work. Dense optical flow estimators are able to assist in semantic segmentation. Furthermore, flow estimators can assist in the development of visual SLAM algorithms, which are necessary for 3D map generation. Distinguishing between moving and non-moving objects in the gathered image data is a key-factor, that can play a valuable role when improving the accuracy of semantic segmentation algorithms. In general, dense optical flow estimators appear more valuable, especially for supporting semantic segmentation tasks, due to their ability of classifying every single pixel in the image. However, even very small waves on the waterway are often registered and tracked as features. The results from the Farneback method are only suitable for segmentation assistance if the flow estimation of the water surface is successfully denoised.

Unfortunately, further investigations in the integration of optical flow estimation are beyond the scope of this work. The valuable and interesting contribution of especially dense optical flow estimation for the application discussed in this work is therefore left for future work.

5. Conclusion

IWT plays a key role for European transportation systems, especially concerning mass goods. Inland vessels are an environmentally friendly alternative to heavy good road vehicles due to the lower emission of greenhouse gases and reduced effects by habitat damages. The transport volume of goods on inland waterways in Germany is expected to grow within the upcoming 10 years by 22 % [1].
Even though accidents on inland waterways are comparably rare, the existing risk of collision with infrastructure is a threat to the undisturbed traffic flow. High definition map broadcasts and constant updates on the chart display may minimize the risk of collisions and damages on the infrastructure. However, traditional means of surveying are intense in costs and time.
This problem can be addressed by equipping a relevant number of inland waterway vessels with optical sensors. The measured data is translated into spatial mapping information and broadcasted to generate a self-updating inland waterway chart.
In this work, an overall processing scope of optical sensor data, suitable for generating such self-updating inland waterway chart was shown. In particular, the work focused on developing a solution for semantic segmentation on RGB images using Convolutional Neural Networks. Due to the lack of appropriate training data, a designated dataset, the BerlinIWT dataset, was generated. For this task, a measurement campaign on the waterways of Berlin was carried out to gather optical sensor data. The most significant images were chosen from the dataset and annotated by four in-house annotators. While meeting the ambitious time constraints for the generation of a new training dataset, a total number of 171 examples were found to be useful as training data.
In a first scenario, the generated dataset was combined with the MaSTr1325 dataset.

The model was trained for 100 epochs to distinguish between the five different classes *sky*, *water*, *object*, *bridge* and *other* on an augmented dataset. While high accuracy values are easy to achieve in semantic segmentation tasks, the mean IoU is a far more suitable metric. Resulting in a mean IoU of 93.12 % on the test set, the performance of the model in this scenario is surprisingly good. The slightly lower corresponding value of 92.52 % examined on the validation set indicated room for further hyperparameter adjustments. Nevertheless, bearing in mind the limitations of available training data and computational resources, these results are very decent.
In a second scenario, the model was trained on 164 examples retrieved from the Berlin-IWT dataset only for performing two-class segmentation. For further optimization, the mean IoU was used as a loss function and the dataset was augmented. The test mean IoU of 91.50 % shows decent performance of the model, even though the value is slightly lower than the corresponding one in the five-class segmentation scenario. This effect is expected to be caused by the smaller training dataset. The validation mean IoU amounts to 89.86 % showing that the hyperparameter setting could be improved. Nevertheless, the results for both of the training scenarios show very decent performance of the model, which is especially true when considering the amount of available training data and the relatively low number of epochs the models have been trained.
Precise spatial mapping information requires not only semantic scene understanding, but also accurate distance measurements to the objects in vicinity. LiDAR sensors are a valuable source for precise distance information, outperforming stereo cameras in both, detection range and accuracy. Semantic scene understanding from 3D LiDAR point clouds only is an extremely complicated task. Therefore, the point clouds are aligned to the segmentation masks of the RGB images. This technique is widely used among the community. Several different approaches have been compared in this work to find the most suitable one for the specific problem in discussion. A proposed method called LDLS seemed to be the most promising one. Unfortunately, the practical implementation of this algorithm was not possible with the existing constraints on time and hardware. Semantic segmentation of LiDAR point clouds in inland waterway scenarios is therefore a topic that needs to be left for future research, even though a very promising concept was already theoretically presented in this work.
Finally, object tracking over time is another important component of spatial mapping

information. For this purpose, different optical flow estimation algorithms were applied in this work using the data collected during the mentioned measurement campaign. Dense optical flow estimation is prone to require a comparably high amount of computational resources, but fits better for the application discussed in this work. It estimates the displacement of every single pixel in the FoV of the camera. In consequence, features moving at similar speed and direction are classified in a similar way. Besides the benefit of motion detection and object tracking, the algorithm can therefore assist in semantic segmentation. In addition to rigid objects, such as the river bank and oncoming vessels, also small waves on the waterway may be detected as features. This circumstance can compromise the result of the dense optical flow estimation, depending on the sea-state of the waterway under survey. In any case, already very small waves, that can occur in light winds are capable of introducing considerable noise into the optical flow estimation mask. Making dense optical flow estimation robust against water waves is therefore another field of subsequent research activities.
This work contributes to the generation of a self-updating inland waterway chart by developing an approach for semantic segmentation on RGB camera data, that can be used for IWT infrastructure recognition. Due to the small dataset, first segmentation tasks are limited to the recognition of bridges. The results are promising: The trained model is able to detect all of the bridges provided in the dataset and recognizes the most relevant shapes of the bridge.

5.1 Traffic-Telematics related Evaluation

The approach presented in this work was developed for the application of a self-updating chart of inland waterways. The main motivation for an application like this is a more time and cost efficient way of surveying the infrastructure. Measurements of the infrastructure require nowadays a survey team, vehicles and appropriate equipment. In the future, they could be performed by vessels, that use the inland waterway for transportation purposes. By this approach, the effort in cost and time for IWT surveys as well as the obstruction for IWT due to survey works would be reduced. As soon as a vessel, equipped with the associated sensors, passes a waterway section and broadcasts the information gained to a platform, intentional and unintentional

changes in the IWT infrastructure would be recognized. For the time being, changes cannot be registered unless a survey team has carried out measurements in the corresponding section of the waterway. With the development for ordinary surveying in place, the approach could be extended for detecting anomalies in the infrastructure, such as damages and other threats to inland vessels. The early detection of compromised IWT infrastructure would minimize hazards, that result from the usage of the compromised section. Furthermore, restoration of damages detected at an early stage is prone to be less intense in cost and time. Spatial mapping information provided by (low cost) optical sensors might not be as exact as measurements carried out by a survey team. However, by numerically interpolating various measurements, performed by several different vessels, that pass through the same section at different times, sufficient accuracy could be reached.

For survey tasks of the infrastructure, optical sensors are particularly useful, as they provide an efficient and straight-forward way of gathering spatial mapping information from the environment. When compared to other perception systems (such as radar), most LiDAR sensors provide inferior maximum range. However, objects in direct vicinity to the sensor can be registered with surprisingly accurate resolution and distance information. For scanning the near environment, extremely large ranges are not needed. With the mentioned abilities, the contribution of LiDAR sensors in this field is very valuable. Semantic scene understanding is a key factor for survey applications. With a camera providing colour images, semantic segmentation can be achieved with excellent accuracy values at reasonable computational effort.

As described in Section 3.4, decent accuracy values are easy to achieve in semantic segmentation problems. On a dataset distinguishing between five different classes (Section 3.4.1), a mean IoU of 92.52, % and 93.12, % was achieved on the validation and test set, respectively. Indeed, this indicates very decent performance of the model. In a real-world use-case, the overall semantic understanding of a scene could be guaranteed with the current state of the implementation. This includes the recognition of bridges. However, not all fine details in the scene might be labelled correctly. This applies especially for the shapes of buildings and bridges. For accurate surveying, also fine structures are important, so the need for training longer and on more data arises. Still, the current implementation is suitable for the identification of navigable

areas. For the two-class-scenario (Section 3.4.2), the mean IoU of the validation and test set amounts to 89.86 % and 91.50 %, respectively, which indicates slightly worse performance. Still, also in this scenario, the recognition of a bridge can be guaranteed. However, the smaller training dataset introduces severe uncertainties in the recognition of finer structures. To fulfil the strict requirements of an application suitable for inland waterway survey, the mean IoU values of validation and test set would need to be improved in both of the training scenarios. Furthermore, the alignment with 3D LiDAR point clouds would be necessary for providing precise spatial mapping information.

The presented approach is not necessarily limited to the inland waterway transportation mode. The infrastructures of railway lines could be monitored in a similar way. Such infrastructure includes railway signals with their associated components, as well as parts of the track infrastructure, such as the state of switches, but also bridges, tunnels and the state of platforms could be monitored. A major difference to the IWT related application is that trains move considerably faster, which might be challenging for recognition and segmentation tasks. Installing optical sensors on modern trains in a way that their FoV remains unconstrained by other components might not always be possible. However, railway infrastructure companies would strongly benefit from an automatic survey application. Railway tracks are often difficult to access for survey teams. In daily operation, the safety of survey staff is not always easy to ensure, especially without impeding operating trains. Restricting the operation on a railway line can have severe effects on the operation throughout the whole network. Furthermore, many railway lines nowadays are working to their full capacity. It is therefore not always feasible to assign designated time-slots for survey vehicles operating on the tracks. In the railway domain, large amounts of persons or goods, and consequently high masses, at comparably high velocities are carried. In case of a collision, this leads to severe consequences, which requires high safety standards in daily operation. This includes the flawless state of the infrastructure. The benefit of performing measurements on the infrastructure simultaneously with the daily operation can be highlighted therefore particularly for the railway domain.

Probably the most straight-forward use-case is to monitor road traffic infrastructure using the discussed approach as an example. For this task, it needs to be ensured, that

the sensors are mounted to vehicles that use the infrastructure frequently and extensively. Namely, private vehicles may not be the best choice for this purpose, as they might be used irregularly. Depending on the road infrastructure that shall be monitored, sensors could be fitted to taxis, city or overland buses vehicles for public maintenance or long or short distance trucks. In particular, city-associated infrastructure, such as road signs, traffic lights, markings for traffic lanes and pavements could be monitored meaningfully by taxis, city buses or locally operating delivery trucks fitted with the necessary sensors. Infrastructure on national roads and highways, such as tunnels, bridges and guard rails would need to be surveyed by long-distance buses or trucks equipped with sensors. Precise global positioning requires the ability of GNSS positioning, that is installed in almost all modern cars. Additionally, an increasing number of vehicles are provided with optical sensors, that are determined to assist in autonomous usage of the vehicle. With the positioning and optical sensors in place, the requirements for precise spatial mapping information are fulfilled regarding the sensor hardware. As discussed throughout this work, a considerable number of training datasets designated for self-driving cars does already exist, that could be used as training data for an application that monitors road infrastructures.
The use of optical sensor data encourages the development towards autonomous vessels in the IWT domain. The semantic segmentation solution proposed in this work may be applied to identify navigable areas and docking locations. The retrieved information is supported with precise spatial mapping information provided by the LiDAR sensor. Optical flow estimation enables distinguishing the river bank and other static objects from oncoming and side-ways moving vessels. Optical sensors can offer a valuable contribution to collision avoidance, especially when supported with dense optical flow estimation. This application is already established in the automotive domain and could be implemented for the IWT application using a comparable approach. Finally, optical sensor data can support in short-term navigation tasks in case of GNSS outages. In the IWT domain, several scenarios can occur, that state a challenge for GNSS positioning. The passage under longer bridges can affect the availability of GNSS signals. In the automotive domain, an effect referred to as *urban canyon* causes reduced satellite visibility due to very high buildings to both sides of the street. Similar effects can be encountered in the IWT domain when a vessels enters a water-way lock at a low water

level. In these cases, a combination of optical sensor and IMU data can be used not only for collision avoidance, but also for short-term navigation and orientation of the vessel until reliable GNSS data is available again.
Overall, optical sensor data offers possibilities for precise and low-effort spatial mapping as well as advantages in autonomous vehicle applications in a broad spectrum of relevant traffic and transportation modes. With the application to spatial mapping of inland waterways, this work examined one of the possible use-cases.

5.2 Outlook and Future Work

In this work, a solution for optical sensor data processing for the application of self-updating inland waterway charts was proposed. Namely, a training dataset for semantic segmentation in the IWT domain has been generated and applied successfully to the associated task. Possibilities of combining camera and LiDAR data for semantic scene understanding as well as precise spatial mapping information have been discussed. Finally, different approaches of optical flow estimation have been compared with respect to the possible use-cases. While investigating the topic, several other interesting fields opened up, of which not all could be covered in this work.
As shown by the results, the semantic segmentation model was trained with averagely optimal hyperparameters. In future research, it would be beneficial to perform further hyperparameter optimization to achieve best performance of the model. One approach to this challenge could be to use stochastic optimization algorithms for hyperparameter optimization. Due to constraints in time and computational resources, this is a field left for future work.
For evaluating semantic segmentation results, the mean IoU is a more suitable metric than the accuracy. In one of the training scenarios, the mean IoU was even used as loss function. Another promising way to assess semantic segmentation results is the *Hausdorff distance*, as a geometrical, rather than a stochastic measure. Generally, the Hausdorff distance measures the distance between two subsets in a metric space, which is a promising approach when performing shape-estimation related tasks. The practical implementation of the Hausdorff distance as a metric in the current implementation of the training environment is an interesting field for future work. The training may ben-

efit from using the Hausdorff distance as loss function, which may exhibit additional challenges and is therefore another point left for future research.
Within this work, a new dataset was generated for semantic segmentation in the IWT domain. The dataset initially contained 190 examples. From the already comparably small dataset, various annotated masks were compromised in a way that made them unsuitable for training a NN. Within ongoing activities in this field, the compromised masks could be corrected to allow for addition to the existing dataset. In general, more images from the data collected during the measurement campaign could be annotated. To achieve the largest possible variety in the training data, also further measurement campaigns in different regions at different times of the year could be performed. Indeed, further measurement campaigns are planned already to enhance the increase of the BerlinIWT dataset. For ongoing annotation activities, however, it would be beneficial to define even stricter standards for the generated mask. This includes a more precise definition of the classes beforehand and even stricter control mechanisms on the quality of the mask. The application of a different annotation tool should be discussed. More application-related training data is a time-costly, but a straight-forward way to improve the performance of the model. The BerlinIWT dataset constitutes a novel and relevant contribution for the research community. Even more relevant, such dataset will encourage the German authorities to commission a testbed for (semi-) autonomous vessels in the waterways of Berlin.
The model in this work has been trained for 100 epochs on a dataset combined from the BerlinIWT and the MaSTr1325 dataset. The training on the augmented dataset required 20 hours on a NVidia Jetson Xavier AGX developer kit. For development of the best possible training conditions (used model, hyperparameters, dataset), it is desirable to decrease the time required for training. Presumably, small gains in the computational time could be achieved by optimizing the code in use. However, for more significant improvements, the use of a more powerful hardware is unavoidable. This would offer the possibility to perform more training epochs and reach the point where the model starts to exhibit overfitting behaviour on the mean IoU metric. With this knowledge, the best possible model for the use-case in discussion could be trained and selected.
With test mean IoU values of over 90 %, the results achieved by this implementation show already very decent performance. For real-world use, further improvements of

the mean IoU would be necessary, as the desired survey task requires high labelling accuracy, even of fine structures. The current state of the implementation might be sufficient to identify navigable areas. However, providing suitable chart data requires, that clearing heights and widths can be measured with centimetre-accuracy. For this goal, more accurate segmentation performance and the alignment of 3D point clouds with the segmentation masks is needed.

For the practical implementation of aligning semantically segmented RGB images and the corresponding 3D LiDAR point clouds, an existing algorithm called LDLS can be used. The installation requires a NVidia GPU. This requirement can be met by the Jetson Xavier used for training the NN. However, the space on the hard-disk of this device is limited. The installation of LDLS requires an additional hard-disk for the device, which could not be added in time. The practical implementation of LDLS and consequently LiDAR point cloud to RGB image alignment therefore needs to be left for future work.

Dense optical flow estimation offers the possibility to assist in semantic segmentation problems. This ability could be exploited in future implementations. To fully benefit from this advantage, the algorithm would need to adjusted in a way that prevents small waves on the waterway from being recognized as features. Furthermore, the current implementation is not capable of estimating the relative speed of the tracked features. A comparison between the speed retrieved from the GNSS solution and the velocity provided by the optical flow estimation would be extremely interesting. Optical flow estimation is particularly useful to estimate the movement of the measuring platform and provide assistance for SLAM-solutions. Towards the development of self-updating inland waterway charts, the implementation of a SLAM algorithm is the next logical step. Within this scope, the discussed solutions for optical flow estimation could perform a valuable contribution to the overall use-case.

Bibliography

[1] H. M. Müller. "Akademie für Raumforschung und Landesplanung (Ed.): Handwörterbuch der Stadt- und Raumentwicklung, ISBN 978-3-88838-559-9," in: ed. by Akademie für Raumforschung und Landesplanung. 2018. Chap. Binnenschifffahrt, pp. 243–251.

[2] Florian Hofbauer and Lisa-Maria Putz. "External costs in inland waterway transport: An analysis of external cost categories and calculation methods". In: *Sustainability* 12.14 (2020), p. 5874.

[3] Rainer Strenge, Michael Hoppe, Martin Bröschel, et al. "Assistenzsysteme für die Binnenschifffahrt basierend auf Hochpräzisions-DGNSS (Forschungsprojekt LAESSI)". In: *Deutsche Beiträge. 34. Internationaler Schifffahrtskongress; Panama City, Panama, 07.-11. Mai 2018* (2018), pp. 59–68.

[4] 2022 BBC. *Ever Given: Ship that blocked Suez Canal sets sail after deal signed*. July 2021. URL: https://www.bbc.com/news/world-middle-east-57746424.

[5] wasserstraßen und Schifffahrtsamt Elbe. *Wasserstraßenkreuz Magdeburg*. Aug. 2022. URL: https://www.wsa-elbe.wsv.de/Webs/WSA/Elbe/DE/Wasserstrassen/04_WassstrkreuzMagdeburg/WassstrkreuzMagdeburg_text.html.

[6] ResearchGate. *Figure 1*. Feb. 2022. URL: https://www.researchgate.net/figure/3D-model-of-Cernadela-Bridge-obtained-by-photogrammetry-and-camera-positions_fig1_273981291.

[7] generaldirektion Wasserstraßen und Schifffahrt. *Verkehrsnetz Bundeswasserstraßen*. Feb. 2022. URL: https://www.gdws.wsv.bund.de/SharedDocs/Downloads/DE/Karten/Karten_neu/DBWK1000_Generaldirektion.pdf;jsessionid=

DC0C84B2AF9BEB81AEF406FEA065DD13.live21322?__blob=publicationFile&v=12.

[8] You Li and Javier Ibanez-Guzman. "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems". In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.

[9] Yuhui Yuan, Xiaokang Chen, Xilin Chen, et al. "Segmentation transformer: Object-contextual representations for semantic segmentation". In: *arXiv preprint arXiv:1909.11065* (2019).

[10] D. Pierrotteta; F. Amzajerdianb; L. Petwayb; B. Barnesb; G. Lockardb; M. Rubio. "Linear FMCW Laser Radar for Precision Range and Vector Velocity Measurements". In: *Materials Research Society symposia proceedings. Materials Research Society* (2008).

[11] You Li and Javier Ibanez-Guzman. "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems". In: *IEEE Signal Processing Magazine* 37.4 (2020), pp. 50–61.

[12] Xiaolong Liu, Zhidong Deng, and Yuhan Yang. "Recent progress in semantic image segmentation". In: *Artificial Intelligence Review* 52.2 (2019), pp. 1089–1106.

[13] Jussi Taipalmaa, Nikolaos Passalis, and Jenni Raitoharju. "Different Color Spaces In Deep Learning-Based Water Segmentation For Autonomous Marine Operations". In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 3169–3173.

[14] Laura Lopez-Fuentes, Claudio Rossi, and Harald Skinnemoen. "River segmentation for flood monitoring". In: *2017 IEEE international conference on big data (Big Data)*. IEEE. 2017, pp. 3746–3749.

[15] Panqu Wang, Pengfei Chen, Ye Yuan, et al. "Understanding convolution for semantic segmentation". In: *2018 IEEE winter conference on applications of computer vision (WACV)*. Ieee. 2018, pp. 1451–1460.

[16] Marvin Teichmann, Michael Weber, Marius Zoellner, et al. "Multinet: Real-time joint semantic reasoning for autonomous driving". In: *2018 IEEE intelligent vehicles symposium (IV)*. IEEE. 2018, pp. 1013–1020.

[17] Farnoush Zohourian, Borislav Antic, Jan Siegemund, et al. "Superpixel-based Road Segmentation for Real-time Systems using CNN." In: *VISIGRAPP (5: VISAPP)*. 2018, pp. 257–265.

[18] Dan Levi, Noa Garnett, Ethan Fetaya, et al. "StixelNet: A Deep Convolutional Network for Obstacle Detection and Road Segmentation." In: *BMVC*. Vol. 1. 2. 2015, p. 4.

[19] Jens Behley, Martin Garbade, Andres Milioto, et al. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9296–9306.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.

[21] Weisong Wen, Yiyang Zhou, Guohao Zhang, et al. "Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2310–2316.

[22] G. Kuschk M. Meyer. "Automotive Radar Dataset for Deep Learning Based3D Object Detection". In: *Proceedings of the 16th European Radar Conference* (2019).

[23] Borja Bovcon and Matej Kristan. "Obstacle detection for usvs by joint stereo-view semantic segmentation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 5807–5812.

[24] Borja Bovcon and Matej Kristan. "A water-obstacle separation and refinement network for unmanned surface vehicles". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 9470–9476.

[25] Wei Wang, Banti Gheneti, Luis A. Mateos, et al. "Roboat: An Autonomous Surface Vehicle for Urban Waterways". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 6340–6347.

[26] Wei Wang, Tixiao Shan, Pietro Leoni, et al. "Roboat II: A Novel Autonomous Surface Vessel for Urban Environments". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 1740–1747.

[27] Wenqiang Zhan, Changshi Xiao, Yuanqiao Wen, et al. "Autonomous visual perception for unmanned surface vehicle navigation in an unknown environment". In: *Sensors* 19.10 (2019), p. 2216.

[28] Thales Shoiti Akiyama, José Marcato Junior, Wesley Nunes Gonçalves, et al. "Evaluating different deep learning models for automatic water segmentation". In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. IEEE. 2021, pp. 4716–4719.

[29] H. Skinnemoen L. Lopez-Fuentes C. Rossi. "River segmentation for flood monitoring". In: *2017 IEEE International Conference on Big Data (BIGDATA)* (2017).

[30] Pedro Santana, Ricardo Mendonça, and José Barata. "Water detection with segmentation guided dynamic texture recognition". In: *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2012, pp. 1836–1841.

[31] Jussi Taipalmaa, Nikolaos Passalis, Honglei Zhang, et al. "High-Resolution Water Segmentation for Autonomous Unmanned Surface Vehicles: a Novel Dataset and Evaluation". In: *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. 2019, pp. 1–6.

[32] Yuwei Cheng, Mengxin Jiang, Jiannan Zhu, et al. "Are we ready for unmanned surface vehicles in inland waterways? The usvinland multisensor dataset and benchmark". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3964–3970.

[33] Borja Bovcon, Jon Muhovič, Janez Perš, et al. "The mastr1325 dataset for training deep usv obstacle detection models". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 3431–3438.

[34] Mathieu Labbé and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation". In: *Journal of Field Robotics* 36.2 (2019), pp. 416–446.

[35] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[36] medium.com. *Implementing A Simple Artificial Neural Network from Scratch in Python*. May 2022. URL: `https://miro.medium.com/max/1200/1*Rfpe_jQjK2iRyKhM9Go6uw.png`.

[37] Bing Xu, Naiyan Wang, Tianqi Chen, et al. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[38] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, et al. "Unsupervised learning of invariant feature hierarchies with applications to object recognition". In: *2007 IEEE conference on computer vision and pattern recognition*. IEEE. 2007, pp. 1–8.

[39] Muhammad Usama, Junaid Qadir, Aunn Raza, et al. "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges". In: *IEEE Access* 7 (2019), pp. 65579–65615.

[40] Mirza Cilimkovic. "Neural networks and back propagation algorithm". In: *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin* 15.1 (2015).

[41] Léon Bottou et al. "Stochastic gradient learning in neural networks". In: *Proceedings of Neuro-Nımes* 91.8 (1991), p. 12.

[42] Michael Meyer and Georg Kuschk. "Automotive radar dataset for deep learning based 3d object detection". In: *2019 16th european radar conference (EuRAD)*. IEEE. 2019, pp. 129–132.

[43] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[45] Valentina Emilia Balas, Raghvendra Kumar, and Rajshree Srivastava. *Recent trends and advances in artificial intelligence and internet of things*. Springer, 2020.

[46] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. "Equivariance through parameter-sharing". In: *International conference on machine learning*. PMLR. 2017, pp. 2892–2901.

[47] Diego Pierrottet, Farzin Amzajerdian, Larry Petway, et al. "Linear FMCW laser radar for precision range and vector velocity measurements". In: *MRS Online Proceedings Library (OPL)* 1076 (2008).

[48] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, et al. "Inception recurrent convolutional neural network for object recognition". In: *arXiv preprint arXiv:1704.07709* (2017).

[49] Li Deng. "The mnist database of handwritten digit images for machine learning research [best of the web]". In: *IEEE signal processing magazine* 29.6 (2012), pp. 141–142.

[50] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[51] Chiyuan Zhang, Samy Bengio, Moritz Hardt, et al. "Understanding deep learning (still) requires rethinking generalization". In: *Communications of the ACM* 64.3 (2021), pp. 107–115.

[52] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[53] Leslie N. Smith. "Cyclical Learning Rates for Training Neural Networks". In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, pp. 464–472.

[54] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, et al. "On large-batch training for deep learning: Generalization gap and sharp minima". In: *arXiv preprint arXiv:1609.04836* (2016).

[55] Kaichao You, Mingsheng Long, Jianmin Wang, et al. "How does learning rate decay help modern neural networks?" In: *arXiv preprint arXiv:1908.01878* (2019).

[56] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[57] Liang-Chieh Chen, George Papandreou, Florian Schroff, et al. "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (2017).

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[59] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[60] Olga Russakovsky, Jia Deng, Hao Su, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[61] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, et al. "Color transfer between images". In: *IEEE Computer graphics and applications* 21.5 (2001), pp. 34–41.

[62] SICK AG. *Operating Instructions - SICK MRS6000*. SICK Sensor Intelligence. July 2019.

[63] Alberding GmbH 2021. *AutonomSOWII*. July 2022. URL: `https://www.autonomsow.de/`.

[64] DLR. *Spree-Oder-Wasserstraße wird digitales Testfeld für hochautomatisierte und vernetze Binnenschifffahrt*. July 2022. URL: `https://www.dlr.de/content/de/artikel/news/2021/03/20210928_von-der-strasse-aufs-wasser.html`.

[65] BEHALA Berliner Hafen und Lagerhausgesellschaft mbH. *BEHALA - individuelle Logistik- und Immobilien-Lösungen aus dem Zentrum Berlins*. July 2022. URL: `https://www.behala.de/`.

[66] Intel technologies. *Computer Vision Annotation Tool*. July 2022. URL: `https://cvat.org/auth/login`.

[67] Piotr Skalski. *MakeSense.ai*. July 2022. URL: `https://www.makesense.ai/`.

[68] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[69] Antonio Torralba, Bryan C Russell, and Jenny Yuen. "Labelme: Online image annotation and applications". In: *Proceedings of the IEEE* 98.8 (2010), pp. 1467–1484.

[70] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, et al. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[71] You Li. "Stereo vision and Lidar based dynamic occupancy grid mapping: Application to scenes analysis for intelligent vehicles". PhD thesis. Université de Technologie de Belfort-Montbeliard, 2013.

[72] Jiaxin Li and Gim Hee Lee. "Deepi2p: Image-to-point cloud registration via deep classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15960–15969.

[73] Zhuangwei Zhuang, Rong Li, Kui Jia, et al. "Perception-Aware Multi-Sensor Fusion for 3D LiDAR Semantic Segmentation". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16260–16270.

[74] Pouria Babahajiani, Lixin Fan, Joni-Kristian Kämäräinen, et al. "Urban 3D segmentation and modelling from street view images and LiDAR point clouds". In: *Machine Vision and Applications* 28.7 (2017), pp. 679–694.

[75] Ran Cheng, Christopher Agia, Yuan Ren, et al. "S3cnet: A sparse semantic scene completion network for lidar point clouds". In: *arXiv preprint arXiv:2012.09242* (2020).

[76] Kanrun Huang et al. "End-to-End Multi-Sensor Fusion for 3d Object Detection in Lidar Point Clouds". In: *Applied & Educational Psychology* 2.1 (2021), pp. 67–72.

[77] Brian H. Wang, Wei-Lun Chao, Yan Wang, et al. "LDLS: 3-D Object Segmentation Through Label Diffusion From 2-D Images". In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 2902–2909.

[78] Jorge Beltrán, Carlos Guindel, Francisco Miguel Moreno, et al. "Birdnet: a 3d object detection framework from lidar information". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3517–3523.

[79] Carlos Guindel, David Martín, and José María Armingol. "Joint object detection and viewpoint estimation using CNN features". In: *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE. 2017, pp. 145–150.

[80] Andres Milioto, Ignacio Vizzo, Jens Behley, et al. "RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4213–4220.

[81] Joel M. Esposito and Mitchell Graves. "An algorithm to identify docking locations for autonomous surface vessels from 3-D LiDAR scans". In: *2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*. 2014, pp. 1–6.

[82] Ulla Wandinger. "Introduction to lidar". In: *Lidar*. Springer, 2005, pp. 1–18.

[83] Diego Pierrottet, Farzin Amzajerdian, Larry Petway, et al. "Linear FMCW laser radar for precision range and vector velocity measurements". In: *MRS Online Proceedings Library (OPL)* 1076 (2008).

[84] Ryan S. Kaminsky, Noah Snavely, Steven M. Seitz, et al. "Alignment of 3D point clouds to overhead images". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 2009, pp. 63–70.

[85] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. "Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11366–11374.

[86] Bichen Wu, Alvin Wan, Xiangyu Yue, et al. "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1887–1893.

[87] Sundar Vedula, Simon Baker, Peter Rander, et al. "Three-dimensional scene flow". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. IEEE. 1999, pp. 722–729.

[88] Zike Yan and Xuezhi Xiang. "Scene flow estimation: A survey". In: *arXiv preprint arXiv:1612.02590* (2016).

[89] Jisoo Jeong, Jamie Menjay Lin, Fatih Porikli, et al. "Imposing Consistency for Optical Flow Estimation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3181–3191.

[90] Guo Lu, Wanli Ouyang, Dong Xu, et al. "Dvc: An end-to-end deep video compression framework". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11006–11015.

[91] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. "Video compression through image interpolation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 416–431.

[92] Myunggi Lee, Seungeui Lee, Sungjoon Son, et al. "Motion feature network: Fixed motion filter for action recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 387–403.

[93] Zixi Cai, Helmut Neher, Kanav Vats, et al. "Temporal hockey action recognition via pose and optical flows". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[94] Kireeti Bodduna and Joachim Weickert. "Removing multi-frame Gaussian noise by combining patch-based filters with optical flow". In: *Journal of Electronic Imaging* 30.3 (2021), p. 033031.

[95] Valéry Dewil, Jérémy Anger, Axel Davy, et al. "Self-supervised training for blind multi-frame video denoising". In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2021, pp. 2724–2734.

[96] Kiran Kale, Sushant Pawar, and Pravin Dhulekar. "Moving object tracking using optical flow and motion vector estimation". In: *2015 4th international conference on reliability, infocom technologies and optimization (ICRITO)(trends and future directions)*. IEEE. 2015, pp. 1–6.

[97] Thomas Brox Huizhong Zhou Benjamin Ummenhofer. "Deeptam: Deep tracking and mapping". In: *Proceedings of the European conference on computer vision (ECCV)* (2018).

[98] Bruce D Lucas, Takeo Kanade, et al. *An iterative image registration technique with an application to stereo vision*. Vol. 81. Vancouver, 1981.

[99] Román Mondragón, Joaquín Alonso-Montesinos, David Riveros-Rosas, et al. "Determination of cloud motion applying the Lucas-Kanade method to sky cam imagery". In: *Remote Sensing* 12.16 (2020), p. 2643.

[100] Ahmad El-Sallab Pavel Krizek Mohamed El-Helw Hazem Rashed Senthil Yogamani. "Optical Flow augmented Semantic Segmentation networksfor Automated Driving". In: *https://doi.org/10.48550/arXiv.1901.07355* (2019).

[101] Gunnar Farnebäck. "Disparity estimation from local polynomial expansion". In: *SSAB Symposium on Image Analysis, March 2001, Norrköping, Sweden*. 2001, pp. 77–80.

[102] Shivangi Anthwal and Dinesh Ganotra. "Optical Flow Estimation in Synthetic Image Sequences Using Farneback Algorithm". In: *Advances in Signal Processing and Communication*. Springer, 2019, pp. 363–371.

Erklärung

Hierdurch erkläre ich, dass ich die von mir am heutigen Tage eingereichte Diplomarbeit selbständig verfasst und andere als die angegebenen Hilfsmittel nicht benutzt habe.

Dresden, 26. August 2022