Autonomous Vehicle Perception and Navigation in Adverse Conditions

Alvari Seppänen
Autonomous Vehicle Perception and Navigation in Adverse Conditions

Alvari Seppänen

A doctoral thesis completed for the degree of Doctor of Science (Technology) is to be defended, with the permission of the Aalto University School of Engineering, at a public examination held at the lecture hall K1/216 of the school on 26 April 2024 at noon.

Aalto University
School of Engineering
Department of Mechanical Engineering
Autonomous Mobility Laboratory
Supervising professors
Professor Kari Tammi
Aalto University
Finland

Preliminary examiners
Professor Wilhelm Stork
Karlsruhe Institute of Technology
Germany

Professor Martin Törngren
KTH Royal Institute of Technology
Sweden

Opponents
Professor Martin Törngren
KTH Royal Institute of Technology
Sweden
**Abstract**

**Author**
Alvari Seppänen

**Name of the doctoral thesis**
Autonomous Vehicle Perception and Navigation in Adverse Conditions

**Publisher**
School of Engineering

**Unit**
Department of Mechanical Engineering

**Series**
Aalto University publication series DOCTORAL THESES 53/2024

**Field of research**
Mechatronics

**Manuscript submitted** 23 October 2023  
**Date of the defence** 26 April 2024

**Permission for public defence granted (date)** 12 February 2024  
**Language** English

**Abstract**

Autonomous mobility has gained popularity in recent years due to the promise of safer and more efficient transportation systems. However, multiple challenges hinder the realization of fully autonomous transportation, e.g., safety, operational environment limitations, and efficiency. This thesis addresses challenges related to the perception and navigation of outdoor mobile robots in adverse conditions. These conditions refer to adverse weather and limited communication between a remote operator and the robot.

Adverse weather conditions affect the perception systems, namely light detection and ranging (LiDAR) sensors, causing specific types of noise to the data. This work aims to denoise this data and thus provide clean data for downstream systems. Two deep-learning-based denoising approaches are proposed: a supervised approach that utilizes a spatiotemporal module and a self-supervised multi-echo approach. The supervised method's spatiotemporal module enables efficient data usage and generalization from semi-synthetic to fully real-world data. The self-supervised approach learns by predicting the correlation of data points to their neighbors and utilizes multi-echo point clouds for recovering the points representing solid objects. Experiments show that both approaches achieved state-of-the-art performance.

Another challenge addressed in this thesis is the navigation in adverse conditions. These challenges refer to limited remote communication caused, for example, by adverse weather conditions. The limited communication between teleoperators and semi-autonomous mobile robots is studied. Semi-autonomous control strategies are proposed to aid the teleoperators when communication signal limits the system’s performance. Experiments with a mobile robot prototype revealed that the strategies improved the navigation.

Future research should focus on testing the denoising with downstream algorithms and assessing the control strategies in more complex environments. Many adverse and unexpected scenarios must be addressed to realize fully autonomous vehicles in complex environments. Therefore, more unified solutions tackling multiple issues simultaneously are desired in future research.

**Keywords**
Autonomous Vehicle, Adverse Conditions, Computer Vision, Deep Learning

**ISBN (printed)** 978-952-64-1714-1  
**ISBN (pdf)** 978-952-64-1715-8

**ISSN (printed)** 1799-4934  
**ISSN (pdf)** 1799-4942

**Location of publisher** Helsinki  
**Location of printing** Helsinki  
**Year** 2024

**Pages** 114  
Tekijä
Alvari Seppänen

Väitöskirjan nimi
Autonomisen ajoneuvon havainnointi ja liikkuvuus epäsuotuisissa olosuhteissa

Julkaisija
Insinöörityyppiden korkeakoulu

Yksikkö
Koneetekniikan laitos

Sarja
Aalto University publication series DOCTORAL THESES 53/2024

Tutkimusala
Mekatroniikka

Käsikirjoituksen pvm
23.10.2023

Väitöspäivä
26.04.2024

Väittelyluvan myöntämispäivä
12.02.2024

Kieli
Englanti

Monografia

Artikkeliväitöskirja

Esseeväitöskirja

Tiivistelmä
Autonomiset ajoneuvot ovat kasvattaneet suosiotaan viime vuosina lupaamalla turvallisempia ja tehokkaampia liikennejärjestelmiä. Täysin autonomisten ajoneuvojen toteutumista haettaavat kuitenkin monet haasteet, kuten turvallisuus, toimintaympäristön rajoitukset ja tehokkuus.

Tämä väitöskirja käsittelee haasteita, jotka liittyvät ulkona liikkuvien robottien havaitsemiseen ja navigointiin epäsuotuisissa olosuhteissa. Nämä olosuhteet viittaavat huonoon sähän ja rajoitettuun viestintään etäkäyttäjän ja robotin välillä.


Avainsanat
Autonominen ajoneuvo, Epäsuotuisat ololisuhteet, Konenäkö, Syväoppiminen

ISBN (painettu)
978-952-64-1714-1

ISBN (pdf)
978-952-64-1715-8

ISSN (painettu)
1799-4934

ISSN (pdf)
1799-4942

Julkaisupaikka
Helsinki

Painopaikka
Helsinki

Vuosi
2024

Sivumäärä
114

URN
# Contents

1. **Introduction**  
   1.1 Motivation and scope  
   1.2 Outline  

2. **Related work**  
   2.1 Adverse weather denoising  
   2.2 Mobile robot navigation with communication issues  

3. **Methods**  
   3.1 Adverse weather denoising from adjacent point clouds  
   3.2 Self-supervised multi-echo denoising in adverse weather  
   3.3 Semi-autonomous control strategies in the presence of large delay fluctuation  

4. **Experiments**  
   4.1 Adverse weather denoising from adjacent point clouds  
   4.2 Self-supervised multi-echo denoising in adverse weather  
   4.3 Semi-autonomous control strategies in the presence of large delay fluctuation  

5. **Discussion**  
   5.1 Adverse weather denoising from adjacent point clouds  
   5.2 Self-supervised multi-echo denoising in adverse weather
List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s Contribution

Publication I: “4DenoiseNet: Adverse weather denoising from adjacent point clouds”

The author conceptualized, developed, and implemented the researched denoising algorithm. The author designed the experiments with the help of Ojala, prepared the research equipment, benchmarked the algorithms, and produced the results. Tammi supervised the work and acquired the project funding. The author wrote the manuscript with the help of the co-authors. All authors commented on the manuscript and reviewed the final publication.

Publication II: “Self-Supervised Multi-echo Point Cloud Denoising in Adverse Weather”

The author conceptualized, developed, and implemented the researched denoising algorithm. The author designed the experiments with the help of Ojala, prepared the research equipment, benchmarked the algorithms, and produced the results. Tammi supervised the work and acquired the project funding. The author wrote the manuscript with the help of the co-authors. All authors commented on the manuscript and reviewed the final publication.

Publication III: “Comparison of Semi-autonomous Mobile Robot Control Strategies in Presence of Large Delay Fluctuation”

The author implemented the researched system, assembled the hardware, and prepared the code. The author designed and conducted the experiments and gathered the measurement data. The author processed the
acquired measurement data and prepared the related figures. Vepsäläinen and Ojala assisted in designing the hardware and experimental setup. Tammi initiated the concept development, supervised the work, and acquired the project funding. The author wrote the manuscript with the assistance of the co-authors. All authors commented on the manuscript and reviewed the final publication.
Abbreviations

2D two-dimensional
3D three-dimensional
4D four-dimensional
4DenoiseNet four-dimensional denoising network
CDA control-dependent assist
CNN convolutional neural network
CSR characteristics similarity regularization
DDA delay-dependent assist
IoU intersection over union
KNN k-nearest neighbors
LiDAR light detection and ranging
ReLU rectified linear unit
SMEDen self-supervised multi-echo denoising
Symbols

**bolded font** matrix or tensor

$I$ intensity

$k$ number of neighbors

$L$ loss

$O$ output of a neural network

$P$ point cloud

$P_r$ point cloud range coordinate

$P_{xyz}$ point cloud Cartesian coordinate

$r, \theta, \phi$ spherical coordinate system

$S$ sparsity

$u, v$ pixel coordinate system

$w$ trainable parameters

$x, y, z$ Cartesian coordinate system
1. Introduction

Autonomous vehicles have gained significant attention and investment in recent years due to their potential to revolutionize transportation and improve road safety. However, one of the critical challenges that autonomous vehicles face is to retain performance in adverse conditions. In this thesis, adverse conditions refer to challenging environmental factors such as poor weather conditions (e.g., heavy rain, snowfall, fog) and compromised communication (e.g., control delay) that can affect the performance and safety of autonomous and semi-autonomous vehicles. The first affects both autonomous and semi-autonomous systems. Compromised communication affects semi-autonomous systems with teleoperation capability, where managing control delay can be safety-critical. This thesis aims to investigate and develop novel perception and navigation techniques for autonomous vehicles to operate effectively in these adverse conditions. The findings of this research contribute to the advancement of autonomous vehicle technology and enhance their safety and reliability in adverse conditions, bringing the field one step closer to realizing the potential of autonomous vehicles for future transportation.

1.1 Motivation and scope

Autonomous vehicles typically utilize advanced sensor systems such as cameras and Light detection and ranging (LiDAR) for perception. Environmental factors, such as weather can affect these systems and consequently disrupt the operation of the vehicle. Therefore, making the sensors and their data processing more reliable should make the operation of the vehicle more reliable. This thesis aims to make perception systems robust against adverse weather and thus make the vehicle more robust in those conditions.

LiDAR is a popular sensor in autonomous vehicle applications because it produces accurate 3D information. However, adverse weather conditions can impact LiDAR point clouds significantly, causing unwanted noise
Introduction

Undesired absorptions, refractions, and reflections of the LIDAR signal are caused by rain droplets [1, 2, 3], fog [4, 1, 5, 3], and snowfall [6, 7, 8, 9, 3]. This manifests as cluttered and missing data points. The issue is critical as a typical use case for LiDAR is to determine the accessible volume of the environment by detecting obstacles. Moreover, downstream perception algorithms such as object detection are affected [10, 11, 12, 13]. These downstream perception tasks are vital components in automated driving and driving assistance systems and directly relate to the safety of these systems. Furthermore, human drivers have a notably higher rate of accidents in adverse weather conditions, as reported by the European Commission [14] and the US Department of Transportation [15]. Therefore, the reliable performance of these systems becomes more crucial.

The first part of the thesis focuses on denoising LiDAR point clouds corrupted by adverse weather conditions. In this context, denoising typically means that the points caused by adverse weather are removed from the point cloud and other points remain unaltered [16, 5, 17]. After this denoising process, the point cloud is more suitable for downstream tasks as it ideally includes no noise caused by adverse weather. Two novel deep learning methods are presented and their applicability to this task is investigated. The first one (PI) uses improved spatiotemporal features and investigates how a model trained with semi-synthetic data generalizes to real-world data. The second (PII) proposes to use multi-echo point clouds and a self-supervised approach. Therefore, the training can be conducted with unannotated real-world data. Both publications aim to train the models without manually produced labels.

Contributions related to the adverse weather denoising. To summarize the contributions of these two works:

• A deep learning approach for denoising weather-corrupted point clouds utilizing spatial and temporal information is presented. More specifically, a novel k-nearest neighbors search convolution on consecutive point clouds is proposed. This approach sets the new state-of-the-art while having a lower computational cost.

• The model is trained with semi-synthetic data generated by a physics-based model [12] and tested on how well it generalizes to real-world data. The results indicate the model generalizes well to real-world data. Thus, the training data is deemed suitable for the task. The dataset, i.e., SnowyKITTI, is published with approximately $4 \cdot 10^4$ LiDAR scans.

• The task of multi-echo denoising for LiDAR in adverse weather is proposed and formulated.

• A novel self-supervised approach that learns this task is presented.
Introduction

The approach is based on a novel blind spot method and neighborhood correlation.

- Characteristics similarity regularization is proposed to boost the performance of self-supervised methods.

The second part of the thesis focuses on the navigation of mobile robots in adverse conditions. Semi-autonomous vehicles that have teleoperation functions are studied. Teleoperation can be used, e.g., when the autonomous operation is unreliable, it cannot perform a given part of the objective, or if the task is safety-critical. Teleoperation is usually implemented over the Internet as a bilateral control system. However, due to the nature of Internet protocols, remote control introduces additional control and feedback delays to the system, which can vary depending on factors such as adverse weather conditions, network congestion, solar activity, geography, and buildings. As a result, the fluctuating control delay poses a challenge to teleoperation and can result in accidents and delayed mission operations.

Previous studies have explored teleoperation under conditions of fluctuating control delay. One approach that has been investigated is delay passivation, which adjusts the control based on the magnitude of the delay. This method has been shown to effectively smooth the control and aid teleoperation when delays are low, as demonstrated in studies such as [18, 19, 20]. However, when delays become too large, the control is halted until the delay falls below a predefined threshold value. The threshold value used in previous research has varied from 300 ms to 2000 ms, as documented in studies such as [18, 21].

Due to the delay, teleoperation missions take longer and may render the system vulnerable in critical situations where continuous motion is essential. To address this, autonomously assisted control strategies for high control delay scenarios are presented, referred to as Delay-Dependent Assist (DDA) and Control-Dependent Assist (CDA). DDA activates autonomous control when the delay threshold is reached and CDA assists the teleoperator when control is deemed inadequate, similar to the typical commercial lane-keeping assistants in passenger vehicles.

Contributions related to the semi-autonomous control strategies. Previous studies have successfully aided a robot manipulator teleoperator using semi-autonomous control, as reported by the authors in [22, 23, 24, 25, 26]. In this thesis, the objective is to implement a semi-autonomous teleoperation system for a maintenance robot operating in the European Organization for Nuclear Research (CERN) particle accelerator tunnel and validate it through experiments conducted in a mock-up environment of the tunnel. In these experiments, different control strategies for teleoperation are compared (DDA, CDA, delayed manual, manual, and au-
tonomous) under fluctuating control delay. In addition to the contribution of the semi-autonomous control strategies, a new RGB-D image-based artificial potential field (RGB-DPF) method is utilized for obstacle avoidance and route following of autonomous control. To summarize, a large-scale practical experiment compares two novel control strategies (DDA and CDA) to manual and autonomous control.

1.2 Outline

The thesis is organized as follows. The related work is presented first, separated into distinctive categories. It also includes research gaps in each article. Then, the methods are presented, where each section is devoted to each article. A similar division is done with the experiments and discussion presented after the methods in the respective order. The last section concludes the thesis.
2. Related work

2.1 Adverse weather denoising

This section summarizes related work that has addressed the issues with LiDAR data in adverse weather conditions. Adverse weather denoising for LiDAR data is typically implemented with classical or learned methods. Classical methods are hand-crafted algorithms based on the noise characteristics, mainly density and intensity. Learned methods are typically deep neural networks, particularly convolutional neural networks (CNN), which are trained in a supervised or self-supervised manner. The learned methods have superior performance but require more data and computation than the classical methods.

A simple classical method is a median filter that operates on the projected point cloud image [27]. Thus, it is reminiscent of traditional image denoising. Experiments on adverse weather denoising indicate that the median filter is suitable for this purpose. Several methods utilize the fact that noise points are typically more sparse than others. Rusu et al. [28] proposed radius outlier removal (ROR) and statistical outlier removal (SOR). They remove points based on their distance to the $k$ nearest neighbors, and a fixed threshold is used for classifying the points. This poses an issue with LiDAR or other point-source radiation sensors producing point clouds as they have inherently varying densities. Dynamic radius outlier removal (DROR) [17] removes this issue by adjusting the threshold based on the measured distance to the sensor. Therefore, points far away from the sensor have a larger threshold than those closer to the sensor. Points caused by adverse weather typically cause smaller intensities in LiDAR. Low-intensity outlier removal (LIOR) [29] utilizes this phenomenon when deciding which points should be removed. It is a combination of ROR and intensity-based threshold. An improvement was made in LIOR by combining it with DROR by Wang et al. [30]. Later on, spatiotemporal features are used to identify the noise. Li et al. [31] utilized the unpredictable
trajectories of the points caused by adverse weather.

Deep learning has been trending in many fields, including adverse weather denoising. Essentially, the task is to segment the noise. Therefore, general-purpose semantic segmentation networks can be utilized. Most popular ones process the raw point cloud format [32, 33, 34, 35, 36, 37, 38, 39], voxelized point cloud [40, 41, 42], or spherically projected point cloud [43, 44, 45, 46]. These can be utilized for adverse weather denoising, but they lack performance as they are not explicitly designed for the task.

The first deep learning method intended for adverse weather denoising was introduced by Heinzler et al. [5]. They trained a specialized CNN (WeatherNet) with supervised learning to identify the noise from projected point clouds. Its performance surpassed all classical methods by a significant margin. WeatherNet was explicitly developed for this task, and the authors noted that dilated convolution is beneficial. WeatherNet established the projection-based approach. Thus, the following work has taken a similar route. Yu et al. [47] constructed a self-supervised loss function using fast Fourier and discrete wavelet transforms. Bae et al. [16] introduced the reconstruction difficulty approach to training in a self-supervised manner.

Deep learning in the context of regular image denoising is a more established field. It has similarities to adverse weather denoising as both eliminate noise from the data. One of the first works in deep image denoising is Noise2Noise [48]. Their method aims to map one image to a second image with the same underlying signal and learn the expected value, which is the clean image acquired. This utilizes the simple statistical principle that the expected value of multiple measurements of the same signal is the signal without noise, assuming that the mean value of the noise is zero. Other methods that do not require image pairs have been developed [49, 50, 51, 52], which have more flexible use cases. However, deep image denoising does not apply directly to adverse weather denoising for LiDAR point clouds, as the task is to remove noise points and leave other points unaltered. Whereas image denoising typically alters all pixels. Dense point cloud denoising methods have an equivalent objective to deep image denoising [53, 54]. However, the adverse weather denoising task has not been tested with dense point cloud denoising methods.

PI improves the performance of supervised deep learning-based filtering by introducing the 4DenoiseNet architecture. Its main difference from previous work is the better use of spatio-temporal information. More precisely, the input is processed with a convolution kernel that collects \( k \) nearest spatial and temporal neighbors. This allows the network to learn more generalizable features from the point cloud.

PII presents a novel task of multi-echo denoising and a self-supervised architecture to denoise point clouds. The task differs from the previous approaches by utilizing multi-echo point clouds, where the benefit is that
Related work

information unavailable in single echo point clouds is used. The key idea is that the echo corresponding to the object of interest is picked from the multi-echo point clouds. The self-supervised architecture differs from previous denoising systems in that it uses a novel neighborhood correlation method, a novel characteristics similarity regularization method, and minor methodological differences.

2.2 Mobile robot navigation with communication issues

In addition to handling the adverse weather-caused issues related to the LiDAR perception. This thesis also addresses navigation when there are communication issues between the teleoperator and the robot. These issues can be caused, for example, by adverse weather conditions. However, in the context of this thesis, delay stems from internet communication and its limitations in long and narrow tunnels.

The field of mobile robot teleoperation is well-established, with studies focusing on delay management to enhance real-time internet-based teleoperation. Internet-based teleoperation encounters significant issues related to delay fluctuations, video feedback delay, and limited bandwidth, as shown in [18]. To address communication issues, semi-autonomous control systems have been developed to support the teleoperator.

In their work, You et al. [22] introduced a semi-autonomous control system for a hexapod robot, where the operator controlled the body velocity. At the same time, the agent automatically computed the corresponding angular velocities of the leg joints. The operator received haptic feedback that was proportional to the error between the actual and desired velocities of the body, with the goal of improving teleoperation accuracy and intuitiveness. The proposed system was validated through experiments in a semi-physical simulation. However, the authors did not investigate the impact of the delay of the control signal.

The control of a semi-autonomous teleoperated system with time-varying delays and input uncertainties has been studied by the authors in [24, 25, 55]. Their systems were based on local and remote robot position and velocity tracking errors. They aimed to improve the performance of teleoperation in a simulation setting. However, the authors only considered periodic delay fluctuations with a high frequency of 0.5 Hz and did not investigate the effects of prolonged and high delay periods.

The semi-autonomous systems typically have some autonomous functions to aid the teleoperator. In this thesis, the autonomous function is a path-planning method known as the artificial potential field (PF) method. Therefore, a short literature summary on the PF method is presented here. PF method uses virtual repulsive and attractive fields to represent obstacles and goals. The total field is the sum of these virtual fields, and
the path is planned by following the descent direction of the total field to ensure that the robot avoids obstacles and reaches the goal. The repulsive field can be generated directly from 3D sensor data, where obstacles have potential if their location does not satisfy a threshold. Khatib et al. [56] first formulated the method, which has since been improved and implemented in various studies such as [57, 58, 59, 60, 61, 62, 63, 64].

One of the main problems of the PF method is the tendency to get stuck on local stable points. Various variants of the PF method have been developed to deal with this problem. Yao et al. [59] proposed a black hole potential field method that utilizes a black-hole analog that improved the traditional PF method. The method was designed to bend the gradient of the PF to avoid local stable points more effectively. Experiments with static and dynamic obstacles and goals demonstrated that the robot performed better with local stable points. Orozco-Rosas et al. [61] developed a hybrid algorithm that used membrane computing, the pseudo-bacterial genetic algorithm, to avoid the local stable point problem. Jang-Ho and colleagues [60] employed the obstacle-dependent Gaussian potential field (ODG-PF), which uses the Gaussian function to define obstacles. To prevent the local stable point issue, they utilized the global minimum of the field as the desired direction. Their simulations and experiments demonstrated the stable performance of the ODG-PF in the presence of static and dynamic obstacles. They conducted their experiments using a mobile robot equipped with a 2D LiDAR. Rasekhipour et al. [65] applied the PF method to road vehicle path planning. Their approach considers road regulations, vehicle dynamics, and obstacle avoidance. They conducted simulations in complex scenarios, which showed that their system could plan an optimal path in terms of vehicle dynamics while avoiding obstacles.

The work presented here has not considered long and large delay periods. PIII addresses this gap by conducting a large-scale experiment with long delay periods with a semi-autonomous mobile robot system. The system has an autonomous function based on the PF method, which is adjusted to suit RGB-D camera data, i.e., data with color and depth information. That is, the repulsive field and the attractive field are formed from the depth and color data, respectively.
3. Methods

Note that each section is devoted to each article. Thus, the mathematical symbols are independent between the sections.

3.1 Adverse weather denoising from adjacent point clouds

A supervised deep learning approach that utilizes spatiotemporal information is proposed for denoising LiDAR point clouds corrupted by adverse weather. The development of the approach started with identifying limitations in previously proposed methods and then empirically designing the architecture. The first limitation is that previous deep learning approaches do not properly utilize local density information. Local point density is a valuable indicator if a point is caused by an airborne particle, as shown by classical methods such as DROR [17], DSOR [66], and DDIOR [30]. However, the previous deep learning approaches, such as WeatherNet [67], utilize local density poorly with a standard convolution as indicated by Figure 3.1. This issue is addressed by defining the first convolution filter to capture \( k \)-nearest neighbors (KNN) in 3D space. The convolution filter is referred to as KNN-convolution, illustrated by Figure 3.1. The filter collects KNN sets for each point and encodes them, whereas a traditional 2D convolution processes the grid neighbors on the projection image. Computing KNN for multiple points is a computational burden if the search space is enormous. Therefore, only local points on the projection image are considered.

For simplicity, a pixel coordinate is denoted as

\[
\tilde{p} = (u, v).
\]  

(3.1)

The spatial-KNN-convolution becomes

\[
\Theta(\tilde{p}) = w \ast P_o^{(t)} = \sum_{\partial \tilde{p} = 0}^k w(\partial \tilde{p}) \cdot \hat{P}_o^{(t)}(\psi(\tilde{p}))(\partial \tilde{p})
\]  

(3.2)
Figure 3.1. The issue with a 2D-convolution on a spherical projection image. It can fail to capture neighbors in 3D space, whereas KNN-convolution captures local points.

where \( w \) denotes trainable weights and \( \psi(\vec{p}) \) is defined as

\[
\psi(\vec{p}) = \text{argmin}_k(|\mathbf{P}^{(i)}_r(\vec{p} - \vec{\zeta}, ..., \vec{p} + \vec{\zeta}) - \mathbf{P}^{(i)}_r(\vec{\bar{p}})|)
\]

(3.3)

where \( \text{argmin}_k(\cdot) \) returns the indicies of the minimum k elements, and \( \mathbf{P}^{(i)}_r \) denotes the range channel. \( \vec{\zeta} \) defines the number of elements considered in the KNN search. It is a hyperparameter that is tuned manually. The collected kernel inputs are activated with a \( \text{ReLU}(\theta(\vec{p})) \) function.

Another limitation of previous approaches is that they do not utilize temporal information. The points caused by airborne particles have more chaotic trajectories compared to other points [31]. This is because the airborne particles have a nearly uniform distribution and can be moved by turbulent airflow. Notably, for a single beam, it is improbable that the reflection is caused by the same airborne particle in consecutive frame pairs. Therefore, the points caused by them have different temporal characteristics than those caused by other objects, such as solid surfaces. A better model can be achieved by making this information available for the
Methods

Figure 3.2 illustrates how temporal information is captured, i.e., KNN set is searched from the previous point cloud \( \mathbf{P}^{(t-1)} \) using anchor points \( \mathbf{a} \in \mathbb{R}^{s_h \times s_w \times 3} \) from the current point cloud \( \mathbf{P}^{(t)} \). To reduce the amount of computation, the KNN sets are searched only from the neighboring area in the projection image. The temporal KNN convolution is defined as

\[
\Delta(\tilde{\mathbf{p}}) = \mathbf{w}_\Delta \ast \mathbf{d}
\]

\[
= \sum_{\partial\tilde{\mathbf{p}} = 0}^k \mathbf{w}_\Delta(\partial\tilde{\mathbf{p}}) \cdot (\mathbf{a}(\tilde{\mathbf{p}}) - \mathbf{P}^{(t-1)}(\psi_\Delta(\tilde{\mathbf{p}})))(\partial\tilde{\mathbf{p}}) \quad (3.4)
\]

where \( \psi_\Delta(\tilde{\mathbf{p}}) \) is defined as

\[
\psi_\Delta(\tilde{\mathbf{p}}) = \arg\min_k (|\mathbf{P}^{(t-1)}(\tilde{\mathbf{p}} - \tilde{\xi}, ..., \tilde{\mathbf{p}} + \tilde{\xi}) - \mathbf{P}^{(t)}(\tilde{\mathbf{p}})|). \quad (3.5)
\]

\( \mathbf{d} \in \mathbb{R}^{k \times s_h \times s_w \times 3} \) is an approximation of the motion of the local points. To convert \( \mathbf{d} \) into the magnitude and the direction of the motion, the coordinate system is switched from Cartesian to the spherical coordinates.

\[
\Gamma_d: \begin{cases} 
\mathbb{R}^3 \rightarrow \mathbb{R}^3 \\
(x, y, z) \rightarrow (r, \theta, \phi),
\end{cases} \quad (3.6)
\]

this is done before multiplying with \( \mathbf{w}_\Delta \). Similarly to the spatial KNN convolution, the kernel output is activated with a ReLU(\( \Delta(\tilde{\mathbf{p}}) \)) function.

Figure 3.2. The temporal information is captured by collecting KNN sets. Distributions given by \( \mathbf{d} = \mathbf{a} - \mathbf{k}^{(t-1)} \), in the spherical coordinate system, are smoother for a set of points with a uniform motion. On the contrary, a set of points caused by a nonuniform motion are more random.
Using the insights of the density and temporal characteristics, a function that utilizes this information should be superior. This function can be approximated by a neural network. This work uses a convolutional neural network trained with the backpropagation algorithm. Figure 3.3 illustrates this network called 4DenoiseNet, where the 4D refers to time as the fourth dimension. The feature extraction from the spatial $P(t)$ and temporal $P(t-1)$ inputs is carried out in parallel. The spatial branch computes the features of the KNN sets in the KNN-convolution block. In the temporal branch, the anchors for the KNN computation are provided by the spatial branch and subtracted from the KNN sets of the previous point cloud $P(t-1)$, from which the features are computed. The branches are fused with the motion-guided attention mechanism, inspired by video salient object segmentation [68]. A residual block further encodes these fused features. The pixel shuffle increases the spatial dimensions and reduces the channel dimension. Then, a residual block concatenates the skip connection with the main features and decodes the features. Finally, a single convolutional layer reduces the channel dimension to the desired size, then the values are normalized with the Softmax layer. The mask module $M$ uses the Softmax probabilities to remove the points predicted as noise, giving the clean point cloud $P(t)’$. 
Methods

Figure 3.3. 4DenoiseNet architecture. K, D, and BN indicate the kernel size, dilation, and batch normalization, respectively. The current point cloud and the previous point cloud are processed by the spatial and temporal branches, respectively. Then a motion-guided attention block fuses the branches. Finally, the encoded representation is decoded by two residual blocks and normalized using the Softmax.
Methods

The network is trained with a standard cross-entropy and Lovász-Softmax loss [69], which optimizes for the Jaccard index [70], i.e., intersection over union (IoU) metric. It is defined as

\[ \mathcal{L}_{ls} = \frac{1}{|C|} \sum_{c \in C} \Delta J_c(m(c)) \quad \text{where} \quad m_i(c) = \begin{cases} 1 - x_i(c), & \text{if } c = y_i(c) \\ x_i(c), & \text{otherwise} \end{cases} \]

where \( C \) denotes the number of classes, the Lovász extension of the Jaccard index is defined as \( \Delta J_c \), and the ground truth label and the prediction of pixel \( i \) for class \( c \) are denoted as \( y_i(c) \in \{0, 1\} \) and \( x_i(c) \in [0, 1] \), respectively. Finally, the total loss function is formulated as follows.

\[ \mathcal{L}_{AD} = \mathcal{L}_{ls} + \mathcal{L}_{ce} \quad (3.7) \]

where the standard cross-entropy loss is denoted as \( \mathcal{L}_{ce} \).

3.2 Self-supervised multi-echo denoising in adverse weather

In addition to the supervised approach, a self-supervised approach is proposed. By examining the previous solutions for the LiDAR point cloud denoising, it is observed that the methods are mainly supervised and rely on single-echo data. Although a self-supervised approach has been presented [16], it performs significantly worse than supervised approaches. Therefore, a self-supervised multi-echo method was developed. The first step in designing the method is to identify a stable source of supervision when labels are not available. Then empirically iterate on designs and regularization methods to achieve satisfactory performance on a benchmark dataset.

The self-supervised approach utilizes multi-echo point clouds to find the points corresponding to the environment’s obstacles, i.e., the objects of interest. In a typical LiDAR paradigm, only the strongest echo is provided. Adverse weather causes noise to LiDAR data when adverse weather-induced particles intersect the laser, causing the strongest echo. In this case, the strongest echo does not represent the object of interest. However, this problem can be avoided using multi-echo data, where the echo representing the object of interest can be recovered. Thus, the idea is to choose the echo representing the object of interest for each emitted laser pulse. Intuitively, this can be thought of as seeing through the adverse weather as other echoes, i.e., those caused by adverse weather, are removed. The concept is called multi-echo denoising and is illustrated by Figure 3.4.

This work defines the concept and proposes a self-supervised deep learning method to solve the problem. The traditional LiDAR paradigm uses single-echo point clouds \( P_s \in \mathbb{R}^{N_p \times N_c} \), where \( N_p \) and \( N_c \) denote the number
of points and channels, respectively. Whereas multi-echo point clouds are defined as follows $P_m \in \mathbb{R}^{N_p \cdot N_e \times N_c}$, where $N_e$ denotes the number of echoes per emitted laser pulse. Thus, an additional dimension is added for the multi-echo data. In this work, an ordered format is used and is defined as a spherical projection of the point cloud $\Gamma : \mathbb{R}^{N_p \cdot N_e \times N_c} \rightarrow \mathbb{R}^{H \times W \times N_e \times N_c}$, where $H$ and $W$ stand for the projection image dimensions. Echoes from the same laser pulse are stacked on dimension $N_e$ because they have the same elevation and azimuth angles. Vectors along dimension $N_e$ are denoted as echo groups $E_g$. Thus, the multi-echo ordered point cloud becomes $P_{meo} \in \mathbb{R}^{H \times W \times N_e \times N_c}$.

In multi-echo denoising, the idea is to pick the echo representing the object of interest and discard adverse-weather induced and other misleading and irrelevant points. To formulate this task, let $M(P_{nm}) = P_c$, where $P_{nm} \in \mathbb{R}^{N_p \cdot N_e \times N_c}$ is the noisy multi-echo point cloud and $P_c \in \mathbb{R}^{(N_p \cdot N_e - N_n - N_r) \times N_c}$ is the obtained clean point cloud. The number of removed noise points is denoted by $N_n$, and the number of other irrelevant points is denoted by $N_r$. The other points are, for instance, duplicate and artifact points caused by the refraction of the laser. The remaining points $P_c$ represent objects of interest, including substitutes recovered from alternative echoes. Thus, the goal is to obtain $P_c$, equivalent to a standard strongest echo point cloud in clear weather. This thesis obtains $P_c$ with a self-supervised neural network, which is called Self-supervised Multi-Echo Denoising (SMEDen). The critical component of SMEDen is the multi-echo neighbor encoder and the characteristics similarity regularisation. The multi-echo neighbor encoder is presented in Figure 3.5.
Methods

Figure 3.5. In the proposed multi-echo neighbor encoder, KNN sets are computed using multi-echo query and KNN reference, i.e., the strongest echo point cloud as it is de facto in single-echo approaches. Two example queries illustrate how KNN sets look. “Exclude self – Include neighbors” simply discards KNN queries and keeps the values. From this, the coordinate learner predicts the queries. The point cloud on the right illustrates the final output of our method, where substitutes are denoted with green.

The idea of the multi-echo neighbor encoder is to compute KNN sets with respect to the KNN reference point cloud, i.e., the strongest echo point cloud. Then, encode these sets into latent vectors with a convolutional filter. To define this more formally

$$\Pi = w_t * P_{\text{meo}} = w_t(P_r[\argmin_{\tilde{p}_2}(\psi(\tilde{p}_2, \tilde{p}_3))]$$

$$\oplus (P_{\theta\phi}[\tilde{p}_2] - P_{\theta\phi}[\argmin_{\tilde{p}_2}(\psi(\tilde{p}_2, \tilde{p}_3))])$$ (3.8)

where $w_t \in \mathbb{R}^{k \times N_e \times S_o}$ indicates trainable weights, $\tilde{p}_2 = (h, w, 0)$, $h \in [0, H]$, and $w \in [0, W]$ indicates a pixel coordinate on the strongest echo point cloud. For simplicity, components of $P_{\text{meo}}$ are denoted as: $P_{\theta\phi}$ – azimuth and elevation coordinates, $P_{xyz}$ – Cartesian coordinates, and $P_r$ – range coordinates. $\argmin_{\tilde{p}_2}()$ returns indices of $k \times N_e$ strongest echo values that minimize the Euclidian distance to multi-echo queries $\tilde{p}_3 = (h, w, \hat{e}), \hat{e} \in [0, N_e]$. $\oplus$ is the concatenation operation, and

$$\psi(\tilde{p}_2, \tilde{p}_3) = ||P_{xyz}[\tilde{p}_2 + \Delta \tilde{p}_2] - P_{xyz}[\tilde{p}_3]||_2$$

$$\Delta \tilde{p}_2 \in A_c \quad \forall \psi(\tilde{p}_2, \tilde{p}_3) < C_r$$ (3.9)

where $C_r$ is a fixed radius cutoff hyper-parameter defining the upper bound for the neighbor search, which ensures that only local points are
considered. \( \mathbf{A} \) includes the elements considered in the search. To preserve the original 3D information, the \( \mathbf{P}_{\theta \phi} \) are encoded. This is done because the grid positions of the neighbors \( \mathbf{P}_{\text{neq}} \) are lost due to the nature of the KNN search. The final step of the multi-echo neighbor encoder is to activate the encoded features with a leakyReLU function. The activated features are the input for the coordinate and correlation learner models, which are described next.

The output of the multi-echo neighbor encoder is processed by two separate deep neural networks: correlation and coordinate learner. The correlation learner learns to predict the correlation of each point with respect to its neighbors. The coordinate learner learns the coordinates of the points by using the coordinates of the KNN sets computed by the multi-echo neighbor encoder. To define the correlation learner more specifically, it learns the predictability of the coordinate of point \( \mathbf{p}_i \), which is an inverse correlation in practice. Thus, the name correlation learner. The correlation is predicted because the points caused by adverse weather typically correlate less with the neighboring points. Therefore, the predicted correlation can define which points are caused by adverse weather. For this reason, the correlation learner is used during inference to predict the noise points.

The coordinate learner learns the coordinates of hidden points by using the information of the neighboring points. The task is simplified by predicting only the range coordinate \( r_i = ||\mathbf{p}_i||_2 \in \mathbb{R} \) instead of the full 3D coordinate \( \mathbf{p}_i \in \mathbb{R}^3 \). However, the coordinate learner uses the complete 3D coordinate information for the prediction. The coordinate learner minimizes the absolute distance error to the actual coordinate of the point. To make the task of the coordinate learner non-trivial, the points that are to be predicted are masked, i.e., blind spots are added to the input data of the coordinate learner. Blind spot methods are standard in deep image denoising models [49, 71, 52]. This thesis presents a novel variant of the blind spot method: Exclude self – Include neighbors. The benefit of this method is that it uses more relevant physically neighboring points compared to the other methods where grid neighbors are used. The idea is to hide the KNN query and use the KNN values, i.e., the neighbors, to predict the hidden coordinate. The positions of the blind spots are random to avoid overfitting. In this thesis 50% of randomly selected points are masked.

To enhance the performance of SMEDen, a novel characteristics similarity regularization (CSR) is developed. The regularization increases the performance of the model and the convergence of the self-supervised learning regime. One notable merit of CSR is that it only requires one tunable hyper-parameter, the size of the KNN search \( k_{\text{CSR}} \). The noise caused by adverse weather conditions has a certain nature, which is utilized to build CSR. The points caused by the noise are assumed to 1) have lower expected intensity compared to other points [9], 2) have higher sparsity compared
to other points [17]. Following these assumptions, a process that guides
the predictions into similar distributions as intensity and sparsity should
be beneficial. The goal of CSR is to do this. However, it is essential to note
here that no specific distributions of the noise are assumed. Instead, the
model is encouraged to learn the connection between the intensity, sparsity,
and other features relative to the output.

A characteristics map is built from intensity and sparsity, and it has the
exact spatial dimensions as the input point cloud, enabling a straightforward
implementation of CSR. Before building the map, the intensity and
sparsity are normalized. An expected intensity for a LiDAR sensor follows
the inverse square law as for any point source radiation \( i \propto \frac{1}{r^2} \), where \( i \) and
\( r \) are the intensity and distance of measurement, respectively. Therefore,
the distance induces an unwanted bias. To solve this, the intensity matrix
is normalized as follows.

\[
I = I_{raw} \odot P_r^2
\]

(3.10)

where \( P_r = ||P_{xyz}||_2 \). The sparsity is normalized similarly, except inversely
and linearly, because the sparsity increases proportionally to the distance.

\[
S = E_d(P_{meo}) \odot \frac{1}{P_r}
\]

(3.11)

where \( E_d(\cdot) \) returns Euclidean distance to the nearest neighbor of each
point. It is noted here that the Euclidian distances to the neighbors are
computed already in the multi-echo neighbor encoder and thus do not
require any additional computation. The characteristics map is obtained
as a concatenation of the intensity and sparsity.

\[
\Theta = I \odot S
\]

(3.12)

The similarity is the Z-score over the normal distributions of the neigh-
boring points. For each point, neighbors are computed in \( \Theta \) using the
Euclidean distance. Then, the Z-score is computed using the distribution
of the neighbors. The absolute error between the Z-score and the output
of the correlation learner forms the regression goal for the correlation
learner. The regression goal is denoted as CSR loss \( \Xi \). This described pro-
cess forces those outputs \( O_{cor} \) with similar characteristics to have similar
values, which is expected to increase the convergence. Finally, the CSR is
summarized as the Algorithm (1).
Methods

Algorithm 1 Characteristics similarity regularization

for a batch $\epsilon$ batches do

$I \leftarrow$ Equation (3.10) $\triangleright$ Intensity
$S \leftarrow$ Equation (3.11) $\triangleright$ Sparsity
$\Theta \leftarrow \text{CONCAT}(I,S)$ $\triangleright$ Characteristics map
$\Phi \leftarrow O_{cor}[\text{ARGNEIGBORS}(\Theta,k_{CSR})]$ $\triangleright$ Characteristics
$\Xi \leftarrow |ZSCORE(O_{cor},\Phi)|$ similarity loss

end for

Now that all the components of SMEDen are described, the complete architecture and the loss function are presented. The proposed architecture is visualized in Figure 3.6. It consists of the multi-echo neighbor encoder, correlation and coordination learner networks, CSR module, and loss function. The coordinate learner has blind spots in the input, whereas the correlation learner does not.
Methods

Figure 3.6. The proposed self-supervised multi-echo denoising architecture (SMEDen). The multi-echo point cloud is first projected, and then the projected point cloud is processed by the Multi-echo Neighbor Encoder. Next, trainable points are masked in the Exclude self – Include neighbors -module. Then, the Coordinate learner processed this masked point cloud. Meanwhile, the Correlation learner processes the point cloud without masks. The loss is computed using the output of the networks and characteristics similarity regularization, which is then backpropagated through the neural networks. Finally, module $T_m$ defines the classes. White modules are used for training only.
The structure of the correlation and coordinate learner networks is visualized in Figure 3.7. They are encoder-decoder convolutional neural networks, similar to the network in PI, except that here only three residual blocks are used and the temporal branch is omitted.

As [16], the networks are trained jointly using a special loss function. Because the blind spots are added to a subset $P_s$ of points, only they contribute to the loss function. Otherwise, the coordinate learner could converge to a trivial solution. The loss function takes the form

$$
L_{me} = \frac{1}{|P_s|} \sum_{p_s \in P_s} \left( \lambda \cdot \frac{|O_{coo}^{p_s} - P_r^{p_s}|}{|P_r|_{p_s} \circ \exp(O_{cor}^{p_s})} + O_{cor}^{p_s} + \Xi_{p_s} \right)
$$

(3.13)

where $O_{cor}$ and $O_{coo}$ are the outputs of the correlation and coordinate learner, respectively. The coordinate prediction $O_{coo}$ error is minimized in the numerator of the loss function by the absolute error to the actual coordinate which is multiplied by a fixed scalar $\lambda$. In the denominator, the exponential of the correlation learner output $O_{cor}$ compensates for the coordinate prediction error. Higher compensation is needed for higher coordinate prediction errors. The key aspect here is that coordinate prediction errors are typical for adverse weather-induced points as they correlate less with their neighbors. Thus, their range prediction error is hard to
minimize without memorizing the actual values. Therefore, they require higher compensation, i.e., higher values of $O_{\text{cor}}$. The exponential is deemed beneficial experimentally. Moreover, $|P'|_p$ is used to compensate for the range bias as the correlation with neighbors is inversely proportional to the range. $|P'|_p$ is rounded to full meters to stabilize the learning process. $+O_{\text{cor}}^p$ regulates the output so that it does not explode to infinity. Finally, the CSR loss $\Xi_p$ summed up with the rest. The gradient of the loss function is backpropagated through the Correlation and Coordinate learner neural networks. Then the weights are updated based on the gradient value scaled by the learning rate.

As mentioned earlier, inference time requires only the usage of the correlation learner. However, $O_{\text{cor}}$ is a floating point number matrix, thus it has to be processed into class labels. As this work is the first to propose multi-echo denoising, the multi-echo denoising classes have to be defined. They are defined as

- valid strongest echo, $(S \land T) \rightarrow VS$,
- potential substitute, $(\neg S \land T \land B \land D) \rightarrow PS \in E_g$,
- discarded, $(\neg VS \lor \neg PS) \rightarrow DI$,

where $S, T, B, D$ are boolean formulas for "the strongest echo", "satisfies a threshold", "the best score", and "a different coordinate to strongest", respectively. The values along the $N_e$-dimension of $O_{\text{cor}}$ are denoted by the echo group $E_g$. In practice, $B$ is derived from $O_{\text{cor}}$ where the smallest value is the best. The threshold is satisfied if $O_{\text{cor}}$ is smaller than a fixed threshold. To avoid preserving duplicate points, coordinates are compared with the strongest echo-point cloud. All points labeled as $DI$ are removed during the inference time in module $T_m$ (Figure 3.6). The remaining points form the output $P_c$ of the pipeline.

Since there are no other multi-echo denoising methods in the literature, a baseline method is formulated by modifying DROR [17] to fit the multi-echo denoising task. The baseline method is called Multi-Echo Dynamic Radius Outlier Removal (MEDROR). The method computes the inlier/outlier classification for each echo separately then within each echo group, the classifications are compared. A potential substitute is found, for example, if the strongest echo is classified as an outlier and the corresponding alternative echo is classified as an inlier. The algorithm 2 describes MEDROR in detail.
Algorithm 2 Multi-Echo Dynamic Radius Outlier Removal

**Input:** Multi-echo point cloud $P_m$

**Output:** Inliers, Outliers, and Substitutes

$\alpha \leftarrow$ Angular resolution

$\beta \leftarrow$ A constant

$k_{\text{min}} \leftarrow$ Neighbor threshold

for $E_g \in P_m$ do

    for $p \in E_g$ do

        $r_p \leftarrow ||p||_2$

        $SR \leftarrow \alpha \cdot \beta \cdot r_p$

        $k \leftarrow \text{COUNTSTRONGESTECHOНЕIGHBORS}(p, SR)$

        if $k < k_{\text{min}}$ then

            Outliers.append($p$)

        else

            Inliers.append($p$)

        end if

    end for

    if $p_s$ is outlier and $\neg p_s$ is inlier then

        Substitutes.append($\neg p_s$)  \(\triangleright p_s\) is strongest echo

        and $\neg p_s$ is other than strongest echo

    end if

end for

3.3 Semi-autonomous control strategies in the presence of large delay fluctuation

Additionally to the perception issues, navigation issues were studied when external conditions caused communication issues between a teleoperator and a mobile robot. PIII uses semi-autonomous control strategies to aid teleoperators in operating a mobile robot with large delay fluctuations in the control signal. These control strategies are a combination of manual and autonomous control. Different policies are studied that decide how to switch between these two control strategies. The first policy is delay-dependent, i.e., it switches to autonomous control when the delay exceeds a threshold. This is called delay-dependent assist (DDA). The second one is control-dependent, i.e., it switches to autonomous control when the manual control disagrees with the autonomous control. Similarly, this is referred to as control-dependent assist (CDA). The control modes are summarized in Table 3.1. Both DDA and CDA use the artificial potential field (PF) method as their autonomous control. Moreover, the publication describes a novel variant of the PF method, where RGB-D data is utilized for navigation (RGB-DPF).
The general assumption for this study is that manual control is more unreliable compared to autonomous control and the focus is on the challenges of manual control. Therefore, in this study, a typical fail-safe state for autonomous control is not utilized. However, the robot has a so-called "safety zone", which is responsible for detecting close obstacles in front of the robot and the robot is stopped if an obstacle is in this zone. Extensive experiments are done with a mobile robot prototype and a teleoperation setup with a simulated control delay fluctuation. The robot used is a differential drive robot. This robot is used in a case-study environment that simulates a CERN particle accelerator’s tunnel, where delay fluctuations are one contributing challenge in teleoperation tasks [72]. This mock-up environment has a similar width as the accelerator’s tunnel and is equipped with a lane line as the accelerator’s tunnel. The lane line is detected from the color information and used in the RGB-DPF method to construct the attractive field, which describes the short-term goal for autonomous navigation. Obstacles, such as tunnel walls, are detected from depth information combined with ground plane extraction. The ground plane extraction is done using a static virtual ground plane, which represents the physical ground in the depth image space. Then, the repulsive field is formed from this. Finally, the minimum of the total field determines the preferred moving direction for the robot, which uses the pure pursuit lateral control algorithm to steer [73]. The case study environment is shown in Figure 3.8.

As described above, the system consists of several modules. The main software communication diagram illustrates how these modules are connected to each other. It is shown in Figure 3.9. Autopilot refers to the module in which autonomous control is handled. It uses the color and depth images from an RGB-D sensor to form the artificial potential field and location information from the localization node. The control manager chooses between autonomous and manual control. The manual control is on the teleoperator side, which is connected to the robot via a WiFi connection. On the side of the robot, the simulated delay is added to the control signal. The simulated delay profiles are periodic and include a noise component. The periodic signal has a wavelength of 10s. Additionally, noise is added, which is zero-mean Gaussian with different values for standard deviation. Examples of delay profiles are presented in Figure 3.10.

The periods of these profiles are long, resulting in the DDA strategy to switch the autopilot on for relatively long time periods, given that its threshold is 300 ms. This aims to study the performance of autonomous control for long time periods. The high noise results in random and high-frequency switching between the control modes, which aims to study how this affects the performance of the teleoperator. A more detailed description of the control strategies that are applied is presented in Table 3.1.
Figure 3.8. The case study environment. The blue squares denote obstacles.

Table 3.1. Description of the control strategies.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Manual control without delay fluctuation</td>
</tr>
<tr>
<td>Delayed manual</td>
<td>Manual control with delay fluctuation</td>
</tr>
<tr>
<td>Autonomous</td>
<td>Autopilot is enabled during the whole run</td>
</tr>
<tr>
<td>Delay-Dependent Assist (DDA)</td>
<td>Autopilot is enabled if the delay exceeds a threshold of 300 ms or if an obstacle is closer than 1.2 m.</td>
</tr>
<tr>
<td>Control-Dependent Assist (CDA)</td>
<td>Autopilot is enabled depending on the control signal difference or if an obstacle is closer than 1.2 m.</td>
</tr>
</tbody>
</table>

CDA and DDA use teleoperating when control or delay triggers have not activated the autopilot. The delay profiles are applied to the manual control.
Methods

Figure 3.9. The software communication diagram of the mobile robot control experiments.

commands on the CDA, DDA, and Delayed manual strategies. The manual strategy has no delay as it is used as a baseline reference. In DDA, the delay threshold is 300 ms because the control strategy typically changes to “control-and-wait” when the delay is larger than that according to [21]. Therefore, the goal of this system is to aid the teleoperator in this scenario. In CDA, the control commands of the autopilot and the teleoperator are compared, and if the difference exceeds a threshold, autopilot is enabled.

A total of 60 runs were carried out in the test environment. The test environment had obstacles to test out the robustness of the obstacle avoidance system and to make the teleoperation more challenging. The difficulty was varied by placing different amounts of obstacles into the environment. During the test runs, the trajectories and the completion times were recorded, which were analyzed to determine the characteristics and the performance of the control strategies.
Figure 3.10. The delay profiles used in the experiments.
4. Experiments

4.1 Adverse weather denoising from adjacent point clouds

The experiments related to the supervised denoising method (4DenoiseNet PI) are conducted with the semi-synthetic SnowyKITTI dataset and on the real-world Canadian Adverse Driving Conditions (CADC) dataset [74]. Quantitative experiments are conducted on the SnowyKITTI dataset as it has point-wise labels. Moreover, the 4DenoiseNet model is also trained on this dataset. Qualitative experiments are conducted with the CADC data captured in Canada during light, medium, heavy, and extreme snowfall. One of the contributions of this thesis is to test how a model trained on the semi-synthetic dataset generalizes to the real-world dataset. This thesis proposes a semi-synthetic dataset SnowyKITTI based on the physics-based simulation model presented in [12]. Specifically, synthetic noise is added to the KITTI [75] odometry benchmark LiDAR data recorded during clear weather. SnowyKITTI is divided into a train-validation-test split with a ratio of 40/10/50. It is important to note that each train, validation, and test set has separate sequences. The specialty of the training and testing procedure used here is that the training set is further divided into subsets depending on the snowfall conditions. This is done to test the generalizability of the trained model to different snowfall conditions, i.e., the model is trained only with light snowfall and tested only with heavy snowfall. These subsets are described in detail in table 4.1. They include a diverse selection of training subsets that aim to test the robustness of the model.

The models are evaluated using the intersection over union (ioU) metric, i.e., the Jaccard index [70], formulated as follows.

\[ J_c(y, x) = \frac{|\{y = c\} \cap \{x = c\}|}{|\{y = c\} \cup \{x = c\}|}. \]  

(4.1)

where \(x\) and \(y\) denote predicted and ground truth labels, respectively. This
Table 4.1. Definitions of different adverse weather conditions and training subsets.

<table>
<thead>
<tr>
<th>Classification</th>
<th>Light</th>
<th>Medium</th>
<th>Heavy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snowfall rate</td>
<td>[0.5,1.5]</td>
<td>[1.5,2.5]</td>
<td>[2.5,3.0]</td>
</tr>
<tr>
<td>Terminal velocity</td>
<td>[1.0,2.0]</td>
<td>[1.0,2.0]</td>
<td>[1.0,2.0]</td>
</tr>
</tbody>
</table>

- All ✓ ✓ ✓
- Subset 1 ✓ ✓ ✗
- Subset 2 ✓ ✗ ✓
- Subset 3 ✗ ✓ ✓
- Subset 4 ✗ ✗ ✓
- Subset 5 ✗ ✓ ✗
- Subset 6 ✓ ✗ ✗

study only measures the IoU of the noise. Thus, IoU refers to the IoU of the noise class. Other metrics such as precision and recall are not measured because IoU has been generally established as the prominent metric in the field. However, precision and recall could provide a more in-depth picture of the performance and whether the model is under or over-segmenting the point cloud.

The 4DenoiseNet model is trained with the Adam optimizer [76]. The dropout probability is set to 0.2. The learning rate is initially set to 0.01 and decayed during training with 0.01 per epoch. An L2 penalty is set to $10^4$, and the momentum is set to 0.9. The data is augmented by applying random rotation, translation, and flipping. Moreover, points are randomly dropped before creating the projection. Each augmentation is applied independently of each other with a probability of 0.5. The device used to evaluate the runtime is an Nvidia GTX 1060 GPU.

The main quantitative results of models trained with all conditions are presented in Table 4.2. The results here are achieved by training the models with the entire training set and testing with the entire testing set. Note that the performance is reported separately for light, medium, and heavy snowfall. 4DenoiseNet achieves the best performance in all of these categories. The second best method is the general-purpose semantic segmentation network SalsaNext [77]. The other method intended for adverse weather denoising, WeatherNet [67], performs approximately 10% worse than 4DenoiseNet. Cylinder3D [41] is included to show the performance difference to a voxel-based approach. Table 4.2 also reports runtime and parameter count, as they matter in the intended application where the denoising model is run as a pre-processing for other tasks. 4DenoiseNet runs approximately 7 ms faster than SalsaNext and requires 8.2 million fewer parameters than SalsaNext [77]. Compared to WeatherNet [67],
Table 4.2. Trained with all conditions. GPU: Nvidia GTX 1060, CPU: 4.0 GHz Intel i5-7600K 5th generation. * - No training required. Bolded font indicates the best values.

<table>
<thead>
<tr>
<th>Method</th>
<th>IoU</th>
<th>Runtime</th>
<th>Param.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Light</td>
<td>Medium</td>
<td>Heavy</td>
</tr>
<tr>
<td>DROR* [17]</td>
<td>0.442</td>
<td>0.445</td>
<td>0.437</td>
</tr>
<tr>
<td>LIOR* [29]</td>
<td>0.445</td>
<td>0.442</td>
<td>0.430</td>
</tr>
<tr>
<td>SalsaNext [77]</td>
<td>0.947</td>
<td>0.947</td>
<td>0.951</td>
</tr>
<tr>
<td>Cylinder3D [41]</td>
<td>0.949</td>
<td>0.943</td>
<td>0.942</td>
</tr>
<tr>
<td>WeatherNet [67]</td>
<td>0.884</td>
<td>0.889</td>
<td>0.865</td>
</tr>
</tbody>
</table>

4DenoiseNet has robust performance across all conditions. It is also consistent regardless of the training subset, and there is marginal variation in IoU compared to the other models. SalsaNext [77] is on par with 4DenoiseNet when trained with Subset2. A hypothesis is that SalsaNext [77] overfits less on Subset2 because it has a relatively high variation in snowfall conditions. Moreover, the worse performance of SalsaNext [77] on Subset5 supports this hypothesis as snowfall conditions vary less. When WeatherNet [67] and SalsaNext [77] are trained with all conditions, their performance is low. This might be caused by the weather conditions imbalance of the dataset, which causes the models to overfit to the most frequent condition. Whereas, 4DenoiseNet performs the best in Medium and Heavy snowfall when trained with all conditions, which indicates that the method is unlikely to overfit.

An ablation study is conducted to study the effects of spatial and temporal KNN convolution. In the study, variants of 4DenoiseNet were studied. The configurations of the variants are presented in Table 4.3 along with the IoU, runtime, and the number of trainable parameters. 2D indicates a traditional 2-dimensional convolution that processes the grid neighbors on the projected point cloud. The first residual block is connected straight to the second residual block, and the motion-guided attention block is omitted in the variants without the temporal branch. This study suggests that the
Figure 4.1. Quantitative results on the Light, Medium, and Heavy test sets with training subsets presented in Table 4.1. 4DenoiseNet performs well regardless of the training dataset.
Table 4.3. Ablations of convolution modules and their contribution to the performance of 4DenoiseNet. The KNN-convolution module is replaced with a traditional 2-dimensional convolution (First conv. 2D), i.e., grid neighbors convolution. ✓ indicates that previous point cloud $P_{t-1}$ and the temporal branch are used.

<table>
<thead>
<tr>
<th>First conv.</th>
<th>$P_{t-1}$</th>
<th>Light</th>
<th>Medium</th>
<th>Heavy</th>
<th>ms</th>
<th>Runtime</th>
<th>Param. $\times 10^{3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>❌</td>
<td>0.599</td>
<td>0.566</td>
<td>0.562</td>
<td>8</td>
<td>477.84</td>
<td></td>
</tr>
<tr>
<td>2D</td>
<td>✓</td>
<td>0.676</td>
<td>0.654</td>
<td>0.666</td>
<td>13</td>
<td>568.92</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>❌</td>
<td>0.882</td>
<td>0.864</td>
<td>0.866</td>
<td>16</td>
<td>480.53</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>✓</td>
<td>0.975</td>
<td>0.976</td>
<td>0.977</td>
<td>19</td>
<td>571.61</td>
<td></td>
</tr>
</tbody>
</table>

spatial KNN convolution increases the performance more than the temporal KNN convolution. However, both of them increase the performance significantly.

The fourth and final experiment with 4DenoiseNet is conducted with the CADC [74] dataset, which is real-world data in snowfall conditions. This experiment aims to study how a model trained with the semi-synthetic SnowyKITTI dataset performs with real-world data. In addition to the data domain shift, the sensor differs since CADC [74] is captured with a 32-channel LiDAR, and SnowyKITTI is captured with a 64-channel LiDAR. Hence, this experiment also studies the generalizability of other sensors. Since CADC [74] does not have point-wise labels, only qualitative experiment is conducted. The result of the experiment is presented in Figure 4.2. The focus should be directed towards the framed image pairs where the image on the left is the input and the image on the right is the output, i.e., the denoised point cloud. A coarse visual analysis indicates that denoising performs well, mostly removing the noise and leaving other points unaltered. Notably, the model has learned a high-level policy since the sparse points that are not noise are left unaltered in the first Medium pair.
Figure 4.2. Denoising performance of 4DenoiseNet on six individual sample point clouds from real snowfall. Raw input and denoised output are on the left and right sides of each image pair, respectively.

4.2 Self-supervised multi-echo denoising in adverse weather

The experiments for the self-supervised method (SMEDen PII) are shown here. First, the datasets and implementation details are specified. The Seeing Through Fog (STF) [3] dataset has multi-echo point clouds in adverse weather conditions in the form of the strongest and last echo. The dataset is collected in northern Europe with a vehicle rooftop-mounted Velodyne HDL64 LiDAR. STF [3] is a suitable dataset because it has traffic scenarios
Experiments

with a relatively high variance in adverse weather conditions. Moreover, traffic scenarios and automated driving is the intended application for SMEDen. However, it does not have point-wise labels, so it is only used for qualitative experiments. To test SMEDen quantitatively, it is converted to single-echo mode and tested with the SnowyKITTI PI dataset.

The multi-echo data on STF [3] has a conditional paradigm regarding the returned echoes. If the time-of-flight of the strongest and last echoes are equal, the strongest and second strongest echoes are returned. However, SMEDen is formulated so that more than two echoes can be used, which is beneficial since an arbitrary echo can represent the object of interest. Therefore, a separate dimension is implemented for the number of echoes and only operations that are agnostic to this dimension are used. Nonetheless, only using the strongest and last echoes can be helpful since the last echo can represent the object of interest.

The hyper-parameters for SMEDen are selected with a grid search. The initial learning rate is fixed to 0.01, and decay is set to 0.99 to avoid overfitting. The fixed hyper-parameter in Equation (3.13) $\lambda = 5$, and the CSR parameter $k_{CSR} = 9$. The model is optimized with the stochastic gradient descent optimizer with a momentum of 0.9 and trained for approximately 30 epochs or until the convergence. The STF dataset is divided into training, validation, and testing splits with a ratio of 55/10/35. Furthermore, the SnowyKITTI is divided with a ratio of 40/10/50. Moreover, all splits include various adverse weather conditions from separate sequences. The inference time threshold is set to $T_n = 0$, which was found empirically optimal. The models related to SMEDen are run on an RTX 3090 GPU using Python 3.8.10 and PyTorch 1.12.1 [78].

Table 4.4 shows the quantitative results of the self-supervised, supervised, and classical models. It should be noted here that SMEDen is converted to single-echo mode to have a fair comparison with the other models. That is, only the strongest echo is used. SMEDen is compared to SLiDE [16], which is the state-of-the-art self-supervised model, and with classical methods DROR [17] and LIOR [29]. The metric used for the evaluation is the widely adopted IoU. SMEDen achieves superior performance and thus is the new state-of-the-art in the self-supervised models. Notably, both self-supervised models perform much better compared to the classical models. Runtime and parameter count are also reported since the intended application is driving and mobile robots. Thus, the computational cost is an important metric. Moreover, denoising is often used as a pre-processing step when paired up with other models. Hence, runtime is a critical metric. SMEDen has approximately equal runtime as SLiDE [16] but around 0.7 million fewer parameters, which is beneficial regarding overfitting, memory usage, and convergence. Overall, SMEDen achieves superior performance.
Table 4.4. Single-echo results on the SnowyKITTI PI dataset. SSEDen is SMEDen in single-echo mode.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Runtime (ms)</th>
<th>IoU 0.100</th>
<th>IoU 0.109</th>
<th>IoU 0.115</th>
<th>Self-supervised improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>4DenoiseNet PI</td>
<td>Supervised</td>
<td>1.13</td>
<td>0.843</td>
<td>0.854</td>
<td>0.933</td>
<td>Self-supervised</td>
</tr>
<tr>
<td>SIIDE [16]</td>
<td>Classical</td>
<td>1.73</td>
<td>0.443</td>
<td>0.745</td>
<td>0.778</td>
<td>Self-supervised</td>
</tr>
<tr>
<td>WeatherNet [67]</td>
<td>Supervised</td>
<td>0.6</td>
<td>0.976</td>
<td>0.979</td>
<td>0.975</td>
<td>Supervised</td>
</tr>
<tr>
<td>4DenoiseNet PI</td>
<td>Supervised</td>
<td>1.5</td>
<td>0.864</td>
<td>0.884</td>
<td>0.887</td>
<td>Supervised</td>
</tr>
<tr>
<td>SLiDE [16]</td>
<td>Self-supervised</td>
<td>1.5</td>
<td>0.448</td>
<td>0.47</td>
<td>0.447</td>
<td>4DenoiseNet PI</td>
</tr>
<tr>
<td>SSEDen PI</td>
<td>Self-supervised</td>
<td>1.20</td>
<td>0.456</td>
<td>0.46</td>
<td>0.465</td>
<td>WeatherNet</td>
</tr>
<tr>
<td>SSEDen</td>
<td>Self-supervised</td>
<td>1.20</td>
<td>0.456</td>
<td>0.46</td>
<td>0.465</td>
<td>SLiDE [16]</td>
</tr>
<tr>
<td>DROR [17]</td>
<td>Self-supervised</td>
<td>1.0</td>
<td>0.439</td>
<td>0.456</td>
<td>0.458</td>
<td>SLiDE [16]</td>
</tr>
</tbody>
</table>

Similarly to 4DenoiseNet PI, SMEDen is studied via an ablation study.
Here, the ablated modules are the CSR, cut-off radius C, the neighbor encoder (NE), and the neural network encoder-decoder (NN). The ablation study is performed on the SnowyKITTI PI dataset. SSEDen is SMEDen in single-echo mode.
Table 4.5. Ablations of different modules. MH indicates multi-hypothesis prediction [16], CSR is the characteristics similarity regularization, NE is the single-echo variant of the Multi-echo neighbor encoder, and $C_r$ is the cutoff radius. NN denotes a neural network, where 4DN is 4DenoiseNet variant PI, and WN is WeatherNet variant [67]. SSEDen is SMEDen in single-echo mode.

<table>
<thead>
<tr>
<th>Method</th>
<th>MH</th>
<th>CSR</th>
<th>$C_r$</th>
<th>NE</th>
<th>NN</th>
<th>IoU Light</th>
<th>IoU Medium</th>
<th>IoU Heavy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLiDE [16]</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>WN</td>
<td>0.778</td>
<td>0.745</td>
<td>0.743</td>
</tr>
<tr>
<td>Variants</td>
<td></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>4DN</td>
<td>0.798</td>
<td>0.758</td>
<td>0.780</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>4DN</td>
<td>0.871</td>
<td>0.819</td>
<td>0.808</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>4DN</td>
<td>0.854</td>
<td>0.795</td>
<td>0.806</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4DN</td>
<td>0.885</td>
<td>0.819</td>
<td>0.792</td>
</tr>
<tr>
<td>SSEDen PII</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>4DN</td>
<td>0.933</td>
<td>0.854</td>
<td>0.843</td>
</tr>
</tbody>
</table>

study is presented in Table 4.5. The removal of CSR indicates that it contributes not only to the convergence but also to the performance of the trained model. The cutoff radius $C_r$ also contributes to the accuracy. This is reasonable because it excludes distant neighbors, which can be misleading for predicting the correlation. The neighbor search is switched from KNN to the standard 2D convolution. The KNN convolution yields significantly better results compared to the 2D convolution, which is because the grid neighbors can be relatively distant from the center point. The encoder-decoder network is switched to WeatherNet variant [67] to have a fair comparison to SLiDE [16], which uses it.

The main experiment for the SMEDen consists of qualitative analysis of multi-echo denoising performance on a real-world multi-echo dataset STF [3]. The result is shown in Figures 4.3 and 4.4, where each sample has its dedicated row. The input, i.e., the noisy point cloud, is visualized on the left-hand side, in which the fuchsia square indicates a detail window. The input images show only the strongest echo, the standard format in most LiDARs. However, the input to our method is a multi-echo point cloud. SMEDen is compared to a baseline MEDROR, visualized in the center column, where the entire image and the detail window are next to each other. Similarly, SMEDen is on the right-hand side. The baseline MEDROR is a variant of the classical DROR [17], modified for multi-echo
Denoising.

The primary purpose of SMEDen is to find substitute points from the alternative echoes, which are visualized in red. The substitute points are from the last or second strongest echo point cloud and represent the object of interest when the corresponding strongest echo point is caused by adverse weather. It can be seen from the figures that SMEDen finds substitutes effectively, which proves the concept of multi-echo denoising in adverse weather. Moreover, the results show that a simple classical method, baseline MEDROR, performs poorly because it finds only a few substitute points and many false negatives. However, baseline MEDROR performs better when noise is relatively low. Contrarily, SMEDen has the merit of performing well in all noise levels. SMEDen performs well also in extreme conditions in Sample (d).
Figure 4.3. Multi-echo denoising performance on real-world data. The corrupted strongest echo is on the leftmost column, the output of the baseline method MEDROR is in the middle, and the output of SMEDen is in the column on the right-hand side. Red indicates potential substitute points. MEDROR mostly fails, whereas SMEDen picks successfully viable substitute points. More samples are visualized in 4.4.
4.3 Semi-autonomous control strategies in the presence of large delay fluctuation

The experiments of PIII are presented here. First, the performance of the RGBD-PF is analyzed qualitatively. Figures 4.5 and 4.6 present the RGBD-
PF in special cases, where the color images are presented as they are more informative compared to the depth images. Furthermore, the potential field and its minimum are visualized to show the input of the autonomous control function. The horizontal distance between the minimum and image center, i.e., the black vertical line, is proportional to the magnitude of the steering command. The obstacle enlargement ensures that the minimum is shifted away from the obstacles.

As mentioned, Figures 4.5 and 4.6 illustrate special cases, which were used to test the robustness of the method. The chair obstacle is successfully detected as the peak in the potential field is as wide as the widest region of the obstacle. The method had difficulties detecting the semi-transparent obstacle as the peak in the potential field is non-uniform. This is caused because the camera system can see partially through the obstacle. On the left-hand side of Figure 4.6, the peaks have relatively constant height because the peak height depends on the distance to the obstacle rather than the height of the obstacle. On the right-hand side of Figure 4.6, it is illustrated that obstacles shorter than approximately 5 cm are not detectable with the configuration. This is because the obstacle is below the threshold of the potential field.

![Figure 4.5. Illustration of the potential field on a variety of obstacles. AF, RRF, RF, PF, and PF min refer to the attractive field, raw repulsive field, repulsive field, potential field, and potential field minimum, respectively.](image-url)
The behavior, performance, and limitations of the control strategies are analyzed on the test tracks. For this purpose, sample trajectories are visualized for each control strategy. Figure 4.7 shows sample trajectories of the autonomous control strategy. Here, the red dots are the location of the robot once per second. As the maximum velocity is 0.5 m/s, the maximum distance between the dots is 0.5 m. On the first track, i.e., the track without obstacles, the robot follows the route line successfully. However, due to the initial heading angle, the trajectory deviates slightly from the route line. On the tracks with obstacles, the robot avoided the obstacles successfully and reached the goal. Overall, the performance was slightly better on the tracks with fewer obstacles, and it tended to struggle more when the obstacle density was high. To its merit, no collisions happened during the test runs, indicating that the control strategy is robust and stable.
Figure 4.7. Autonomous strategy: typical trajectories on the test tracks.

Figure 4.8 presents typical trajectories of the Delayed manual strategy. The color spectrum indicates the control delay. The trajectories indicate that the steering is oscillating, which was seen on all of the tracks. The manual control strategy is beneficial, e.g., on the third track, where the robot keeps a straight trajectory after the first obstacle. It was seen that the autonomous mode returned to the route line, as it did not see obstacles further than 3 m.
Figure 4.8. Delayed manual strategy (delay profile 1): typical trajectories on the test tracks.

Figure 4.9 illustrates the typical trajectories of the delay-dependent assist (DDA) strategy. The section where the autopilot was enabled is indicated in red. Whereas the sections driven with the Delayed manual strategy are indicated with the color spectrum. A qualitative analysis of the trajectories reveals that the strategy performs well, and the autonomous mode can take over depending on the delay. The autonomous mode took over successfully in challenging conditions, for instance, when the trajectory deviated drastically from the center of the path on the fourth track at the 6m and 11 m marks.
Finally, the typical trajectories of the Control-dependent assist (CDA) mode are illustrated in Figure 4.10. Here again, the red color indicates the sections where the autonomous mode was active, and the color spectrum indicates the control delay. The oscillations with the manual control were prevented by the CDA as the autonomous mode was activated. Moreover, the autonomous mode was activated significantly more on the fourth track, where the density of the obstacles was high.
Experiments

The performance of the control strategies was further analyzed by measuring the time of completion of the tracks. The average times are presented in Figure 4.11, where the standard deviation is denoted in gray. The straight trajectory on the track without obstacles causes nearly constant completion times regardless of the control strategy. The completion times are larger on tracks with more obstacles which indicates that mission time correlates with the density of obstacles. On all tracks, the baseline manual without the delay was the quickest. Contrarily, the manual with delay was the slowest on all tracks. DDA was quicker than Autonomous on the third track but slower on the second and the fourth tracks. CDA reached almost similar performance as the baseline manual without delay on the second and third tracks. The times with delay profile 2 are relatively similar to the ones with delay profile 1. However, they are smaller on average compared to delay profile 1. Notably, DDA was slower than Autonomous on the second and fourth tracks with delay profile 1, whereas with delay profile 2, it was quicker. Based on the times on the most challenging fourth track, CDA is the best, followed by Autonomous, then DDA, with 12%, 9%, and 6% faster operation than Delayed manual strategy, respectively.
Figure 4.11. The average track completion time, the standard deviation is denoted in gray.
5. Discussion

5.1 Adverse weather denoising from adjacent point clouds

The experiments demonstrate that the deep learning method is superior compared to the classical methods. This confirms that adverse weather denoising can be performed better with a deep learning pipeline that can learn a more complex function compared to a hand-coded classical function. The deep learning method learns a latent function that accurately describes the noise caused by adverse weather. The function is latent, so its details are challenging to describe. However, the ablation studies reveal that the removal of the spatial KNN convolution reduces the accuracy substantially, which indicates that the function learned by the network utilizes the information about the $k$ nearest neighbors. Similarly, the ablation of the temporal KNN convolution reveals that the network learns temporal features, i.e., that the points caused by adverse weather are unlikely to occur in the same positions in consecutive point clouds. Nevertheless, a more in-depth nature of the learned function remains unknown, which is a drawback of deep learning approaches and poses challenges in safety-critical applications. More insight into the learned function would be gained with, e.g., visualizations of the kernel weights of the network and more ablation studies.

This work investigated how the network generalizes to a real-world dataset when trained with a semi-synthetic dataset. Furthermore, how a model trained with a high-resolution LiDAR generalizes to a low-resolution LiDAR was investigated. Overall, the network generalized well. However, the snowfall simulation method that generates the noise, simulates the intensity inaccurately. This resulted in overfitting to the intensity, which decreased the performance of the model on the real-world dataset. More specifically, the model relied too much on the intensity of the training set, and the slight difference in the intensity of the testing set caused the performance to decrease. This could be solved by modifying the snowfall
simulation method to match better the real-world snowfall or modifying the network to rely less on the intensity. A drastic solution is to omit the intensity channel altogether, which worked well in this experiment. The other challenge was to generalize to sensors with different intrinsic parameters. In this study, the resolution parameter was the main difference between the sensors. More specifically, the resolution of the LiDAR used for training was higher than that of the LiDAR used for testing. This results in a lower-density point cloud during testing. Despite this difference, the model generalized well to the lower-density point cloud, which indicates that the function learned by the network is invariant to the density, and rather is dependent on some higher-level features.

5.2 Self-supervised multi-echo denoising in adverse weather

As mentioned earlier, one of the limitations of the study is that the method is tested with a strongest/last echo point cloud. Hypothetically, the echo caused by the object of interest can be other than the strongest or last echo. Thus, more echoes would be beneficial. Therefore, the proposed method should be studied with a dataset with more echoes, for example, 1st strongest, 2nd strongest, 3rd strongest, ..., and last echo. However, such a dataset is not publicly available at the time of writing. Therefore, this remains a topic for future research.

The multi-echo experiments were only analyzed qualitatively as the STF [3] does not include point-wise labels. A future study can address this issue by analyzing the performance of the model quantitatively with a dataset with point-wise labels. However, such a dataset can be very laborious to obtain. Thus, alternative methods could also be used, such as partly or fully simulated data. One interesting quality this dataset should include is the labels for the points caused by objects of interest, i.e., solid objects the robot can collide into. However, the notation of the object of interest can be ambiguous. Therefore, a more accurate notation for such a class should be defined.

The SMEDen model has the limitation of not accurately removing extremely dense clutter, which is caused by extremely dense adverse weather. For example, dense fog causes densely cluttered points close to the sensor. It is hypothesized that this issue is caused because the clutter is so dense that the model learns some form of representation and predicts relatively high correlation values for those points. Adjustments to the hyperparameters of the model could solve this problem. However, this remains an issue for future studies to solve.

The self-supervised SMEDen model is sub-par compared to the state-of-the-art in supervised models, i.e., 4DenoiseNet PI. However, SMEDen has a better indication for generalizability, whereas 4DenoiseNet PI strug-
Discussion

gled to generalize when the input included the semi-synthetic intensity. Furthermore, due to the self-supervised loss function of SMEDen, it can be trained during inference, which means that there is the possibility to adjust to new adverse weather conditions during operation. For example, the model can be trained only with light snowfall and tested in dense snowfall, but the neural network weights can be continuously adjusted while testing. This is also a topic for future research.

The current version of SMEDen uses a single multi-echo neighbor encoder. The idea is to provide information for the encoder about all echoes. However, an alternative solution would be to process each echo individually, ensuring an identical process for all echoes. Therefore, there could be a smaller risk of learning an undesired representation. Nevertheless, testing this alternative solution remains a topic for future research.

SMEDen method performs better as it does not have to converge to multiple hypotheses as SLiDE [16]. This is because SMEDen collects the only relevant points in the multi-echo neighbor encoder. Consequently, there is no need for predicting multiple hypotheses, which also increases the convergence. However, it was found that training Correlation and Coordinate learners simultaneously was difficult, as the learning process of SMEDen is not fully stable resulting in the hyperparameters’ sensitivity. That is, the performance depends on the hyperparameter settings. To tackle this issue, the loss function or the architecture has to be revisited. One change to the architecture would be to update the weights of just one of the networks with gradient descent and copy the weights to the second network with a moving average. This is a well-proven practice in, for example, self-supervised image understanding tasks, such as DINO [79]. Changes to the loss function might also improve the stability problem. For example, some additional regularisation terms could be introduced.

5.3 Semi-autonomous control strategies in the presence of large delay fluctuation

One of the issues of the proposed RGBD-PF is that it does not detect short obstacles. The issue is visualized in Figure 4.6. This was caused because the obstacle is located between the virtual and the physical ground planes. However, the purpose of the gap is to prevent the undesired intersection of the virtual and physical ground planes, e.g., in case of uneven ground or the movement of the camera relative to the frame of the robot. Nevertheless, the implications of not detecting short obstacles can be negligible, given the maneuverability of the robot. One solution to this problem would be to calibrate the ground plane more accurately, assuming that the ground plane is flat. Moreover, the camera can be mounted more rigidly to the frame of the robot to reduce motion. The method could also be coupled
with a more sophisticated ground plane detector, which does not assume a flat ground, which would enable even outdoor use.

The test environment caused problems for the RGBD camera system, which shows that the reliability of the system depends on the conditions of the environment. These conditions would be, e.g., lighting conditions and surface materials. The effect of lighting conditions and surface materials is fairly evident for the RGB sensor. However, the surface materials can affect the depth-sensing capability as the system uses an infrared array. For example, the surface material can absorb the infrared light emitted by the sensor and prevent a proper depth measurement. Furthermore, highly reflective surfaces can distort the depth measurement. Choosing another depth sensor, such as LiDAR, could solve some of the issues of the RGBD sensor. However, LiDAR has its own problems, and the trade-off would have to be analyzed.

The performance of the RGB-DPF method was decent on all test tracks, which indicates that the system is robust. Notably, no collisions happened during testing, which indicates a reliable system. The field of view hindered the performance of the robot, as it is only 86 degrees horizontally (at a 2 m distance). This causes obstacles to be outside of the field of view and appear late, which causes issues with obstacle avoidance. This can be seen on the third track at a 9 m distance in Figure 4.7. These issues decrease the movement speed of the robot and occasionally stop the movement completely. This manifests as increased intervention time, which slows down the mission operation. Furthermore, the trajectory of the robot overshoots the goal route occasionally, e.g., on the second track at a distance of 10 m, which is suspected to be caused by the poor detection of the route line. This is a reasonable hypothesis, as the lighting in that part of the track was dim, making the detection of the route line more challenging. Nevertheless, in the well-lit regions of the test environment, the detection of the route line was reliable. The detection of the route line affects the attractive field of the RGBD-PF, which affects the route-following capability.

Both delay profiles made the manual control difficult, which caused oscillating steering. The difficulty of the control was proportional to the velocity of the robot, which indicates that assisted or autonomous control would be more beneficial with a higher velocity. As intended, the DDA strategy prevented the "control-and-wait" issue because the autonomous control activated when the delay became large. One disadvantage of the DDA strategy was that the teleoperator had difficulties predicting when autonomous control would be enabled. The main consequence of this was decreased confidence in teleoperation. The CDA proved to be a smoother strategy as the autonomous mode could assist regardless of the magnitude of the delay. This constant ability to assist proved to increase the confidence of the teleoperator. Additionally, the teleoperator could focus more on the
longitudinal control as the lateral control was assisted by the autonomous mode.

The autonomous and semi-autonomous strategies made the teleoperation smoother, which is indicated by the quicker completion times. This was apparent with both delay profiles. However, the density of the obstacles affected the completion times, indicated by short times on the empty track and longer times on the tracks with obstacles. Moreover, the differences between track completion times were higher when the density of obstacles was higher. The control strategy did not affect the completion time in the empty track. Furthermore, the track was only 15 m long. Thus, the time differences were minor and would be more apparent with longer tracks. Therefore, to better understand the performance of the strategies, longer experiments should be conducted. The autonomous mode spends time maneuvering on the center of the track between obstacles on the third track, even though it could have gone straight. Given the obstacle arrangement, the mode is more beneficial on the second track because the robot has to move on the other side of the route line.

The completion times were generally quicker with the delay profile two because the rate of low-delay commands was higher. This resulted in better control over the robot. For the DDA strategy, the autonomous mode was activated more frequently due to the nature of the delay profile, which also resulted in better control. The autonomous strategy was quicker than the manual strategy on the fourth track, with many obstacles. Therefore, autonomous navigation could be feasible for long-term navigation in repetitive environments where long-term path planning is not critical. The intended accelerator tunnel environment is repetitive, relatively simple, and structured. Thus, the presented system can be feasible. However, an accurate localization system would be required. The benefit of the presented system is that it uses only a modest amount of computation and memory, as maps or long-term planning are not used.
6. Conclusion

This thesis discussed problems related to autonomous mobile robots in adverse conditions. Adverse conditions in this thesis refer to adverse weather and limited communication between the vehicle and the remote operator. The effects of adverse weather on LiDAR were studied as it is a popular sensor in autonomous vehicles. Communication issues can be a problem in multiple autonomous vehicle applications. In this thesis, the focus was on a case study of a mobile robot operating in the CERN accelerator tunnel.

Adverse weather causes noise to the LiDAR point clouds. To address this issue, two deep learning-based denoising approaches were presented. 4DenoiseNet PI was trained with a semi-synthetic dataset and tested with a real-world dataset. Unlike previous methods, it utilizes a spatiotemporal KNN convolution layer for the input for the CNN. The second method, SMEDen PII, is trained in a self-supervised manner and takes advantage of multi-echo point clouds. It is fundamentally different from previous approaches, which were single echo methods. This method can utilize information from multi-echo point clouds that is unavailable to the single echo approaches and find the obstacle locations in adverse weather.

4DenoiseNet performed better than the previous best method in the literature. The most notable result was that the performance was much better than the performance of the classical hand-coded algorithms. This shows that the deep learning method learned a superior function compared to the classical ones. Compared to other deep learning methods, 4DenoiseNet was more robust and consistently performed across multiple train-test splits. 4DenoiseNet utilizes spatiotemporal information more effectively compared to the other methods, which explains the superior performance. The method also generalized well to a separate real-world dataset with different sensor intrinsics. However, this was achieved by omitting the intensity information as it was subject to overfitting. This remains an issue for future work to address.

SMEDen was trained in a self-supervised manner, i.e., without any labels. However, it was still able to perform better than the classical methods.
It almost reached a similar performance as the supervised method 4De-noiseNet. By predicting the coordinates of hidden points and correlations of the points, SMEDen was able to learn a function that describes the noise accurately. The key aspect of the method is that it predicts the echo caused by the object of interest from multi-echo point clouds instead of merely removing the noise points as the single-echo approaches. However, in order to compare its performance to other methods, it was converted to a single-echo mode. The quantitative analysis revealed that the method performed better than the previous best self-supervised method in the literature. The multi-echo performance was analyzed with a hand-coded baseline MEDROR qualitatively, which showed that the self-supervised method was superior. The main limitation of the SMEDen study is that it was tested only with dual-echo point clouds. Tests with more echoes should be therefore conducted.

External conditions can cause communication issues between a teleoperator and a mobile robot. The thesis addressed a delay issue between the robot and the teleoperator in long and narrow tunnels where the internet communication is imperfect. Limited communication causes delays and fluctuations in the communication signal. Long and large delay periods were applied, and semi-autonomous control strategies were tested to alleviate the control issues. The strategies were delay and control-dependent, and both of them improved the navigation performance under the delay. Nevertheless, the study was conducted in a well-controlled environment, and more elaborate experiments remain for future research.

To realize the potential of fully autonomous vehicles in complex human environments, research is required in the area of adverse conditions. This thesis addressed some of those conditions. However, the space of possible conditions and edge cases is large, which attracts the idea of more unified solutions, such as end-to-end neural networks that process sensor data directly to control commands, which could potentially be robust in various edge cases.
References


References


References


