A STUDY ON DEPTH ESTIMATION AND DIGITAL TERRAIN MODEL RECONSTRUCTION FOR MOBILE MAPPING SYSTEMS

AMIN ALIZADEH NAEINI

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN EARTH AND SPACE SCIENCE

YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2024

© Amin Alizadeh Naeini, 2024

Abstract

Mobile Mapping Systems (MMSs) represent advanced geospatial data collection platforms that integrate cutting-edge technologies such as Light Detection and Ranging (LiDAR) and digital imaging to capture highly precise environmental information. These systems comprise two main components: georeferencing and mapping.

Georeferencing aligns digital data with coordinate systems using Global Navigation Satellite Systems (GNSS), achieving high accuracy. However, in GNSS-denied environments, Simultaneous Localization and Mapping (SLAM) technology is employed to map unknown environments and track MMS locations. The accuracy and density of depth maps are vital for effective SLAM performance, influencing the system's ability to construct environmental maps and track positions. To address challenges in depth estimation within MMSs, this dissertation introduces the Double-stage Adaptive Refinement Scheme (DARS), designed to enhance depth estimation in dynamic environments. DARS can be integrated with pre-trained networks and extended to Panoramic DARS (PanoDARS) for SLAM applications, where the use of panoramic images is prevalent. PanoDARS improves the performance of the HDPV-SLAM system by addressing LiDAR depth sparsity and depth association issues.

This thesis also explores the reconstruction of Digital Terrain Models (DTMs) from Digital Surface Models (DSMs), a critical component in accurate mapping within MMSs. DTMs provide essential elevation data that form the foundation for producing various geospatial products, including true orthophotos, photomaps, topographic maps, hydrological maps, and 3D urban models. To achieve accurate DTM reconstruction, a geospatially induced autoencoder named SB-SUBNET is proposed. This autoencoder incorporates two geospatial

biases, termed geospatial inductive biases, which leverage the relationship that DTM = DSM - nDSM. The first bias involves indirect DTM reconstruction by subtracting the network's output (nDSM) from its input (DSM). The second bias is a subtractive skip connection that directly integrates this geospatial information into the network. These biases enhance the autoencoder's performance by effectively utilizing the inherent geospatial relationships within the data, leading to more accurate DTM reconstruction.

Although SB-SUBNET shows promising results, this approach alters all DSM values, including those of the terrain, affecting DTM accuracy. To address this issue, a new multi-task learning approach, named DB-SUBNET, was developed. This approach focuses on segmenting non-ground points and performing regression specifically for these points while preserving ground points. This method not only improves the accuracy of DTM reconstruction but also ensures that the resulting DTMs provide a reliable foundation for generating high-precision geospatial products. This enhanced DTM reconstruction process is vital for creating detailed and accurate maps, supporting various applications such as urban planning, infrastructure development, environmental monitoring, and disaster management.

March 15^{th}, 2024

This thesis is dedicated to my beloved family: my wife, ***Mina****, and my children,* ***Parham*** *and* ***Afra****. Their unwavering support and patience through numerous challenges have been the cornerstone of my perseverance and success.*

Acknowledgements

I would like to express my heartfelt gratitude to all those who have supported and contributed to completing my Ph.D. thesis.

First and foremost, I would like to extend my most profound appreciation to my supervisor, **Professor Gunho Sohn** of the Earth & Space Science Department at York University, for his guidance, mentorship, and expertise. His dedication, patience, and continuous support have been instrumental in shaping the direction and quality of my research. I am grateful for his encouragement, feedback, and valuable insights that have enriched my work.

I am also honoured to thank my committee members, **Professor Costas Armenakis** and **Professor Baoxin Hu** of the Earth & Space Science Department at York University, for their dedicated time and effort in examining my thesis and providing insightful comments.

I want to thank **Mohammad Moein Sheikholeslami** from the AUSM laboratory at York University, who has always been there for me both academically and personally whenever I needed his help. I am also grateful to **Dr. Mohammad Koushfar** from the Air Pollution Lab at York University, for his invaluable support throughout my research journey and **MohammadJavad Ghorbanlivakili**, **Mostafa Ahmadi**, and **Muhammad Kamran**, from the AUSM laboratory at York University, who willingly contributed their precious time, expertise, and insights.

Lastly, I am grateful to my family for their unwavering love, support, and encouragement throughout my Ph.D. journey. Their belief in me and their constant motivation have been a driving force in my pursuit of excellence. I am grateful for their sacrifices, understanding, and encouragement during the ups and downs of this journey.

Table of Contents

Abstract **ii**

Acknowledgements **v**

Table of Contents **vi**

List of Tables **xi**

List of Figures **xiii**

Abbreviations **xviii**

1 Introduction **1**

1.1 Depth Estimation 4

1.1.1 Image-based Approach 5

1.1.2 LiDAR-guided Approach 6

1.2 Depth Completion: Bridging the Gaps in Depth Data 7

1.2.1 Guided/Informed Depth Completion Methods 7

1.2.2 Blind Depth Completion Methods 8

1.3 Goals and Objectives 8

1.3.1 Addressing the Generalization Problem in DL-based Depth Estimation 9

1.3.2 Adapting Depth Estimation Methods for Panoramic Images 9

1.3.3 Advancing DTM Reconstruction Through a Geospatially Informed Single Task Network . . . 9
1.3.4 Advancing DTM Reconstruction Through a geospatially Informed Multi Task Network . . . 10
1.4 Research Contributions . . . 10
1.4.1 Enhancements to Depth Estimation Techniques . . . 10
1.4.2 Enhanced Depth Estimation for Panoramic Images . . . 11
1.4.3 Enhanced DTM Reconstruction as a Regression Problem . . . 11
1.4.4 Advanced Multi-Task Learning for DTM Reconstruction . . . 11
1.4.5 Contributions to Safe and Autonomous Navigation Technologies . . . 12
1.5 Thesis Overview . . . 13
1.6 Summary . . . 14

2 Background **16**
2.1 Introduction . . . 16
2.2 Optimization . . . 16
2.2.1 Constraints . . . 19
2.2.2 Inherent Nature . . . 19
2.2.3 Search Strategy . . . 20
2.3 Deep Learning . . . 24
2.3.1 Mathematical Parameters . . . 24
2.3.2 Layers . . . 25
2.3.3 Activation Function . . . 30
2.3.4 Architectural Frameworks . . . 33
2.3.5 Training Components . . . 36
2.3.6 Optimization Methods . . . 37
2.3.7 Training Approaches . . . 44
2.3.8 Domain Adaptation . . . 52
2.4 Depth Estimation . . . 55

2.4.1 Supervised Learning Methods 55
2.4.2 Unsupervised Learning Methods 57
2.4.3 Semi-supervised Learning Methods 58
2.4.4 Tasks 59
2.4.5 Interactive Refinement 60
2.5 Deterministic Interpolation Methods 61
2.5.1 Triangulation Methods 61
2.5.2 Gridding Methods 63
2.5.3 Geostatistical Interpolation Methods 64
2.6 Quality Assessment Criteria 66
2.6.1 Predictive Error Metrics 66
2.6.2 Classification Accuracy Metrics 68
2.6.3 Profile Analysis Metrics 69
2.6.4 SLAM Evaluation Metrics 70
2.7 Summary 71

3 Model Agnostic Depth Estimation **72**
3.1 Introduction 72
3.2 RELATED WORK 75
3.2.1 Unsupervised Depth Estimation Methods 75
3.2.2 Supervised Depth Estimation Methods 76
3.2.3 Functionally Adaptive Neural Networks 78
3.3 Operational Framework 80
3.3.1 f-BRS 81
3.3.2 PSO-oriented f-BRS 81
3.3.3 Delaunay-based Interpolation (DBI) 82
3.4 PROPOSED METHOD 82
3.4.1 DARS 82
3.4.2 PanoDARS 85

3.5 EXPERIMENTS . . . 89
3.5.1 Datasets . . . 89
3.5.2 Discussion and Results . . . 94
3.6 Summary . . . 104

4 Geospatially informed Single Task Autoencoder for DTM Reconstruction 106
4.1 Introduction . . . 106
4.2 Related work . . . 110
4.3 Methodology . . . 112
4.3.1 Problem Formulation . . . 112
4.3.2 Regression Task . . . 112
4.4 Experiments . . . 115
4.4.1 Characteristics of Datasets . . . 115
4.4.2 Experimental Setup . . . 134
4.4.3 Ablation Study . . . 135
4.4.4 Discussion . . . 136
4.4.5 Summary . . . 142

5 Geospatially informed Multi Task Autoencoder for DTM Reconstruction 144
5.1 Introduction . . . 144
5.2 Related Work . . . 149
5.3 Methodology . . . 152
5.3.1 Problem formulation . . . 152
5.3.2 Regression Task . . . 153
5.3.3 Segmentation Task . . . 155
5.3.4 Multi Task Learning . . . 157
5.4 Experimental Result . . . 158
5.4.1 Quality Assessment Criteria . . . 158
5.4.2 Experimental Setup . . . 158

5.4.3 Competing Methods 159
5.4.4 Results and Discussion 160
5.5 Summary 187

6 Conclusion and future direction **189**
6.1 Conclusion 189
6.2 Future studies 192

Bibilography **194**

Appendices **218**
A.3 Mapping and Depth Estimation in Panoramic Imaging 218
A.3.1 Mappings Between 3D Points and Pixels in the Panoramic Image . . 218
A.3.2 Depth Estimation Module 221
A.4 ORB Feature Detection and Description 223
A.5 HDPV-SLAM Overview 224

List of Tables

3.1 Ablation study for KITTI. 95
3.2 Ablation study for NYUv2. 96
3.3 Comparative study for KITTI. 99
3.4 Comparative study for NYUv2. 99
3.5 RMSE (m) of the two depth estimation methods on LiDAR points 101
3.6 Keyframe statistics 102

4.1 Structural characteristics of USGS dataset, where each subset comprises three semantic classes: non-ground, ground, and noise. 115
4.2 Quantitative analysis of statistical features in the USGS dataset (unit: meter). 116
4.3 Structural characteristics of OpenGF dataset, where each subset comprises three semantic classes: non-ground, ground, and noise. 123
4.4 Statistical features in the three regions of OpenGF dataset (unit: meter). . . 123
4.5 Structural characteristics of ALS2DTM dataset. 127
4.6 Semantic information of DALES and NB subsets. 127
4.7 Statistical analyses of DALES and NB subsets (unit: meter). 128
4.8 Summary of ablation studies. 136
4.9 Comparative performance analysis of the proposed method and its leading competitor, CNETI, across statistical and predictive metrics (unit: meter). . 139

5.1 Summary of intra-domain experiment, where the ratio of training to testing is 80 to 20. 161

5.2 Comparative analysis of different elements of the proposed method. 163
5.3 Evaluation of the proposed methods and their rivals on KAII. 164
5.4 Evaluation of the proposed methods and their rivals on SUII. 164
5.5 Evaluation of SB-SUBNET over seven unseen data. 172
5.6 Evaluation of DB-SUBNET over seven unseen data. 172
5.7 Evaluation of pix-to-pix over seven unseen data. 172
5.8 Evaluation of HDCNN over seven unseen data. 173
5.9 Evaluation of EyeNET over seven unseen data. 173
5.10 Evaluation of RandLA over seven unseen data. 173

List of Figures

2.1 Performance of Gradient Descent Method. The x-axis represents the weight parameter w_1, the y-axis represents the weight parameter w_2, and the z-axis represents the error function $E(w_1, w_2)$. The surface plot illustrates the error landscape of the function $E(w_1, w_2)$, and the red dots indicate the steps taken by the gradient descent algorithm. The path traced by the red dots shows the algorithm's progression toward the minimum error, demonstrating the effectiveness of gradient descent in minimizing a two-dimensional error function. 39

2.2 Performance of Gradient Descent for Different Values of Learning Rate (η) . 40

2.3 Comparison of gradient descent methods on an error function. The diagram on the left demonstrates traditional gradient descent, while the diagram on the right illustrates gradient descent with momentum, showing enhanced convergence [54]. 42

3.1 Overall performance of the proposed scheme. Keeping the pre-trained weights fixed, channel-wise scales and biases are applied to an intermediate activation map and are optimized to improve the predictions for the input dataset. The input dataset can be identical to the training dataset or a new one. 74

3.2 Delaunay-based interpolation on a set of points. First, a Delaunay triangulation is done on the points. Then, a plane is fitted to each triangle, and finally, the value for points on each of them is obtained based on the fitted plane. 82

3.3 The double-stage adaptive refinement scheme (DARS). DARS refines depth maps estimated by baselines through optimizing (green arrows) an intermediate set of feature maps during inference (red arrows). Hence, given a pretrained baseline with fixed weights and an RGB input image, an initial depth D_0^d is estimated. In the first stage (correction), a Sliced Delaunay Correction (SDC) module corrects D_0^d with the guidance of an SD map D^s. Afterward, an optimizer module (PSO) tries to update the intermediate feature maps to minimize the distance between the dense corrected depth $\hat{D}^d$ and D^s. As the scheme can be used for any deep architecture, the baseline (here, monodepth2) is illustrated minimally. 83
3.4 Distance-based correction (DBC). Each SD map is first divided into three slices with overlap, and then the correction value (ΔD) is interpolated using DBI in each of them independently. In the overlapped areas, the average of the values coming from the two slices is taken into account. 86
3.5 PanoDARS schematic architecture. 86
3.6 Depth association hypothetical scenario 88
3.7 A sample of used datasets. 91
3.8 Color textured point-cloud of YUTO MMS 94
3.9 Visual results related to ablation study of KITTI dataset. Numbers on the right side of error patterns are in meters. 97
3.10 Visual results related to ablation study of NYUv2 dataset. Numbers on the right side of error patterns are in meters. 98
3.11 Depth estimation module qualitative evaluation 100
3.12 Comparison of ATE 102
3.13 Comparison of RTE 103
3.14 Comparison of RRE 104

4.1 Architecture of the proposed method. 113
4.2 Detailed architecture of the regression task. 114

4.3 illustration of statistical features in the USGS dataset. 116
4.4 Normal distribution fits for USGS. 119
4.5 Topographic detail of SU region samples. 120
4.6 Topographic detail of a KA region sample. 121
4.7 Topographic detail of an RT region sample. 121
4.8 Topographic detail of a KW region sample. 122
4.9 Qualitative illustration of statistical features in three regions of OpenGF dataset. 124
4.10 Normal distribution fits for three regions of the OpenGF dataset. 125
4.11 Topographic detail of three samples related to OpenGF. 126
4.12 Normal distribution fits for ALS2DTM. 129
4.13 Topographic detail of two samples related to ALS2DTM dataset. 130
4.14 DSMs' distribution of USGS and ALS2DTM datasets. 131
4.15 DTMs' distribution of USGS and ALS2DTM datasets. 132
4.16 Comparison of DSM and DTM values across datasets. 132
4.17 Difference between DSM and DTM values across datasets. 133
4.18 Distribution of RT and KW Regions. 133
4.19 Comparative study of different versions of autoencoders, along with the proposed method, in terms of statistical and predictive metrics. 137
4.20 Visual inspection of the proposed method and its leading competitor, CNETI, focused on RMSE. 141
4.21 Visual inspection of the proposed method and its leading competitor, CNETI, focused on maximum error. 142

5.1 A scheme of three different surface models. 146
5.2 Architecture of the proposed method. 152
5.3 Detailed architecture of the regression task. 154
5.4 Detailed architecture of the segmentation task 156
5.5 Methods' stability: left - SUII, right - KAII. 165

5.6 DTM generation results for KAII: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 167
5.7 Cross-sections of each method for a KAII sample. 168
5.8 DTM generation results for SUII: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 169
5.9 Cross-sections oF each method for an SUII sample. 170
5.10 DTM generation results for DALES: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 174
5.11 Cross-sections of each method for a DALES sample. 175
5.12 DTM generation results for S4: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 176
5.13 DTM generation results for S8: (A) Input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 177
5.14 Cross-sections of each method for a S4 sample. 178
5.15 Cross-sections of each method for a S8 sample. 178
5.16 DTM generation results for S9: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 179
5.17 DTM generation results for NB: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 180
5.18 DTM generation results for KW: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 181
5.19 Cross-sections of each method for an S9 sample. 182
5.20 Cross-sections of each method for a NB sample. 182
5.21 Cross-sections of each method for a KW sample. 183
5.22 DTM generation results for RT: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN. 184
5.23 Cross-sections of each method for an RT sample. 185
5.24 Cross-sectional analysis for classification-based methods versus DB-SUBNET. 186

6.1 Various coordinate systems representation [156]. 219
6.2 LiDAR points & projection on panoramic image 221

Abbreviations

AD: Annotated Depth

AdaBins: AdaptiveBins

Adagrad: Adaptive Gradient Algorithm

Adam: Adaptive Moment Estimation

AE: Autoencoders

AI: Artificial Intelligence

ALS: Airborne Laser Scanning

ATE: Absolute Trajectory Error

BFGS: Fletcher–Goldfarb–Shanno

BRS: Bachpropagation Refinement Scheme

CNETD: Concatenative NETwork with Direct DTM regression

CNETI: Concatenative NETwork with Indirect DTM regression

CGAN: Generative Adversarial Network

CGI: Computer Generated Imagery

CNNs: Convolutional Neural Networks

CSF: Cloth Simulation Filter

DARS: Depth Adaptive Refinement Scheme

DB-SUBNET: Double Branch Network with Subtractive skip connections

DBC: Distance Based Correction

DBI: Delaunay-Based Interpolation

DEMs: Digital Elevation Models

DL: Deep Learning

DTM: Digital Terrain Model

DSM: Digital Surface Model

ELU: Exponential Linear Unit

EMD: Earth Mover's Distance

F-BRS: Feature-based Backpropagating Refinement Scheme

FN: False Negative

FP: False Positive

FNNs: Feedforward Neural Networks

GANs: Generative Adversarial Networks

GIS: Geographic Information System

GPS: Globa Positioning System

GRUs: Gated Recurrent Units

HDCNN: Hybrid Deep Convolutional Neural Network

IDW: Inverse Distance Weighting

IMU: Inertial Measurement Unit

INS: Inertial Measurement Units

IoU: Intersection Over Union

KL: Kullback-Leibler

L-BFGS: Limited-memory BFGS

LiDAR: Light Detection and Ranging

LSTMs: Long Short-Term Memory Networks

LP: Linear Programming

MAE: Mean Absolute Error

MMS: Mobile Mapping System

MSE: Mean Squared Error

MTL: Multi-Task Learning

MVS: Multi-view stereo

nDSM: Normalized Digital Surface Model

NLP: Non-linear Programming

PanoDARS: Panoramic Depth Adaptive Refinement Scheme

PReLU: Parametric Rectified Linear Unit

PSoftplus: Parametric Softplus

PSO: Particle Swarm Optimization

RandLA-Net: Random Point Sampling and Labeling

ReLU: Rectified Linear Unit

ResNets: Residual Networks

RMSprop: Root Mean Square Propagation

RNNs: Recurrent Neural Networks

RRE: Relative Rotation Error

RTE: Relative Trajectory Error

RTK: Real Time Kinematics

RPV-SLAM: Range-augmented Panoramic Visual Slam

SAM: Segment Anything Model

SB-SUBNET: Single Branch Network with Subtractive skip connections

SD: Sparse Depth

SUBNETD: SUBtractive NETwork with Direct DTM regression

SUBNETI: SUBtractive NETwork with Indirect DTM regression

SGD: Stochastic Gradient Descent

SLAM: Simultaneous Localization and Mapping

TAAF: Transformative Adaptive Activation Function.

TAE: Traditional/Typical Autoencoder

TAEI: TAE with Indirect DTM regression

TAED: TAE with Direct DTM regression

Tanh: Hyperbolic Tangent

TIN: Triangulated Irregular Networks

TKCGAN: Topographic Knowledge-Constrained Conditional GAN

TN: True Negative

TP: True Positive

UDA: Unsupervised Domain Adaptation

USGS: United States Geological Survey

VAE: Variational Autoencoders

WGAN: Wasserstein Generative Adversarial Network

YUTO: York University Teledyne Optech

ZSL: Zero-Shot Learning

Chapter 1

Introduction

This thesis explores the dynamic field of computer vision with a focus on depth generation. Depth generation encompasses two critical components: depth estimation and depth completion. These components are vital in enhancing the capabilities and efficiency of numerous computer vision applications [1].

Depth estimation involves computing the depth map of a given scene, which is crucial for 3D reconstruction and scene understanding. It provides the foundational data needed to perceive the spatial structure of the environment [2]. Depth completion, on the other hand, aims to generate a complete and accurate depth map by filling in missing values in a partially observed depth map, such as a digital terrain model (DTM). This process is particularly significant for ensuring the reliability and accuracy of depth data in practical applications [3].

The importance of depth generation is underscored by its extensive real-world applications, especially in Mobile Mapping Systems (MMSs). MMSs are geospatial data collection platforms that integrate various technologies to capture highly accurate location-based information. These systems rely on two primary components: georeferencing and mapping, both of which are essential for accurately representing spatial data [4].

Georeferencing in mobile mapping is the process of aligning digital data with a coordinate system, ensuring that it can be viewed and analyzed alongside other geographic data. Achieving this with an accuracy of up to one centimeter is made possible by the Global

Navigation Satellite System (GNSS), which provides global coverage and precise location data [5]. However, in environments where GNSS signals are unavailable or unreliable, such as tunnels or densely built-up areas, georeferencing becomes challenging. To mitigate this, Simultaneous Localization and Mapping (SLAM) technology is employed. SLAM enables the mapping of an unknown environment while concurrently tracking the location of the mobile mapping system within it [6].

The effectiveness of SLAM is fundamentally dependent on the accuracy and density of the depth maps generated, as these factors directly influence the algorithm's capability to concurrently construct an environmental map and track the mobile platform's position. High-accuracy depth maps provide precise spatial measurements of environmental features, essential for minimizing localization drift and ensuring that the SLAM system can accurately estimate its position over time [2]. For example, in an urban canyon where GNSS signals are often unreliable, accurate depth measurements allow the SLAM system to correctly identify and map buildings and other structures, maintaining precise localization [7]. High-density depth maps contribute to the detailed representation of the environment, enabling the detection of subtle structural nuances and providing robust data for feature extraction and matching [8]. This data richness is particularly crucial in GNSS-denied environments, where the absence of satellite-based positional information necessitates a greater reliance on sensor-derived data [9]. For instance, in a densely forested area, high-density depth maps allow the SLAM system to accurately identify and navigate around trees and other obstacles, maintaining a reliable map and navigation path [10]. Therefore, integrating advanced depth estimation techniques within MMSs is imperative for achieving high-precision georeferencing in challenging environments [3].

Digital Terrain Models (DTMs) are fundamental components in the field of mapping within mobile mapping systems, providing essential elevation data that underpins the creation of a wide range of precise geospatial products. These products include, but are not limited to, true orthophotos, photomaps, topographic maps, hydrological maps, and 3D urban models, all of which rely on the detailed and accurate representation of the terrain provided by DTMs

[11].

True orthophotos are one of the primary products generated using DTMs. These are aerial photographs that have been geometrically corrected to have a uniform scale, free from distortions caused by terrain relief and camera tilt [12]. The accurate elevation data from DTMs is crucial in this process as it allows for the rectification of these distortions, ensuring that the orthophotos can be used as reliable maps with precise spatial relationships. This is particularly important in urban planning, infrastructure development, and environmental monitoring, where accurate spatial data is essential for decision-making [13].

Photomaps, which are composite images created from multiple overlapping aerial photographs, also heavily rely on DTMs. The DTM aids in correcting for terrain-induced parallax and elevation differences, facilitating the seamless stitching of individual photographs into a coherent and accurate photomap [14]. These maps are invaluable for large-scale mapping projects, topographic analysis, and detailed landscape assessments, providing a detailed visual representation of the terrain and surface features [15].

Topographic maps, which depict the natural and man-made features of an area, including its terrain relief, also utilize DTMs. The accurate elevation data from DTMs allows for the generation of contour lines and other topographic features, providing a detailed representation of the terrain [16]. These maps are essential for various applications, including hiking, urban planning, and military operations, where an accurate understanding of the terrain is crucial [17].

Hydrological maps, which illustrate the features of water bodies and the flow of water over the land, are another important application of DTMs. The elevation data from DTMs is used to model water flow and drainage patterns, identify watershed boundaries, and predict areas susceptible to flooding [18]. Accurate DTMs are essential for flood risk assessment, water resource management, and environmental conservation efforts [19].

3D urban models, which provide a three-dimensional representation of urban environments, also benefit significantly from DTMs. The elevation data from DTMs is used to accurately model the terrain, ensuring that buildings and other structures are placed correctly relative to

the ground surface [20]. These models are used in urban planning, infrastructure development, and disaster management, providing a detailed and accurate representation of the urban environment [21].

The process of generating these mapping products involves the integration of data from various sensors, such as LiDAR (Light Detection and Ranging) and photogrammetry, and advanced data processing techniques [22]. LiDAR provides high-resolution elevation data by measuring the time it takes for laser pulses to return from the ground, while photogrammetry uses images captured from multiple angles to create detailed 3D models [23]. The data from these sensors is processed under a category named DSM filtering to create DTMs, which serve as the foundation for the various mapping products [24].

In summary, DTMs are integral to the creation of true orthophotos, photomaps, topographic maps, hydrological maps, and 3D urban models, all of which are critical components of MMSs [25]. The detailed and accurate elevation data provided by DTMs ensures that these mapping products are reliable and can be confidently used for various geospatial applications. The high fidelity of DTMs is essential for correcting image distortions, ensuring precise spatial relationships, and supporting detailed terrain analysis, underscoring their importance in the field of digital mapping [25].

After understanding the role of depth estimation and depth completion, particularly emphasizing their application in MMSs, the subsequent sections will introduce the background of the research subjects being studied. These sections will also explain the challenges that this thesis aims to address, providing a comprehensive foundation for the research conducted.

1.1 Depth Estimation

Depth estimation, as comprehensively discussed in 3.1, represents a fundamental challenge in computer vision, involving the process of determining the depth of various elements within a scene from images. This task, essential for numerous applications, is broadly categorized into two main approaches: image-based methods and LiDAR-guided methods, each with unique

methodologies and applications [26].

1.1.1 Image-based Approach

Image-based methods, a traditional approach to depth estimation, are further divided into monocular and stereo methods. Monocular depth estimation, utilizing a single image, is particularly challenging due to the lack of spatial information that is readily available in multiple-image techniques [27]. Despite this, recent advances in machine learning and computer vision have led to significant progress in monocular depth estimation. These techniques enable the extraction of depth information from a single viewpoint, making them particularly useful in situations where only one camera is available or feasible, such as mobile devices or certain types of drones. However, supervised learning methods face challenges associated with their reliance on training data, which may not always be readily available. Additionally, these methods generate depth estimates that are up-to-scale, posing an important problem for their practical application [27].

To overcome this limitation, an image-based approach may be adopted that makes use of stereo images or video sequences, providing a self-supervised alternative [27]. This approach does not require labeled data and, thus, can be useful in scenarios where such data is scarce or non-existent. This self-supervised methodology not only alleviates the need for labeled datasets but also enhances the system's ability to generalize across various scenarios. However, the application of stereo-image-based depth estimation itself presents several notable challenges. The reliance on a stereo camera setup imposes constraints due to the necessity for precise calibration and synchronization, critical factors that, if not meticulously managed, can significantly undermine depth accuracy. Furthermore, stereo vision systems must contend with occlusions and disparities, which can introduce depth estimation errors, particularly detrimental in environments where robust spatial understanding is crucial. The issue of generalization is also paramount due to the dynamic nature of MMSs.

With the advent of LiDAR-equipped MMSs, a viable solution to the aforementioned problems has emerged. To this end, LiDAR-guided depth estimation techniques have been

developed.

1.1.2 LiDAR-guided Approach

LiDAR technology has revolutionized the field of monocular depth estimation, providing a critical benchmark for evaluating and refining depth perception models. When integrated into the loss function of a depth estimation model, LiDAR data acts as a high-fidelity ground truth, enabling the model to compare its depth predictions directly against LiDAR's precise measurements. This incorporation is critical in improving the model's learning process, allowing for a more adjustment of parameters to align the estimated depth values closely with the actual measurements. However, this methodology presents certain challenges. Heavy reliance on LiDAR data for loss computation can introduce a dependence that overshadows the model's capacity for learning from image data alone. This, in turn, has the potential to limit the model's generalizability and performance in environments where LiDAR data is scarce or insufficient [28].

Early fusion is a technique that combines LiDAR's depth data with image data at the beginning of the processing pipeline. This method creates a rich, multidimensional input that feeds the model with concatenated RGB-D frames. It allows the model to make use of both the textural complexities captured in the images and the spatial accuracy of LiDAR measurements simultaneously. However, this approach also faces significant obstacles, one of which is the different modalities of these two data sets. This aspect makes it challenging for deep learning-based methods to extract information from them [29].

Late fusion presents an alternative strategy, where image and LiDAR data are processed through distinct pipelines before their features are combined at a later stage. This segregation allows each data type to be analyzed using methods best suited to its characteristics, potentially enhancing the model's ability to extract relevant features from both modalities. However, this approach has its pitfalls. If an error occurs in the early processing stages of one modality, it may not be adequately compensated for by the other, leading to compounded inaccuracies. Despite these challenges, late fusion offers flexibility, allowing for different

fusion techniques, such as feature fusion or decision fusion, which can be tailored to the specific needs of the task at hand. By wisely selecting the fusion method, one can optimize the balance between leveraging the unique strengths of each data type and mitigating their individual limitations, thereby enhancing the model's depth estimation performance [29].

The three categories of information of LiDAR technology face a common challenge in incorporating it into available depth estimation methods, such as monodepth, without requiring significant modifications. The first category, which utilizes LiDAR in loss, would need to undergo re-training to incorporate this information effectively. The two other groups would need to redesign their network architecture to accommodate LiDAR technology.

1.2 Depth Completion: Bridging the Gaps in Depth Data

Depth completion is essential in refining depth maps or 3D point clouds by addressing their missing or incomplete depth information. This task is particularly vital in applications where precision and completeness of depth data are crucial. One of the practical applications of depth completion is in the reconstruction of DTMs, which is a key area of focus for this dissertation. Depth completion methods are typically categorized into two broad types: guided (or informed) methods and blind methods, both of which are relevant to the DTM reconstruction domain [30].

1.2.1 Guided/Informed Depth Completion Methods

Guided or informed depth completion methods employ a mask or a similar indicator to identify the regions within a depth map or 3D point cloud that require filling or refinement. This mask serves as a guide, informing the depth completion algorithm about the specific areas lacking depth information. The advantage of these methods lies in their ability to focus on specific areas of the depth map, which can lead to more accurate and efficient completion processes.

In the domain of depth reconstruction, as thoroughly discussed in sections 5.1 and 5.2,

binary segmentation techniques are commonly employed to classify the DTM points into ground and non-ground categories. The non-ground points are subsequently removed, and the gaps are filled using interpolation-based approaches. However, these methods have two major limitations. Firstly, interpolation techniques often fail to fill gaps accurately, particularly when the gaps are extensive or located close to the edges of the DTM, primarily due to the limitations of extrapolation. Secondly, these methods do not account for the strong correlation between segmentation and gap completion.

1.2.2 Blind Depth Completion Methods

Blind depth completion methods, in contrast, do not rely on any prior knowledge or indicators of where the depth information is missing. Instead, these methods work by estimating the missing depth values across the entire depth map or point cloud, applying various computational techniques to infer this data. The primary challenge for blind methods is ensuring the cohesiveness and accuracy of the depth information across the entire map, especially in the absence of specific indicators.

Blind depth completion is a technique that proves to be particularly useful in scenarios where no prior information about the depth map is available. One such scenario is the reconstruction of DTMs from DSM, where non-ground areas are not available in advance. In this regard, DL-based methods, as discussed in detail in sections 4.1 and 5.1 have been employed to regress DTMs from DSMs. However, these methods face a significant challenge as they regress DTMs without considering a crucial factor - that most areas of DTMs are similar to DSMs and should not be altered because they represent accurate LiDAR points.

1.3 Goals and Objectives

This thesis focuses on enhancing MMSs, with particular emphasis on depth estimation and terrain reconstruction, which are identified as pivotal components within the field. It introduces novel methods and frameworks to tackle the most complex challenges in depth

estimation and DTM reconstruction. The research aims to advance the state-of-the-art in the MMS, providing significant contributions to the depth estimation and DTM reconstruction approaches. The following goals and objectives encapsulate the core ambitions of this research:

1.3.1 Addressing the Generalization Problem in DL-based Depth Estimation

Objective: Develop and introduce a novel framework that can be seamlessly integrated with existing networks during the inference phase. This framework is designed to enhance the accuracy of depth estimation across various domains, particularly in dynamic environments. The achievement of this goal has been substantiated through rigorous research, and its findings have been published [31].

1.3.2 Adapting Depth Estimation Methods for Panoramic Images

Objective: Modify and extend the capabilities of the proposed depth estimation framework to effectively handle panoramic images. This adaptation is particularly focused on applications in SLAM, where panoramic imaging is prevalent. The achievement of this goal has been demonstrated through empirical results and has been documented in a published work [32].

1.3.3 Advancing DTM Reconstruction Through a Geospatially Informed Single Task Network

Objective: Present and validate an innovative, Geospatially informed network that is specifically designed for DTM reconstruction as a practical application. The network in question is a regression model that is developed to estimate DTM from DSM using a supervised framework. Our goal is to illustrate the efficacy of geospatially informed algorithms and emphasize the significance of integrating physical constraints into deep learning frameworks. We specifically leverage geospatial data pertaining to terrain topography, where the relationship is defined as nDSM = DSM - DTM. In this context, DSM serves as the input, DTM represents the final

output for loss computation, and nDSM is the network output. This approach underscores the pivotal role of topographic data in improving the performance of DTM reconstruction using deep learning models.

1.3.4 Advancing DTM Reconstruction Through a geospatially Informed Multi Task Network

Objective: The inherent regression nature of a geospatially informed network gives rise to an issue wherein common voxels between DSM and DTM are rendered, which is not desirable. To address this, a multi-task geospatially informed network has been proposed and validated to estimate values for non-ground points, which represent the differences between DSM and DTM. This approach ensures that the network does not alter ground points, which are accurately and meticulously collected via LiDAR. The proposed network thus offers a novel solution to the issue of invaluable common voxels, enabling better accuracy and precision in the context of DSM and DTM.

1.4 Research Contributions

The research outlined in this thesis makes a significant contribution to the field of MMS, specifically in depth estimation and DTM reconstruction. By introducing innovative approaches and methodologies, this research carries substantial implications both academically and practically, across various domains where precise depth perception is indispensable. The following subsections detail the key impacts of this research.

1.4.1 Enhancements to Depth Estimation Techniques

This dissertation tackles the prevalent problem of generalization in DL-based depth estimation, with the aim of establishing dependable and accurate depth perception in dynamic environments. The proposed novel framework introduces an adaptive Backpropagation Refinement

Scheme (BRS) that improves the performance of existing networks during the inference phase, resulting in superior depth estimations. This enhancement is of utmost importance for applications in autonomous vehicles and robotics, where the ability to adjust to rapidly changing environments is critical.

1.4.2 Enhanced Depth Estimation for Panoramic Images

This dissertation introduces an extension to depth estimation methods for panoramic images, which involves adapting the interpolation module of the proposed adaptive BRS for Panoramic images. This extension broadens the applicability of these techniques, particularly in the context of SLAM applications. Moreover, this expansion not only enhances the quality of spatial understanding in panoramic views but also enables the deployment of depth estimation in a wider range of environments and scenarios.

1.4.3 Enhanced DTM Reconstruction as a Regression Problem

The presented thesis introduces a geospatially informed network that incorporates the correlation between the input and output of a network, DSM and DTM, respectively. The normalization of DSM referred to as nDSM, is included in the network through symmetric skip connections. This integration allows for a more comprehensive representation of the network's behavior and enhances its performance.

1.4.4 Advanced Multi-Task Learning for DTM Reconstruction

The present work introduces a multi-task variant of a regression-based network that aims to overcome a crucial challenge associated with the regression-based approach. Specifically, the issue with a regression-based geospatially-informed network is that it modifies all pixels within the regression process, while only non-ground points ought to be modified, thereby leaving the remaining pixels unchanged. The multi-task version of the network addresses this issue by concurrently considering ground segmentation and regression. This approach allows

the network to exclusively regress values for non-ground points. In summary, the multi-task network variant offers a promising solution to the challenge of accurately identifying and modifying non-ground points while leaving other pixels unchanged.

1.4.5 Contributions to Safe and Autonomous Navigation Technologies

Lastly, the thesis details the advancements in depth estimation and DTM reconstruction as an application of depth completion techniques. These advancements significantly enhance MMS's capabilities in various aspects, notably in safety and navigation.

The integration of DL-based techniques in MMS technology facilitates the creation of more precise and dense depth maps. This improvement is especially significant when there is a lack of LiDAR data, enabling MMS to rely on image-based depth prediction using neural networks. Such advancements are crucial not only for the navigational safety of autonomous systems but also for their overall accuracy. In the context of DTM reconstruction, depth completion techniques play a crucial role in generating a smooth and complete DTM. Such a continuous and detailed representation of the terrain can be particularly beneficial for MMS, as it can facilitate safer and more efficient navigation paths for autonomous vehicles and robotics. These systems can effectively mitigate the risks associated with unforeseen obstacles or challenging terrains, thereby enhancing the overall performance and reliability of such systems. Furthermore, the improved depth estimation and completion techniques contribute significantly to the precision and reliability of SLAM processes, which are essential for the autonomous operation of vehicles and robots in unknown environments. Enhanced SLAM capabilities, supported by accurate depth information, enable these systems to create more reliable spatial maps, improving their ability to navigate autonomously and safely.

1.5 Thesis Overview

This thesis is organized into six chapters, each addressing a specific dimension of this dissertation, emphasizing depth estimation and DTM reconstruction. The structure of the thesis is as follows:

- Chapter 1 sets the context for the research, introducing the topic and highlighting its significance in the field of MMS. It reviews relevant literature, outlines the research objectives, and discusses the expected impact of the study.

- Chapter 2 provides a comprehensive overview of the fundamental theoretical concepts that form the basis for advanced techniques and methodologies employed in this thesis. It delves into topics such as optimization and DL, enabling a better understanding of complex theories that are relevant to this research. This chapter's thorough exploration of these foundational concepts lays the groundwork for the more sophisticated analyses that follow, making it an essential read for anyone interested in the research's findings.

- Chapter 3 provides a comprehensive analysis of monocular depth estimation and presents a technique based on BRC. Additionally, the chapter discusses an extension of the model for panoramic images, which can be used in a visual SLAM system. The proposed method provides promising results for depth estimation in both indoor and outdoor environments, demonstrating its robustness and reliability.

- Chapter 4 introduces a geospatially informed regression-based network for extracting DTMs from DSMs. The network is designed based on the physical relationship between DSM and DTM, which is nDSM. The proposed network aims to achieve accurate DTM extraction by leveraging the inherent correlation between the input and output networks. The network's performance has been evaluated using various metrics, demonstrating its effectiveness in producing high-quality DTMs.

- Chapter 5 introduces a fresh geospatially informed multi-task network that overcomes

the limitations of the regression-based network discussed in chapter 4. The regression-based approach modifies all pixels of DSM, including ground pixels that should not be altered. Ground pixels are costly to collect and have high accuracy, making it crucial to preserve them. This innovative network simultaneously addresses the ground segmentation issue.

- Chapter 6 summarizes the key findings and contributions of the thesis. It offers concluding remarks and discusses potential avenues for future research, highlighting the next steps in the field of depth generation in computer vision.

1.6 Summary

In the introductory chapter of this thesis, the research field of depth generation in computer vision is thoroughly explored, highlighting its two key categories: depth estimation and depth completion. The chapter begins by establishing the context and relevance of the research area, then delves into the specific goals and objectives aimed at addressing critical challenges within this field. Following this, the chapter outlines the potential impact of the research, emphasizing its significance in both theoretical and practical applications. Finally, the structure of the thesis is presented, systematically laying out the content and focus of each subsequent chapter. This introductory chapter serves as a roadmap, guiding the reader through the scope, intentions, and structure of the research presented.

These chapters lead to the following publications:

- Alizadeh Naeini A, Ahmad A, Sheikholeslami MM, Claudio P, Sohn G. An Unsupervised Registration of 3D Point Clouds to 2D CAD Model: A Case Study of Floor Plan. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2020;2:9-13.

- Alizadeh Naeini A, Sheikholeslami MM, Sohn G. An Adaptive Refinement Scheme for Depth Estimation Networks. Sensors. 2022;22(24):9755.

- Ahmadi M, Alizadeh Naeini A, Sheikholeslami MM, Arjmandi Z, Zhang Y, Sohn G. HDPV-SLAM: Hybrid depth-augmented panoramic visual SLAM for mobile mapping system with tilted LiDAR and panoramic visual camera. In: 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE); 2023. p. 1-8.

- Alizadeh Naeini A, Sheikholeslami MM, Sohn G. RESSUB-NET: Residual Subtraction Network for DTM Extraction from DSM. In: 2024 IEEE International Geoscience and Remote Sensing Symposium (IGARSS); 2024. Accepted as an oral presentation.

- Alizadeh Naeini A, Sheikholeslami MM, Yoo S, Sohn G, "Synergistic Multi-Task Autoencoder with Subtractive Skip Connections for Enhanced DTM Generation," submitted to ISPRS Journal of Photogrammetry and Remote Sensing.

- Alizadeh Naeini A, Sheikholeslami MM, Sohn G, "Advancing geospatially-informed Autoencoders for DTM Generation," submitted to Remote Sensing Journal.

Chapter 2

Background

2.1 Introduction

In this chapter, we establish the foundation for the complex interweaving of mathematical frameworks and conceptual methodologies that build the basis of the following chapters. The exploration of these fundamental elements is a scholarly exercise and an essential journey into theoretical foundations that will clarify our comprehension and practical use of advanced techniques in the following chapters. This section analyzes and clarifies the core mathematical theories and methodological approaches, creating a solid base for our upcoming analytical efforts.

2.2 Optimization

Optimization is a fundamental branch of mathematics and computer science focused on finding the most efficient or effective solution to a problem. The essence of optimization lies in either minimizing or maximizing some function by systematically choosing the values of certain variables within an allowed set [33]. Mathematically, an optimization problem can be expressed as:

$$\text{Find } x \text{ that minimizes (or maximizes) } f(x) \tag{2.1}$$

$$\text{subject to } g_i(x) \leq 0, \text{ for } i = 1, \ldots, m \tag{2.2}$$

$$\text{and } h_j(x) = 0, \text{ for } j = 1, \ldots, p \tag{2.3}$$

where $f(x)$ is the objective function, $g_i(x)$ represents inequality constraints, and $h_j(x)$ represents equality constraints.

This section endeavors to present an in-depth examination of optimization, elaborating on various categorizations essential for grasping this intricate domain. Optimization problems are fundamentally classified based on the presence or absence of constraints. Constrained optimization entails variables that must adhere to specific restrictions delineating feasible solutions, unlike unconstrained optimization, where no such constraints are applied to the variables [34].

The distinction between deterministic and stochastic optimization is another critical classification. Deterministic optimization pertains to scenarios with known and constant data and parameters, facilitating precise predictions and outcomes. Conversely, stochastic optimization is applied under data uncertainty, often with random variables, requiring strategies adept at managing unpredictability [35].

The variable nature is a pivotal aspect, with continuous optimization allowing variables to adopt any value within a range, providing a seamless landscape for problem-solving. Discrete optimization, on the other hand, involves variables with distinct, separate values, introducing unique challenges and solution methodologies [36].

Further broadening the categorization, it is advantageous to differentiate between static and dynamic optimization. Static optimization pertains to single-stage decision-making, whereas dynamic optimization involves a series of decisions over time, vital in contexts necessitating sequential choices [37].

Regarding objectives, optimization problems are classified as single-objective or multi-objective. The former concentrates on optimizing a singular criterion, while the latter addresses the optimization of multiple, often competing, objectives, requiring a refined approach to determine the best trade-offs [38].

Incorporating global optimization into our algorithmic framework enriches the categoriza-

tion, creating a clear distinction between methods aimed at identifying global optima—those solutions that are optimal across the entire search landscape—and those that focus on local optima, which are optimal within a limited region but not necessarily across the broader landscape [39]. Global optimization techniques, such as specific evolutionary algorithms and metaheuristic approaches like Simulated Annealing and algorithms like particle swarm optimization (PSO), are indispensable for complex problems [40], aiming to bypass the limitations of local optima. On the other hand, local optimization methods, particularly those that are gradient-based, such as gradient descent and its variants, specialize in efficiently progressing toward the nearest optimum. These local optimizers are crucial for many optimization tasks, providing rapid convergence in well-behaved landscapes. However, their effectiveness is typically constrained to finding the nearest local maximum or minimum, which may not represent the absolute best solution in a landscape characterized by multiple optima. The application of these optimization categories in deep learning is profound. For instance, while gradient-based methods like SGD are typically used for local optimization in neural network training, global optimization techniques are pivotal when the goal is to ensure the avoidance of local minima and the achievement of the best possible model performance. In deep learning, this distinction influences how models are trained and how they generalize to unseen data.

In deep learning, deterministic optimization is often seen in the backpropagation process, where a known dataset is used to adjust weights with certainty. Stochastic optimization comes into play with techniques like SGD, where the inherent randomness in selecting data points for training introduces a stochastic element. The choice between continuous and discrete optimization can be observed in different aspects of deep learning, such as weight adjustments (continuous) or structure optimization (discrete, such as in neural architecture search).

By integrating these categorizations and their applications, especially in the domain of deep learning, the section not only offers a comprehensive understanding of optimization but also highlights its multifaceted nature and the diverse strategies employed to address

various challenges, illustrating the critical role optimization plays in the advancement of deep learning methodologies.

2.2.1 Constraints

In constrained optimization, the objective is to find the optimal solution to an objective function within a set of specified constraints. This approach is essential in scenarios where solutions must comply with certain restrictions or conditions. Linear Programming (LP) is a technique used to optimize a linear objective function, subject to linear equality and inequality constraints. The mathematical representation involves maximizing or minimizing a function, $\max(\text{or}\min)\,\mathbf{c}^T\mathbf{x}$, subject to the constraints $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$, where $\mathbf{x}$ is the variable vector, $\mathbf{c}$ and $\mathbf{b}$ are coefficient vectors, and $\mathbf{A}$ is a coefficient matrix.

Nonlinear Programming (NLP) focuses on optimizing a nonlinear objective function alongside nonlinear constraints. Unlike LP, the functions involved in NLP, denoted as $f(\mathbf{x})$, $g_i(\mathbf{x})$, and $h_j(\mathbf{x})$, are not necessarily linear. This makes NLP suitable for more complex problems where the relationship between variables is not linear.

Unconstrained optimization delves into optimizing an objective function without any external constraints. This type of optimization is pivotal in situations where the decision variables are not bound by any restrictions, allowing for free exploration of the solution space to find the optimal value of the objective function.

2.2.2 Inherent Nature

This subsection delves into categorizing optimization problems based on the nature of the data and the variables involved, which can be deterministic and stochastic.

In deterministic optimization, the scenario is one where clarity prevails. Every parameter and piece of data within the optimization model is known with certainty, eliminating any room for ambiguity. This approach is particularly effective in environments that are stable and predictable, where outcomes can be reliably forecasted based on the input variables. It is a framework where the knowns outweigh the unknowns, making it a solid foundation

for decision-making in a controlled setting. On the other hand, stochastic optimization introduces an element of unpredictability. This method is called upon when the data is shrouded in uncertainty, a common scenario in dynamic and complex environments. Instead of fixed values, the variables and parameters are described using probability distributions, acknowledging the inherent variability and unpredictability of the real world. This approach is not about eliminating uncertainty but rather embracing it, providing a way to make informed decisions even when the future is not fully known.

As far as deep learning is concerned, it typically does not rely on deterministic optimization alone due to the inherent complexity and unpredictability of the environments and data it deals with. The dynamic, uncertain, and complex nature of the problems addressed by deep learning makes stochastic optimization methods more suitable than purely deterministic approaches [41].

2.2.3 Search Strategy

This subsection focuses on the different algorithms used in optimization, highlighting their unique characteristics and applications.

Gradient-based Methods

Gradient-based methods stand out in areas such as deep learning, where they excel in managing complex, high-dimensional objective function landscapes. Their utility shines when exhaustive or random search methods fall short, offering a practical approach to navigating intricate optimization challenges.

First-order methods are particularly esteemed in deep learning for their computational efficiency, scalability, and flexibility. These attributes make them well-suited for the vast-scale problems characteristic of deep learning, where models often encompass millions of parameters. The efficiency of first-order Methods is that they require fewer computational resources than their higher-order counterparts, which is vital for processing large datasets [42].

Among the prevalent first-order methods are gradient descent, Stochastic Gradient Descent

(SGD) [43], and various momentum-based and adaptive learning-rate methods [44]. gradient descent, though straightforward, may become computationally intensive with large datasets. In contrast, SGD, by leveraging single samples or mini-batches, enhances computation speed and avoids local minima. Momentum-based methods, such as SGD with momentum and Nesterov accelerated gradient, incorporate a momentum term to steer gradients more effectively, thereby enhancing convergence [44]. Adaptive learning-rate methods like Adagrad [45], RMSprop [46], Adam [47], and AdaDelta [48] dynamically adjust the learning rate, often resulting in swifter convergence.

Second-order methods, which utilize both the first and second derivatives of the objective function, offer a deeper understanding of the optimization landscape. However, they are less favored in deep learning due to their high computational demands and complexity. These methods' computational intensity primarily arises from the need to calculate and store the Hessian matrix, a daunting task for large-scale issues prevalent in deep learning. The Hessian's size and the computational and memory resources required render second-order methods less practical for deep learning applications. Moreover, their theoretical robustness does not always translate into practical efficiency, particularly for complex objective functions with numerous local minima and maxima, making them less appealing for large neural networks [49].

Examples of second-order Methods: Newton's method, a classic second-order method, uses the Hessian matrix directly to find function minima. Its application, however, is limited in large-scale problems due to the computational intensity involved.

Quasi-Newton methods, such as the Broyden–Fletcher–Goldfarb–Shanno (BFGS [50]) method and the Limited-memory BFGS (L-BFGS [51]), provide a more practical approach. They approximate the Hessian matrix, thereby reducing computational demands. The BFGS method updates the Hessian approximation as follows:

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \tag{2.4}$$

Where I denotes the identity matrix, $\mathbf{s}_k$ represents the step taken from the current point to the next, and $\mathbf{y}_k$ signifies the difference in gradients between two successive points. The term

$\frac{\mathbf{y}_k^T \mathbf{s}_k}{\mathbf{s}_k^T \mathbf{y}_k}$ and its transpose are key components in the update formula, crucial for adjusting the Hessian approximation to ensure its positive definiteness and to incorporate the curvature information from the most recent step. These adjustments are essential for the method's efficiency and accuracy in navigating towards the function's optimum.

In contrast, L-BFGS is designed for large-scale problems by storing only a limited number of vectors that represent the Hessian's approximations, significantly cutting down on memory requirements:

$$H_k^{approx} = V_k^T H_k^0 V_k + \rho_k s_k s_k^T \tag{2.5}$$

Where H_k^{approx} is the approximate Hessian matrix, V_k is a matrix composed of the stored vectors, H_k^0 is an initial approximation of the Hessian matrix, ρ_k is a scaling factor, and $\mathbf{s}_k$ is as defined above.

In conclusion, while second-order methods have certain theoretical advantages, their practical use in deep learning is constrained by high computational demands and inherent complexity. This makes first-order methods the more commonly used approach in large-scale optimization problems in this field.

Evolutionary and Population-based Algorithms

Evolutionary and population-based algorithms, encompassing methods like genetic algorithms and PSO, draw inspiration from the intricacies of natural biological processes [52]. These algorithms adopt the principles of natural selection and the social behaviors observed in various species, facilitating the exploration of complex search spaces and the convergence towards optimal solutions. PSO, in particular, stands out as a computational technique that was introduced by [53]. Inspired by the social behavior patterns of birds and fish, PSO is a pivotal element of swarm intelligence methodologies, renowned for its efficacy in addressing global optimization challenges.

PSO is adept at handling objective functions that are intricate, nonlinear, or non-differentiable, positioning it as a preferred option in situations where conventional gradient-based methods may falter. The algorithm simulates a swarm of particles, each representing

a potential solution, moving through the search space. The trajectory of these particles is influenced by their personal experiences and the collective knowledge of the swarm, facilitating a dynamic balance between the exploration of new possibilities and the exploitation of known advantageous regions.

The operational essence of PSO is encapsulated in two fundamental equations: the velocity update equation and the position update equation. The velocity update equation is crucial for adjusting a particle's velocity, considering its present velocity, the best position it has encountered, and the best position known to the swarm. This equation is articulated as follows:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (p_{best,i} - x_i(t)) + c_2 \cdot r_2 \cdot (g_{best} - x_i(t)) \tag{2.6}$$

Here, $v_i(t+1)$ represents the new velocity of particle i, w denotes the inertia weight affecting the particle's momentum, c_1 and c_2 are cognitive and social coefficients, respectively, r_1 and r_2 are random values within the [0,1] range, $p_{best,i}$ is the optimal position particle i has discovered, g_{best} signifies the best position identified by any particle in the swarm, and $x_i(t)$ is the current position of particle i.

Subsequent to the velocity update, the position of each particle is recalibrated based on its velocity, following the equation:

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.7}$$

The initialization of the PSO process involves assigning each particle a random position and velocity, followed by iterative updates to their velocities and positions based on individual and collective experiences. This iterative process persists until the algorithm reaches a predetermined number of iterations or attains a specified solution quality.

PSO's versatility is evident in its successful applications across various domains, including engineering design, economic systems, and neural network training. Its adaptability to diverse optimization problems and straightforward implementation underscores its value as a robust tool in the optimization field, demonstrating its broad applicability and effectiveness.

2.3 Deep Learning

Deep learning, a transformative branch of machine learning, mimics the human brain's structure to model complex data through multi-layered neural networks. This document explores deep learning from multiple perspectives, emphasizing architecture, optimization, training, inference processes, and domain adaptation.

2.3.1 Mathematical Parameters

In deep learning, various mathematical parameters and operations are used consistently across different models and layers:

- Input data, denoted as X, is the raw data fed into the network, often represented as a matrix or tensor. Hidden input, represented by x, denotes the initial input to the layer or block.
- Scaling (γ) and shifting (β) parameters are used in normalization layers. γ and β are learnable parameters that adjust the scale and shift of normalized data, respectively.
- Weights, represented by W, are learnable parameters of the model. The interpretation of W depends on the layer type: 1) In fully connected layers, W refers to weight matrices connecting neurons. 2) In convolutional layers, W represents convolutional kernels or filters.
- Biases, denoted by b, are learnable parameters added to the weighted inputs.
- Activation functions, denoted by f, are non-linear functions applied to the weighted sum of inputs.
- Layer output, denoted by h, is the output after applying weights, biases, and activation functions.

- The convolution operation, denoted by $*$, is a key operation in convolutional layers involving sliding a kernel over the input.
- Element-wise multiplication, represented by $\odot$, refers to the Hadamard product or element-wise multiplication in matrices or vectors.
- Concatenation, denoted by $\oplus$, combines vectors or matrices along a specified axis.
- Probability distribution, denoted by P, represents the distribution of data points in the feature space, essential for modeling and predictions.
- Feature space, denoted by F, is the set of input variables or features used by the model.
- Source and target domains refer to different environments or contexts where the model is trained and applied. The feature distribution in the source domain, $P_{\text{source}}(F)$, and in the target domain, $P_{\text{target}}(F)$, represent the distributions where the model is trained and applied, respectively.

2.3.2 Layers

Interface Layers

At the commencement of the neural network, the **input layer** serves as the gateway for data, where the raw input X is introduced into the network without undergoing any computational transformation. This straightforward passage of data ensures that the network receives the unaltered information necessary for subsequent processing.

Conversely, the **output layer** marks the network's conclusion, delivering the final results. This layer's functionality is tailored to the specific task at hand, whether it be classification or regression.

In classification tasks, the output Y is typically computed using the softmax function applied to the linear transformation of the last hidden layer's output H, represented as $Y = \text{softmax}(W_o \cdot H + b_o)$. For regression tasks, a linear function is employed, yielding

$Y = W_o \cdot H + b_o$, where W_o and b_o symbolize the output layer's weights and biases, respectively [54].

Core Computational Layers

Here, we delve into essential components of DL, like dense or fully connected layers, which form the fundamental building blocks of neural networks through their complete inter-neuronal connectivity. Convolutional layers are crucial for processing grid-like data, employing learnable filters to extract spatial features. pooling layers reduce data dimensionality, aiding in feature summarization, while recurrent layers are key for sequential data analysis, leveraging memory mechanisms to maintain temporal information. Additionally, various normalization techniques, including Batch, layer, and instance normalization, are employed to stabilize and accelerate training, enhancing the network's performance and efficiency. These layers collectively contribute to the network's ability to learn and model complex patterns and relationships in data, underscoring their pivotal role in the field of deep learning.

Dense or Fully Connected Layers are foundational building blocks within neural networks, characterized by their complete inter-neuronal connectivity. The output of these layers, H_l, is derived from applying a non-linear activation function f to the linear combination of the previous layer's output H_{l-1} and the current layer's weights W_l and biases b_l [54].

Convolutional Layers stand as a cornerstone in processing grid-like data structures, such as images. These layers utilize learnable filters to perform convolution operations, capturing spatial hierarchies and features from the input. The output is computed by sliding these filters over the input and applying a non-linear activation function to the convolution results, enhancing the network's feature extraction capabilities [54].

Pooling Layers, essential in reducing the dimensionality of data, come in two main varieties: max pooling and average pooling. Max pooling selects the maximum value from a subset of the input, while average pooling calculates the mean, both contributing to the network's resilience and feature summarization [54].

Recurrent Layers, pivotal in handling sequential data, incorporate a memory mechanism

that allows the network to retain information from previous inputs. This is achieved by updating a hidden state h_t based on the current input x_t and the previous state h_{t-1}, enabling the network to make informed predictions based on past context. The output y_t at each timestep is calculated using the current hidden state, facilitating the network's ability to perform tasks that require an understanding of temporal sequences [54].

Batch Normalization plays a pivotal role in stabilizing and accelerating the training process of deep neural networks. Normalizing the activations of each layer helps in reducing internal covariate shift, which in turn leads to faster convergence and reduced sensitivity to initialization. The mathematical representation of batch normalization is given as $BN(x) = \gamma\left(\frac{x-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\right) + \beta$, where μ_B and σ_B^2 are the mean and variance of a batch, and γ and β are learnable parameters that restore the representation capacity of the network [55, 54].

Layer Normalization, often utilized in recurrent networks and Transformer models, normalizes the inputs across the features for each data point in a batch. Its equation, $LayerNorm(x) = \gamma\left(\frac{x-\mu}{\sqrt{\sigma^2+\epsilon}}\right) + \beta$, highlights the normalization across features rather than across the batch [56].

Instance Normalization and **Group Normalization** are variations that provide flexibility in handling different data structures and training scenarios. Instance normalization, used extensively in style transfer, normalizes each instance in a batch independently. Group normalization divides the channels into groups and normalizes these groups, beneficial for small batch sizes [57].

Dropout Layers: Dropout layers play a pivotal role in neural networks, predominantly in reducing overfitting during the training phase. Overfitting is a common issue where a model excessively learns the details and noise in the training data, negatively impacting its performance on unseen data. The dropout layer mitigates this by randomly omitting a subset of neurons in the network during training. Mathematically, this process is described by the equation:

$$y = x \odot m \tag{2.8}$$

m is a mask vector with elements drawn from a Bernoulli distribution with a probability p,

known as the dropout rate. The dropout rate p, typically between 0.2 and 0.5, indicates the likelihood of any neuron being omitted during training. Notably, during the testing phase, dropout is disabled to leverage the full capacity of the model. This strategy of randomly dropping neurons simulates training multiple neural network architectures concurrently, enhancing the model's ability to generalize well to new data by reducing neuron co-adaptation [54].

Attention Layers: Attention layers in neural networks have revolutionized the handling of sequence data, offering significant improvements in tasks like natural language processing and machine translation [58]. These layers enable models to selectively focus on relevant parts of the input sequence, akin to human attention mechanisms. This selective focus is crucial for tasks where different segments of the input carry varying levels of importance.

Mathematically, the attention mechanism is formulated in three main steps: score calculation, weights calculation, and the computation of the context vector. The score calculation step computes the relevance score for each key-query pair, usually through a dot product or a neural network function. The scores are then normalized using a softmax function to derive weights that sum up to one. These weights indicate the importance of each input segment. The final step involves computing the context vector, a weighted sum of values based on the derived weights. This context vector is then used in subsequent processing steps or as input to other network layers.

The equations governing these steps are as follows:

$$score(q, k_i) = q \cdot k_i \tag{2.9}$$

In this equation, q represents the query, and k_i represents the key. This method is straightforward and computationally efficient, making it a popular choice in many attention-based neural network architectures.

$$\alpha_i = \frac{\exp(score(q, k_i))}{\sum_j \exp(score(q, k_j))} \tag{2.10}$$

$$context = \sum_i \alpha_i v_i \tag{2.11}$$

Here, α_i are the weights, v_i are the values associated with each input part, and *context* is the aggregated context vector.

The attention mechanism has significantly improved the ability of neural networks to handle long-range dependencies, especially in complex sequence modeling tasks. It has become a key component in state-of-the-art models, enhancing their ability to discern which parts of the input are relevant to each step of the output. This capability is particularly beneficial in translation tasks, where the relevance of different parts of the input sentence can vary significantly throughout the process of generating a translated sentence [59].

Transformer Layers: In the context of transformers, a state-of-the-art architecture in deep learning, particularly for handling sequences, the model is primarily built upon self-attention mechanisms and position-wise feed-forward networks. A transformer layer typically consists of two main components: a multi-head self-attention mechanism and a position-wise feed-forward network. The self-attention mechanism allows the model to weigh the significance of different parts of the input sequence differently, enabling it to capture dependencies without regard to their distance in the input sequence. This is followed by layer normalization, which stabilizes the learning process by normalizing the layer inputs [60].

Skip Connections: In neural network architectures like U-Net or ResNet [61], skip connections are used in two primary ways:

Element-wise Addition: $Y = F(X, \{W_i\}) + X$,

where X is the input to a block of layers, $F(X, \{W_i\})$ represents the transformation within the block [61].

Concatenation: $Y = \text{concat}(F(X, \{W_i\}), X)$,

where concat denotes the concatenation operation, $F(X, \{W_i\})$ is the transformation from a deeper layer, and X is the feature map from an earlier layer. This function is parameterized by a set of weights $\{W_i\}$ [62].

2.3.3 Activation Function

Activation functions play a crucial role in deep learning networks by introducing non-linearity to the model. They enable neural networks to learn complex patterns and relationships within data, making them capable of solving a wide range of tasks, from image recognition to natural language processing. In this section, we will explore several commonly used activation functions and their mathematical formulations, along with a detailed analysis of their properties [63].

Sigmoid Activation Function

The sigmoid activation function, also known as the logistic function, maps its input to a range between 0 and 1. It is often used in binary classification problems as it resembles a probability distribution [63].

Mathematically, the sigmoid activation function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.12}$$

where x is the input to the function. The sigmoid function has a smooth, S-shaped curve, which allows it to introduce non-linearity to the network.

Hyperbolic Tangent Activation Function

The hyperbolic tangent (tanh) activation function is similar to the sigmoid but maps its input to a range between -1 and 1. It is zero-centered, which can help alleviate some issues with gradients during training [63].

Mathematically, the tanh activation function is defined as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.13}$$

The tanh function is also S-shaped like the sigmoid but symmetric around the origin.

Rectified Linear Unit (ReLU) Activation Function

ReLU is one of the most widely used activation functions in deep learning due to its simplicity and effectiveness. It introduces non-linearity by setting all negative values to zero and leaving positive values unchanged [63].

Mathematically, the ReLU activation function is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{2.14}$$

The ReLU function is computationally efficient and helps mitigate the vanishing gradient problem. However, it is not without its issues, such as the "dying ReLU" problem, where neurons may become inactive during training.

Leaky ReLU Activation Function

To address the dying ReLU problem, the Leaky ReLU activation function allows a small, non-zero gradient for negative inputs. This slight modification ensures that neurons with negative inputs continue to learn during training [63].

Mathematically, the Leaky ReLU activation function is defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{otherwise} \end{cases} \tag{2.15}$$

where α is a small positive constant, typically set to a small value like 0.01.

Parametric ReLU (PReLU) Activation Function

PReLU is an extension of Leaky ReLU, where the slope of the negative part is learned from the data during training, rather than using a fixed value like α. This allows the network to adaptively determine the slope for each neuron [63].

Mathematically, the PReLU activation function is defined as:

$$\text{PReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ ax, & \text{otherwise} \end{cases} \tag{2.16}$$

where a is a learnable parameter.

Exponential Linear Unit (ELU) Activation Function

ELU is another variation of the ReLU activation function that addresses the vanishing gradient problem and has negative values for all inputs. It introduces a smooth, exponential curve for negative inputs [63].

Mathematically, the ELU activation function is defined as:

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases} \tag{2.17}$$

where α is a positive constant, often set to 1.

Softmax Activation Function

The softmax activation function is primarily used in the output layer of a neural network for multi-class classification problems. It converts the network's raw scores into a probability distribution over multiple classes [63]. Mathematically, the softmax activation function for a vector of inputs $\mathbf{z}$ is defined as:

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.18}$$

where z_i represents the raw score for class i, and K is the total number of classes.

These are some of the commonly used activation functions in deep learning networks. Each activation function has its own characteristics, advantages, and disadvantages, making their choice dependent on the specific requirements of the task and the network architecture. The proper selection and combination of activation functions can significantly impact the performance and training of deep neural networks.

2.3.4 Architectural Frameworks

Deep learning architectures are foundational structures designed to tackle a wide array of tasks across various domains. These architectures are broadly categorized into two primary types: Feedforward Neural Networks and Recurrent Neural Networks, each encompassing specialized sub-architectures to address distinct challenges and data modalities.

Feedforward Neural Networks (FNNs)

FNNs represent a class of networks where connections between the nodes do not form cycles. This category includes several specialized architectures:

Basic FNNs: The simplest form of FNNs consists of an input layer, multiple hidden layers, and an output layer. Information flows in one direction—from input to output—without any feedback loops. These networks are fundamental for understanding neural network operations and are widely applied in straightforward prediction and classification tasks [54].

Convolutional Neural Networks (CNNs): CNNs are designed to process data with a grid-like topology, such as images. They utilize convolutional layers to automatically learn and hierarchically organize spatial features from input data, making them exceptionally effective for image and video analysis tasks. Well-known architectures like ResNet, VGGNet, and U-Net fall under this category, each with unique mechanisms for feature extraction and representation [54].

Residual Networks (ResNets): A pivotal advancement within CNNs, ResNets introduce skip connections that allow gradients to flow through the network more effectively, addressing the vanishing gradient problem and enabling the training of very deep networks. These connections provide a pathway for earlier layers to directly feed into later layers, enhancing feature reuse and network learning capacity [61].

U-Net Architecture: U-Net is specifically tailored for biomedical image segmentation, featuring a distinctive U-shaped design that incorporates a contracting path to capture context and an expansive path for precise localization. This architecture is adept at handling

limited data scenarios while providing detailed segmentation, attributed to its effective use of skip connections and concatenation of feature maps at various resolution scales [62].

Recurrent Neural Networks (RNNs)

RNNs are designed to handle sequential data, where the order and context of the information are crucial. Unlike FNNs, RNNs have cyclic connections, making them capable of maintaining a form of internal state or memory.

Basic RNNs: Basic RNNs process sequences by maintaining a hidden state that evolves over time, influenced by both the new input and the previous hidden state. This enables them to capture temporal dependencies but often at the cost of struggling with long-term dependencies due to issues like vanishing and exploding gradients [64].

Long Short-Term Memory Networks (LSTMs): LSTMs are an advanced variant of RNNs designed to overcome the limitations of basic RNNs in learning long-term dependencies. They incorporate a complex system of gates—input, output, and forget gates—that regulate the flow of information, allowing the network to maintain and update its state over long sequences effectively [65].

Gated Recurrent Units (GRUs): GRUs simplify the LSTM architecture by combining the forget and input gates into a single update gate and merging the cell state and hidden state. Despite this simplification, GRUs retain the ability to handle long-term dependencies, offering a more efficient alternative to LSTMs for certain applications [66].

In conclusion, the landscape of deep learning architectures is rich and varied, with each architecture designed to address specific types of data and tasks. Feedforward networks offer a powerful tool for tasks where the flow of information is unidirectional and non-temporal, while recurrent networks provide the necessary framework for capturing dynamic temporal behaviors and dependencies in sequence data.

Generative Models

Generative models are a critical category of neural network architectures designed to learn the underlying data distribution and generate new data samples that closely resemble the training data. Three prominent generative models are Autoencoders (AE), Variational Autoencoders (VAE), and Generative Adversarial Networks (GANs), each is discussed in [67].

AE: AEs serve as a foundational generative model. They consist of two primary components: an encoder and a decoder. The encoder takes an input data point, x, and maps it to a lower-dimensional latent representation, z. Mathematically, this is expressed as $z = f_{encoder}(x)$, where $f_{encoder}$ is the encoder function. The decoder then reconstructs the original input, x_{hat}, from the latent representation, z, as $x_{hat} = f_{decoder}(z)$. During training, AEs aim to minimize a reconstruction loss, typically using Mean Squared Error (MSE): $L_{reconstruction} = ||x - x_{hat}||^2$ [54].

VAE: VAE extends the capabilities of AE by introducing probabilistic elements. In VAEs, the encoder maps the input 'x' to two sets of parameters: 'z_mean' and 'z_log_var,' which parameterize a Gaussian distribution in the latent space [54]. Sampling from this distribution is achieved using a reparameterization trick:

$$z = z_mean + \exp(0.5 \cdot z_log_var) \cdot \epsilon \tag{2.19}$$

where ϵ is sampled from a standard Gaussian. The decoder in VAEs retains the same structure as in AEs. VAEs are trained to minimize a loss that combines the reconstruction loss and the Kullback-Leibler (KL) divergence between the learned latent distribution and a unit Gaussian.

The KL divergence (Equation 2.20) quantifies how close the learned latent distribution is to a unit Gaussian distribution.

$$KL_{divergence} = -0.5 \cdot \sum_i (1 + z_log_var - z_mean^2 - \exp(z_log_var)) \tag{2.20}$$

where z_mean is the mean of the latent Gaussian distribution, and z_log_var is the logarithm of the variance of the latent Gaussian distribution.

The combination of the reconstruction loss and the KL divergence in the loss function encourages VAEs to produce meaningful latent representations and generate data that closely resembles the training data.

GANs: GANs are another class of generative models that involve two neural networks: a generator and a discriminator. The generator takes a random noise vector, z, as input and generates a data sample, $G(z)$. In contrast, the discriminator attempts to classify data samples as either real or fake. Mathematically, the generator is denoted as $G(z)$ and the discriminator as $D(x)$. GANs are trained adversarially, with the generator minimizing the loss $L_G = -\log(D(G(z)))$ and the discriminator minimizing $L_D = -\log(D(x)) - \log(1 - D(G(z)))$. Through this adversarial process, GANs learn to generate data by iteratively improving the generator's ability to produce realistic samples and the discriminator's ability to distinguish between real and fake samples [68].

2.3.5 Training Components

Loss Functions

A loss function, or a cost/objective function, is critical in evaluating the performance of a neural network. It measures the discrepancy between the network's predicted output and the actual target values. The choice of a loss function is contingent on the task at hand, such as classification or regression [69].

Cross-Entropy Loss: Used predominantly in classification problems, the cross-entropy loss function aims to measure the dissimilarity between the predicted probabilities and the true class labels. It is designed to encourage the network to assign high probabilities to the correct class. The mathematical representation is given by:

$$H(y, \hat{y}) = -\sum_i y_i \log(\hat{y}_i) \tag{2.21}$$

where $\hat{y}$ represents the predicted probabilities, y the true probabilities, and i indexes over the classes [69].

Mean Squared Error (MSE): Commonly employed in regression tasks, MSE quantifies the squared difference between the predicted and actual values, making it suitable for continuous output scenarios. The MSE is mathematically defined as:

$$MSE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.22}$$

where $\hat{y}$ is the predicted value, y is the true value, and n is the number of samples [69].

Mean Absolute Error (MAE): Widely used in regression analysis. MAE measures the average magnitude of errors in a set of predictions without considering their direction. It's particularly favored for its interpretability, as it directly reflects the average error in the same units as the output. The MAE is mathematically formulated as:

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.23}$$

where $\hat{y}$ represents the predicted value, y denotes the actual value, and n is the total number of observations. Unlike MSE, MAE provides a linear score that penalizes all errors equally, offering a robust perspective against outliers and ensuring a consistent interpretation of the error magnitude across different contexts [69].

2.3.6 Optimization Methods

This name integrates the concept of optimization into the context of training algorithms. Optimization algorithms play a pivotal role in training neural networks by adjusting the model's parameters to minimize the loss function.

Gradient Descent: Gradient Descent is an essential optimization algorithm in the field of machine learning, which iteratively refines model parameters to minimize the loss function. This optimization method adjusts the parameters in the opposite direction of the loss function's gradient concerning the parameters, guiding the parameters toward the loss function's minimum. The updated formula for the parameters is expressed as:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \tag{2.24}$$

Here, θ_t symbolizes the parameter vector at the t^{th} iteration, η represents the learning rate, which determines the step size in the optimization journey, and $\nabla L(\theta_t)$ is the gradient of the loss function L with respect to the parameters at iteration t. This process is depicted in Figure 2.1, illustrating how gradient descent methodically progresses through the error function's landscape to locate the minimum error point. This figure and the underlying concept are further elaborated in the [70], which details the practical implementation and effectiveness of gradient descent in model optimization.

The learning rate is a pillar hyperparameter in the context of gradient-based optimization algorithms, dictating how much the computed update influences the current parameter values. A learning rate of 1 signifies the application of the entire calculated update, whereas a value of 0.1 implies that only 10% of the gradient's magnitude is utilized to adjust the parameters. Commonly adopted values for the learning rate include 0.1, 0.01, and 0.001, reflecting a balance between the need for rapid convergence and the avoidance of overshooting the minimum. Unlike model parameters, the learning rate is manually configured. It remains constant throughout the training process, classifying it as a hyperparameter alongside other model configuration elements, such as the number of layers and units per layer. Figure 2.2 illustrates the impact of various learning rate values on the optimization process, as discussed in [71].

Stochastic Gradient Descent (SGD): SGD is an essential optimization algorithm in machine learning, especially beneficial for handling large-scale data sets and high-dimensional spaces. Unlike batch gradient descent, which calculates the gradient of the cost function across the entire data set, SGD optimizes the parameters incrementally for each individual training instance $x^{(i)}$ and its corresponding label $y^{(i)}$. This attribute makes SGD particularly adept for applications with extensive data like large language models [73]. The update rule for SGD is articulated as follows:

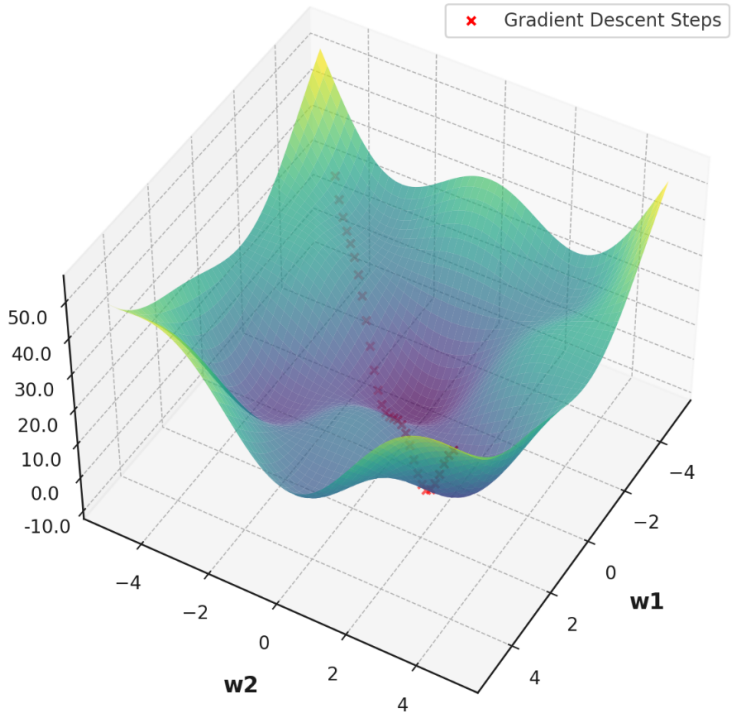
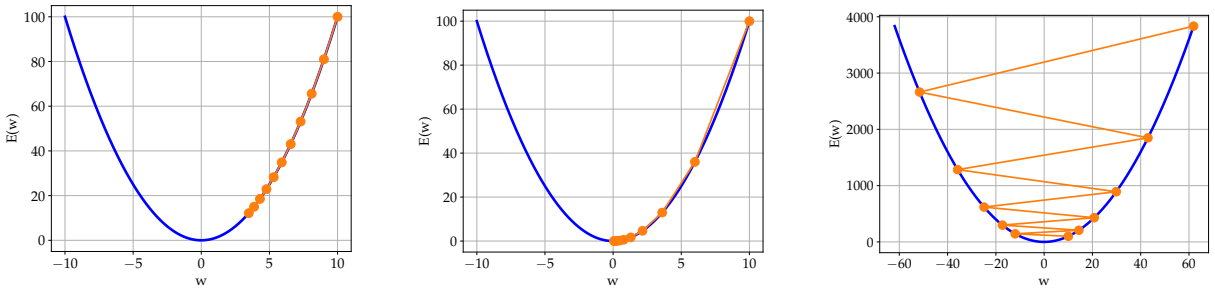


Figure 2.1: Performance of Gradient Descent Method. The x-axis represents the weight parameter w_1, the y-axis represents the weight parameter w_2, and the z-axis represents the error function $E(w_1, w_2)$. The surface plot illustrates the error landscape of the function $E(w_1, w_2)$, and the red dots indicate the steps taken by the gradient descent algorithm. The path traced by the red dots shows the algorithm's progression toward the minimum error, demonstrating the effectiveness of gradient descent in minimizing a two-dimensional error function.



(a) $\eta = 0.05$ (b) $\eta = 0.2$ (c) $\eta = 1.1$

Figure 2.2: Performance of Gradient Descent for Different Values of Learning Rate (η) . The x-axis represents the weight (w), and the y-axis represents the error function ($E(w)$). The blue curve is the quadratic error function. The orange dots indicate the steps taken by the gradient descent algorithm [72].

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.25}$$

In this equation, θ symbolizes the vector of parameters, η represents the learning rate, and $\nabla_\theta J(\theta; x^{(i)}; y^{(i)})$ denotes the gradient of the cost function J concerning θ for the specific data point $x^{(i)}$ and its label $y^{(i)}$. SGD's efficiency stems from its frequent and targeted updates, which are particularly effective in escaping local minima and accommodating online or streaming data.

However, the path to convergence for SGD can be erratic, necessitating precise tuning of the learning rate. Strategies like implementing learning rate schedules or incorporating momentum are often employed to achieve more stable and consistent convergence. Moreover, SGD variants such as momentum SGD or Adam enhance the basic update mechanism, offering improved performance across various optimization landscapes [43, 54].

Mini-Batch SGD: Mini-Batch SGD is an optimization algorithm that represents a compromise between SGD and Batch Gradient Descent. In Mini-Batch SGD, the dataset is divided into small batches, and the parameters are updated for each mini-batch. This approach can

be expressed with the following update rule:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

where θ represents the parameter vector, η is the learning rate, and $\nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$ denotes the gradient of the cost function J with respect to θ calculated over a mini-batch containing n samples, starting from the i-th sample. Mini-Batch SGD combines the advantages of SGD and Batch Gradient Descent, offering a balance between computational efficiency and the stability of the convergence path. It handles large datasets more effectively than Batch Gradient Descent and often converges faster and more smoothly than pure SGD. Furthermore, it can use vectorized implementations, making it more computationally efficient. The mini-batch size is a critical hyperparameter that can influence the performance and convergence characteristics of the algorithm [74].

Momentum: Momentum is an optimization technique that serves as a modification to the traditional gradient descent process. It incorporates a velocity component into the parameter update mechanism, thereby facilitating a more rapid convergence and reducing the propensity for oscillatory movements. The underlying principle of momentum is akin to its physical counterpart, where it enables the system to navigate past local minima, thereby augmenting the likelihood of attaining the global minimum.

The mathematical formulation of the momentum update rule is delineated as follows:

$$v_{t+1} = \beta v_t + (1 - \beta)\nabla L(\theta_t) \tag{2.26}$$

$$\theta_{t+1} = \theta_t - \eta v_{t+1} \tag{2.27}$$

Here, v_t denotes the velocity at the t^{th} iteration, and β represents the momentum coefficient, a hyperparameter that controls the extent of momentum's influence. The term $\nabla L(\theta_t)$ signifies the gradient of the loss function L with respect to the parameters θ at iteration t, and η is the learning rate.

The efficacy of momentum in propelling the optimization process toward the global minimum, rather than getting ensnared in local minima, is a noteworthy advantage, particularly in the training of neural networks. This characteristic is instrumental in enhancing the training efficiency and the eventual performance of the neural network model.

For further reading on the influence and application of momentum in neural network training, refer to the work by [75], which provides an in-depth discussion on the subject. Figure 2.3 shows the role of momentum in reducing the error function.

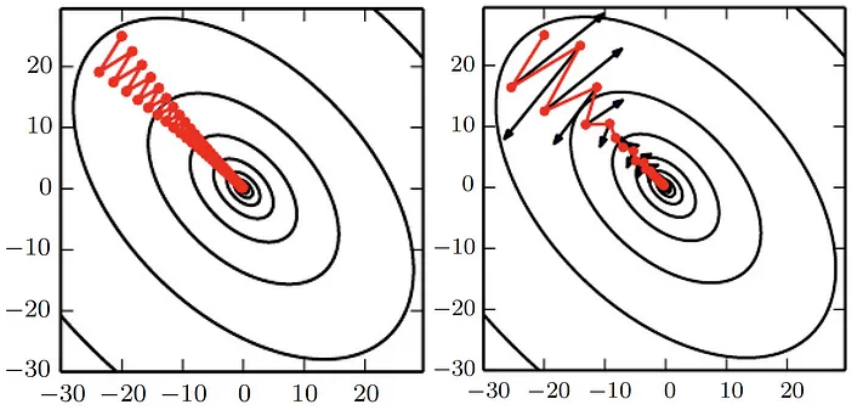


Figure 2.3: Comparison of gradient descent methods on an error function. The diagram on the left demonstrates traditional gradient descent, while the diagram on the right illustrates gradient descent with momentum, showing enhanced convergence [54].

RMSProp: RMSProp is an algorithm that modifies the learning rate for each parameter based on the history of gradients. It is particularly effective in speeding up convergence. The update rule for RMSProp is given by

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\nabla L(\theta_t))^2 \tag{2.28}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla L(\theta_t) \tag{2.29}$$

where $E[g^2]_t$ is the exponentially moving average of squared gradients, β is the decay factor, η is the learning rate, and ϵ is a small constant to avoid division by zero.

Adam: Adam combines aspects of momentum and RMSProp. It maintains moving averages of both gradients and squared gradients and adapts the learning rates for each parameter. The parameter update rule for Adam is:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla L(\theta_t) \tag{2.30}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L(\theta_t))^2 \tag{2.31}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.32}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.33}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.34}$$

where m_t and v_t are the first and second moments of the gradients, β_1 and β_2 are decay factors, and ϵ is a small constant to prevent division by zero.

Regularization Techniques

Regularization techniques are employed to prevent overfitting in models, ensuring they generalize well to new, unseen data. They achieve this by imposing constraints on the model's parameters during training like dropout discussed in section 2.3.2.

L1 and L2 Regularization (Lasso and Ridge): L1 and L2 regularizations, commonly called Lasso and Ridge regularization, are techniques employed to prevent overfitting in machine learning models by introducing penalty terms to the loss function. These penalties encourage the minimization of the magnitude of the coefficients, thereby promoting simpler models that are less likely to overfit.

The L1 regularization (Lasso) adds the absolute value of the magnitude of coefficients as a penalty term to the loss function. The regularized loss function in the context of L1 regularization is given by:

$$L_{\text{L1}} = L + \lambda \sum_{i=1}^{N} |w_i|, \tag{2.35}$$

where L represents the original loss function, w_i denotes the coefficients or parameters of the model, N is the number of parameters, and λ is a non-negative hyperparameter that controls the strength of the regularization.

On the other hand, L2 regularization (Ridge) incorporates the square of the magnitude of coefficients as the penalty term. The regularized loss function for L2 regularization is expressed as:

$$L_{\text{L2}} = L + \lambda \sum_{i=1}^{N} w_i^2, \tag{2.36}$$

where the terms represent the same as described in the L1 regularization.

These regularization methods are fundamental in statistical learning to enhance the generalization capability of predictive models, particularly in situations where the number of parameters is relatively large compared to the number of observations [76].

Early Stopping: Early stopping is a straightforward yet effective technique that involves halting the training process when the validation error starts to increase. This approach does not modify the loss function but rather relies on monitoring the validation error and stopping the training once a predefined threshold is reached.

2.3.7 Training Approaches

Supervised Learning

Supervised learning is a foundational approach in machine learning where the model is trained on a dataset that contains input-output pairs. In this paradigm, the dataset is labeled, meaning each input (or a set of inputs) is associated with a corresponding output label. The primary goal in supervised learning is to learn a mapping from inputs to outputs, enabling the model to predict the output for new, unseen inputs based on this learned mapping. The basic equation for supervised learning can be represented as:

$$y = f(x; \theta) \tag{2.37}$$

In this equation, x represents the input, y is the output, f denotes the model itself, and θ symbolizes the parameters of the model that are adjusted during training to optimize the

model's performance. The training process involves adjusting these parameters to minimize the difference between the predicted output and the actual output in the training data [77].

Unsupervised Learning

Unsupervised learning involves training models on data that does not have labeled outputs. The objective here is not to predict a label based on the input but to discover underlying patterns, groupings, or distributions in the data. This approach is particularly useful for identifying hidden structures in data when explicit labels are not available. The general form of unsupervised learning can be described as

$$z = g(x; \theta) \tag{2.38}$$

In this context, x is the input, z represents the features, patterns, or groupings discovered by the model, and g signifies the model in use. Unlike in supervised learning, where the model's performance is measured against known labels, unsupervised learning algorithms must find ways to infer the dataset's structure without explicit guidance [78].

Semi-Supervised Learning

Semi-supervised learning bridges the gap between supervised and unsupervised learning by utilizing a dataset that is partially labeled, consisting of both labeled and unlabeled data. This approach is particularly beneficial when acquiring fully labeled data is costly or impractical, yet there is an abundance of unlabeled data available. Semi-supervised learning aims to leverage the large volume of unlabeled data to enhance learning accuracy, guided by the limited labeled data. This learning paradigm often involves a blend of techniques from both supervised and unsupervised learning. A common strategy in semi-supervised learning involves training an initial model on the labeled data and then using this model to infer labels or features from the unlabeled data. These inferred labels or features are then used to further refine and improve the model iteratively. The mathematical formulation of semi-supervised learning can be viewed as a combination of the supervised and unsupervised

learning frameworks. A general representation could be:

$$y = f(x; \theta) \quad \text{for labeled data} \tag{2.39}$$

$$z = g(x; \theta) \quad \text{for unlabeled data} \tag{2.40}$$

Here, $y = f(x; \theta)$ represents the supervised learning component where y is the output for labeled input x, and f is the model trained on labeled data. In contrast, $z = g(x; \theta)$ represents the unsupervised component, where z signifies inferred labels or features for unlabeled input x, and g is the model or mechanism applied to the unlabeled data. This hybrid approach allows for a more comprehensive understanding of the data, leveraging the strengths of both supervised and unsupervised learning methods [79].

Self-supervised Learning

Self-supervised learning is a novel approach in machine learning that blurs the line between supervised and unsupervised learning. It involves using a part of the data to supervise the learning process, effectively creating labels from the data itself. This approach is particularly useful when labeled data is scarce or expensive to obtain. In self-supervised learning, an auxiliary task is created from the input data, and the model is trained to predict some aspect of the data based on other aspects. For example, a model might learn to predict the next word in a sentence or the colorized version of a grayscale image. The general formulation of self-supervised learning can be expressed as:

$$\hat{y} = f(x; \theta) \tag{2.41}$$

In this equation, $\hat{y}$ is the prediction made by the model, x represents the input data, f denotes the learning model, and θ symbolizes the model's parameters. The key in self-supervised learning is the generation of $\hat{y}$, which is derived from x itself, unlike in supervised learning, where y is provided as part of the training data. This method allows the model to leverage large amounts of unlabeled data, making it particularly powerful for tasks where labeled data is limited. The process typically involves a feature learning phase, where the model

learns rich data representations, followed by a fine-tuning phase on a smaller set of labeled data, if available. Self-supervised learning has shown remarkable success in various domains, especially in natural language processing and computer vision [80].

Multi-Task Learning

Multi-Task Learning (MTL) is a learning paradigm in machine learning where multiple learning tasks are solved simultaneously, with the aim of improving generalization by leveraging the domain-specific information contained in the training signals of related tasks. The fundamental concept of MTL is that the tasks share commonalities and differences, and by learning them together, the model can effectively improve its performance on each task. This approach contrasts with traditional machine learning methods, where each task is solved in isolation. In MTL, the training process involves optimizing a joint loss function encompassing multiple tasks. A generalized formulation of Multi-Task Learning can be written as:

$$\min_{\theta} \sum_{i=1}^{T} \lambda_i L_i(f(x^{(i)}; \theta), y^{(i)}) \tag{2.42}$$

Here, T represents the total number of tasks, λ_i are task-specific weighting factors, L_i is the loss function for the i-th task, f is the shared model or models across tasks, $x^{(i)}$ are the inputs, $y^{(i)}$ are the outputs for each task, and θ are the shared parameters of the model. The goal of MTL is to find a set of parameters θ that minimizes the joint loss function, allowing knowledge to be shared across tasks. This shared representation can lead to more robust models that better generalize individual tasks, particularly when the tasks are related but not identical. MTL is particularly effective in scenarios where the amount of data for each task is limited, as it enables the model to benefit from the information contained in the training data of all tasks [81].

Transfer Learning

Transfer Learning is a strategy in machine learning where a model developed for a particular task is reused as the starting point for a model on a second task. It is particularly useful

in scenarios where the data available for training on the second task is limited. Transfer Learning leverages the knowledge gained while solving one problem and applies it to a different but related problem. This approach often involves two main phases: pre-training and fine-tuning. In the pre-training phase, a model is trained on a large, often generic dataset. Then, during the fine-tuning phase, the model is adapted to a specific task using a smaller, task-specific dataset. A general representation of transfer learning can be expressed as adapting a pre-trained model $f_{\text{source}}(x; \theta_{\text{source}})$ to a target task, resulting in a fine-tuned model $f_{\text{target}}(x; \theta_{\text{target}})$, where:

$$\theta_{\text{target}} = \text{adapt}(\theta_{\text{source}}, \text{data}_{\text{target}}) \tag{2.43}$$

In this equation, θ_{source} are the parameters of the model trained on the source task, and θ_{target} are the parameters adapted for the target task. The function adapt represents the process of fine-tuning the model on the target task data $\text{data}_{\text{target}}$. Transfer learning is particularly powerful in deep learning, where models pre-trained on large datasets can learn general features that are applicable to a wide range of tasks, reducing both the need for large amounts of task-specific data and the computational resources required for training [82].

Active Learning

Active Learning is a form of supervised machine learning in which the algorithm can interactively query the user (or some other information source) to obtain the desired outputs at new data points. In contrast to passive learning, where the learning algorithm is given a fixed set of labeled data, active learning allows it to choose the data from which it learns. This approach is particularly beneficial when labeled data is scarce or expensive to obtain, as it aims to reduce the number of training examples needed to train a model effectively. The core idea of active learning is to select the most informative samples for which to query labels. A general approach in active learning involves a model $f(x; \theta)$ and a selection criterion ψ, which determines the informativeness of an unlabeled instance. The process can be represented as:

$$x^* = \arg\max_{x \in \mathcal{U}} \psi(x; \theta) \tag{2.44}$$

$$\theta = \text{train}(f, \mathcal{L} \cup \{(x^*, y^*)\}) \tag{2.45}$$

Here, $\mathcal{U}$ is the pool of unlabeled data, and $\mathcal{L}$ is the set of labeled data. The function ψ evaluates and selects the most informative sample x^* from $\mathcal{U}$, which is then labeled as y^*. The model parameters θ are updated by retraining the model f with the new labeled sample added to $\mathcal{L}$. This iterative process of selecting the most informative data points and updating the model continues until a satisfactory level of performance is achieved or resources are exhausted. Active learning is particularly useful in scenarios where unlabeled data is plentiful, but labeling is costly or time-consuming [83].

One-Shot Learning

One-shot learning is a concept within machine learning, particularly in pattern recognition and classification. It refers to the ability of a machine learning model to learn from a very limited amount of data—typically from only one or a few training examples per class. This approach contrasts traditional machine learning methods, which often require large datasets for accurate results.

A common mathematical approach in one-shot learning involves using Siamese networks or similar architectures. A Siamese network consists of two identical subnetworks, each processing one input. The outputs of these subnetworks are compared to determine the similarity between the inputs.

One-shot learning has applications in fields where data is scarce or difficult to collect. For example, in facial recognition, where collecting thousands of images of a single individual can be impractical, one-shot Learning enables the development of systems that can recognize faces with minimal training data. Similarly, in medical imaging, it assists in diagnosing rare diseases with limited examples.

The potential of one-shot Learning lies in its ability to adapt quickly to new information, essential in dynamic environments or applications where data collection is costly or infeasible [84].

Zero-Shot Learning

Zero-Shot Learning (ZSL) is an advanced machine learning technique where a model learns to recognize objects or concepts that it has never seen during training. This approach is particularly valuable in scenarios where it is impractical to have a labeled example for every class that the model needs to recognize.

The essence of ZSL lies in the model's ability to generalize from seen classes to unseen classes by leveraging semantic relationships. Typically, ZSL involves embedding both the visual features and the class labels (or attributes) into a shared space. The relationship between seen and unseen classes is often captured through an attribute or a class description vector. A common formulation of the problem in a ZSL setting is as follows:

$$f : X \rightarrow Y_{seen} \cup Y_{unseen} \tag{2.46}$$

Here, f represents the learning function, X denotes the feature space, and Y_{seen} and Y_{unseen} are the sets of seen and unseen class labels, respectively. The objective is to learn a mapping from the feature space to a space where both seen and unseen classes are represented.

A critical component in ZSL is the side information, which bridges the gap between seen and unseen classes. This information could be in the form of class attributes, word vectors of class names, or textual descriptions. For example, if the model has learned about 'cats' and 'dogs', and it knows that 'tigers' are similar to 'cats' but bigger and with stripes, it can recognize 'tigers' despite not having been trained on them directly.

Applications of ZSL are widespread, especially in fields where exhaustive labeling is infeasible. Examples include image recognition systems that need to recognize thousands of object categories, natural language processing for language understanding, and more.

ZSL pushes the boundaries of what is possible in machine learning by enabling models to make inferences about objects or concepts they haven't explicitly learned, showcasing the power of abstraction and generalization in artificial intelligence [85].

Few-Shot Learning

Few-shot learning is a significant approach in machine learning, particularly useful in scenarios where acquiring a large amount of labeled data is challenging. In few-shot learning, the model is trained to make accurate predictions with only a small number of training examples for each class. This method is crucial for tasks where data is scarce or expensive to collect [86].

The mathematical challenge in few-shot learning is to develop algorithms that can generalize well from a small dataset. A common technique used is meta-learning, where the model is trained on a variety of learning tasks, and the aim is to learn a learning algorithm (or meta-learner) that can quickly adapt to new tasks with minimal data [87]. The objective function in a meta-learning scenario can be represented as follows:

$$\min_{\theta} \sum_{\tau \in T} \mathcal{L}(f_\theta(X_\tau^{train}), Y_\tau^{train}) + \lambda \mathcal{R}(\theta) \tag{2.47}$$

Here, T represents a set of tasks, τ is a specific task, X_τ^{train} and Y_τ^{train} are the training inputs and labels for task τ, respectively. The function f_θ is the model parameterized by θ, $\mathcal{L}$ is the loss function, and $\mathcal{R}$ is a regularization term with a balancing parameter λ. The goal is to find parameters θ that minimize the loss across all tasks, thereby enabling the model to learn effectively from limited data.

Few-shot learning finds applications in various fields, such as computer vision for object recognition, natural language processing, and medical diagnosis. For instance, in medical imaging, few-shot learning can diagnose rare diseases with only a few examples available.

This approach is particularly valuable in the current era of AI, as it moves towards creating more efficient and adaptable models that can perform well under the constraints of data scarcity. The development of Few-Shot Learning techniques is a step forward in making machine learning models more versatile and applicable in real-world scenarios where large datasets are not always available.

2.3.8 Domain Adaptation

Domain Adaptation (DA) is a pivotal concept in machine learning that addresses the challenge of transferring knowledge from a well-labeled source domain to a less-labeled or unlabeled target domain. The essence of DA lies in its ability to leverage the knowledge from the source domain, where ample data is available, and apply it to the target domain, where data is sparse or comes with different distributions. This adaptability is crucial in real-world applications where it's often impractical to collect extensive labeled data for every possible scenario. DA aims to overcome the limitations caused by the distributional shift between the source and target domains, enabling models to maintain high performance even when confronted with new, unexplored data environments [88].

Domain Adaptation is broadly divided into four types, each tailored for specific variations in domain characteristics [89]. Here, we explore each type through the lens of depth estimation.

Intra-Domain Adaptation

Intra-domain adaptation involves adapting to different distributions within the same overarching domain. For depth estimation, this might mean adapting a model trained on indoor scenes during daylight to perform accurately in indoor scenes at night. The main challenge in this adaptation is the variation in lighting conditions within the same spatial context, described by:

$$P_{\text{source}}(X) \neq P_{\text{target}}(X) \tag{2.48}$$

where X represents the features in indoor scenes under different lighting conditions. In the application of DTM generation, it can be adapting a model to handle DSM data taken under varying weather conditions. While the overarching domain of terrain modeling remains the same, the change in weather conditions (like clear vs. foggy) alters the data distribution.
here, two approaches stand out: f-BRS [90] and fine-tuning. F-BRS is particularly adept at refining segmentation models in response to user input, making it suitable for scenarios where depth estimation models need to adjust to subtle changes in lighting or scene composition

within the same type of environment. This interactivity allows for rapid model adaptation to new conditions within the same overarching domain. On the other hand, fine-tuning is a well-established method where a pre-trained model on a general dataset is subsequently refined with a smaller, more specific dataset. In depth estimation, this involves initially training a model on a diverse set of images and then fine-tuning it with images that represent specific conditions of the target domain. Such an approach is effective in adjusting to subtle intra-domain shifts [91].

Inter-Domain Adaptation

Inter-domain adaptation requires adapting across different domains. In the context of depth estimation, this can be seen in adapting a model trained on simulated, computer-generated imagery (CGI) to real-world images. This adaptation requires the model to bridge the gap between synthetic and real-world data, characterized by:

$$\mathcal{D}_{\text{source}} \neq \mathcal{D}_{\text{target}} \tag{2.49}$$

reflecting the transition from CGI to real-world imagery.
In DTM generation, this could be adapting a model trained on urban DSM data to generate DTMs for rural or forested areas. This represents a shift in domain types, from urban landscapes to natural terrains. This highlights the shift from urban to rural terrain types.
This problem can be effectively addressed using domain-adversarial Training. This method trains a model to perform its primary task while also being unable to distinguish between source and target domain data [92]. In the context of depth estimation, this might mean training on both synthetic and real-world images, encouraging the model to learn features that are invariant across these domains. This approach ensures that the depth estimation model maintains high performance regardless of the domain shift from synthetic to real-world environments.

Homogeneous Domain Adaptation

Homogeneous domain adaptation occurs when the feature space remains constant, but data distribution varies. An example of depth estimation would be adapting a model from dense urban environments to sparse rural landscapes. Though both involve outdoor scenes, the density and structure of objects significantly differ, as captured by:

$$P_{\text{source}}(Y|X) \neq P_{\text{target}}(Y|X) \tag{2.50}$$

indicating the variation in object density and structure within the same outdoor environment. In the context of DTM generation, this could involve adapting a model from DSMs of flat terrains to mountainous terrains. Though both involve terrain modeling, the variation in terrain elevation significantly changes the data distribution. This indicates the change in terrain elevation while the feature space of DSM remains constant.

Here, instance weighting is a valuable approach. It involves adjusting the importance of samples in the training process based on their similarity to the target domain. For depth estimation, if a model trained on urban scenes is to be adapted to rural scenes, instance weighting can prioritize urban images that resemble rural settings, thereby aligning the model more closely with the target domain [93].

Heterogeneous Domain Adaptation

Heterogeneous domain adaptation applies when both feature space and data distribution change. In depth estimation, this might involve adapting a model from processing 2D images to processing 3D point clouds. This adaptation encompasses a shift in data type (from 2D to 3D) and distribution, as represented by:

$$\mathcal{F}_{\text{source}}(X) \neq \mathcal{F}_{\text{target}}(X') \tag{2.51}$$

and

$$P_{\text{source}}(X) \neq P_{\text{target}}(X') \tag{2.52}$$

highlighting the shift from 2D image-based features to 3D point cloud features and their respective distributions .

An example in DTM generation is adapting a model from processing optical imagery DSMs to processing LiDAR-based DSMs. This involves a shift in data type (from optical to LiDAR) and distribution. It highlights the shift in both DSM data type and its distribution.

can be tackled through Feature Transformation. This method transforms the feature space of the source domain to align it with that of the target domain [93]. In depth estimation, if a model trained on 2D images is to be adapted for 3D point cloud data, a feature transformation approach can map the 2D image features into a 3D feature space. This transformation allows the depth estimation model to leverage its 2D training in processing 3D data.

This can be tackled through feature transformation. This method transforms the feature space of the source domain to align it with that of the target domain [94]. In depth estimation, if a model trained on 2D images is to be adapted for 3D point cloud data, a feature transformation approach can map the 2D image features into a 3D feature space. This transformation allows the depth estimation model to leverage its 2D training in processing 3D data.

2.4 Depth Estimation

2.4.1 Supervised Learning Methods

CNNs-based Methods

These methods involve learning depth features through convolution kernels and recovering depth maps via deconvolution. For instance, [95] proposed a dual-stream framework integrating depth regression and gradients for enhanced depth map coherence. Absolute and relative depth learning are highlighted, with the former achieving high accuracy and the latter being robust against data homogeneity issues.

RNNs-based Methods

RNN-based approaches capture spatial and temporal information from image sequences. These networks have internal mechanisms to remember and integrate information over time, making them suitable for capturing dynamic changes in a scene. By processing a sequence of images, RNNs can understand the continuity and evolution of the scene, providing a more nuanced depth perception than what might be inferred from a single image. For example, in a monocular video sequence of a driving scenario, an RNN can help infer the depth of various objects like cars, pedestrians, and buildings by observing their size, shape, and movement over time. A car moving away from the observer will gradually decrease in size, and an RNN can learn such patterns to infer depth. Similarly, if the observer's perspective changes, such as moving closer to an object, the RNN can detect the corresponding changes in the object's appearance over the sequence to estimate its depth. Moreover, the integration of RNNs with other neural network architectures, like CNNs, allows for the extraction of both spatial features from individual frames and temporal features across the sequence. This combined approach enables a comprehensive understanding of the scene's spatial-temporal dynamics, enhancing the depth estimation process. DepthNet [96], using ConvLSTM [97] layers, exemplifies this by leveraging temporal data for depth prediction, demonstrating the significance of sequential data processing in depth estimation.

GANs-based Methods

GANs generate depth maps that closely resemble ground truth data. GAN-based approaches for monocular depth estimation are an innovative application of GANs to generate depth maps from a single image. The fundamental architecture of a GAN consists of two main components: a generator and a discriminator. The generator aims to produce depth maps that are indistinguishable from the real depth maps, while the discriminator evaluates whether the depth map is real (ground truth) or synthetic (generated by the generator). In the context of depth estimation, the generator takes an image and attempts to produce a corresponding depth map. The discriminator, on the other hand, receives either the generated depth map or

the real depth map and tries to determine its authenticity. This adversarial process drives the generator to produce increasingly accurate and realistic depth maps over time. [98] utilized a GlobalNet and RefinementNet within a GAN framework to extract global and local features, showing the potential of adversarial training in refining depth estimation.

2.4.2 Unsupervised Learning Methods

Stereo Matching

These methods derive inspiration from traditional stereo vision techniques, where depth is computed by analyzing the disparities between a pair of stereo images. By comparing corresponding pixels in the left and right images, the disparity map is calculated, which in turn provides depth information. The depth network in these methods estimates disparity by aligning features between the stereo images, a process that is inherently geometric. The network is trained to minimize the reconstruction error between the predicted depth map and the observed stereo images, often employing loss functions like L1 loss or Structural Similarity Index Measure (SSIM). For instance, [99] proposed a method that enforces consistency between the depth maps estimated from the left and right images, thereby improving the depth estimation's accuracy. Their approach, which does not rely on ground-truth depth maps, uses a novel loss function that encourages the network to predict depth maps that are consistent when applied to both the left and right stereo images.

Monocular Sequences

The approach to monocular sequences in unsupervised learning for depth estimation leverages sequential images to deduce scene structure and camera motion. This method is profoundly influenced by [100], which simultaneously estimates depth and camera motion, showcasing the capability of unsupervised learning to infer depth from the temporal variations in image sequences.

Zhou et al. [100] developed a framework that utilizes two separate networks, one for depth

estimation and another for camera motion estimation. These networks are trained either jointly or separately, employing reconstruction loss and photometric consistency loss functions. Their research emphasizes the use of geometric constraints built on adjacent frames within a monocular sequence, contrasting with stereo matching that relies on stereo image pairs.

The model operates by analyzing consecutive frames to understand how objects move or change appearance over time, which provides cues about their relative depth. For example, if an object appears larger in successive frames, it's likely moving closer to the camera. By applying this logic across a sequence of frames, the model can deduce the depth map of the scene.

The work [100] is significant as it demonstrates that depth and camera motion can be extracted from monocular sequences without the need for ground truth depth maps. This advancement not only mitigates the dependency on extensive labeled data but also broadens the applicability of depth estimation in scenarios where only monocular video is available. Their methodology offers a pivotal reference for subsequent research in the field, illustrating the potential of monocular sequences in enhancing the understanding of scene depth through unsupervised learning.

2.4.3 Semi-supervised Learning Methods

Combined with Synthetic Data

Incorporating synthetic data helps bridge the gap between synthetic and real-world imagery, enhancing depth prediction accuracy. The DispNet model's use of image style transfer exemplifies how synthetic data can augment depth estimation training [101].

Combined with LiDAR

Integrating LiDAR data introduces real-world depth cues into the learning process, offering a balance between dense ground truth and sparse but accurate LIDAR measurements, as demonstrated by a semi-supervised model [102].

Combined with Surface Normal

By correlating surface normals with depth, this method refines depth estimation, as shown by GeoNet [103], which uses a depth-to-normal network to exploit geometric relationships for improved depth prediction.

2.4.4 Tasks

Single-Task Learning

Regression-based: This approach involves learning scene structure features from inputs and performing regression to estimate continuous depth values that fit the input. Regression methods can directly obtain continuous pixel-level depth values [104].

Classification-based: This approach involves discretizing continuous depth values and learning corresponding classification labels for these discretized values. These classification labels are then combined to produce the final depth map [105].

Multi-Task Learning

There are several MTL approaches in this community, two of which are explained in the following:

Combined with Semantic Segmentation: This method integrates depth estimation with semantic segmentation, sharing encoder networks, and using separate decoders for each task. This approach takes advantage of context information from semantic segmentation to enhance the accuracy of depth estimation [28].

Combined with Other Tasks: This includes integration with tasks like visual odometry and optical flow estimation, particularly in video-based depth estimation scenarios. This approach leverages the complementary nature of different features to improve overall performance [28].

2.4.5 Interactive Refinement

Before diving into the intricacies of our method, it is pivotal to elucidate the foundational concepts of BRS [106] and its advanced iteration, f-BRS [90]. BRS is a framework designed to refine estimations or predictions by integrating additional user inputs or external data points, enhancing the precision and reliability of the output. This system is particularly effective in scenarios where the initial estimations are derived from limited or ambiguous data sources, and the inclusion of user inputs can significantly steer the refinement process towards more accurate outcomes.

Building upon the BRS framework, f-BRS introduces a more nuanced approach by incorporating feature-based data, allowing for a more granular refinement process. In the context of depth estimation, f-BRS leverages sparse point cloud data and user-generated inputs as features to refine the depth map. This feature-based approach not only enhances the fidelity of the depth estimation but also provides a mechanism for users to directly influence the refinement process, ensuring that the final output aligns more closely with the real-world data or user expectations.

In the subsequent sections, we delve into the adaptation of the f-BRS framework for enhancing depth estimation from monocular images, a process that becomes significantly more intricate when amalgamated with sparse point cloud data. Our approach is designed to refine depth estimations by leveraging user inputs alongside sparse point cloud data as constraints, offering a novel methodology in the realm of depth estimation.

Depth estimation initiates with generating an initial depth map D from a monocular image, which is subsequently refined through incremental updates ΔD. These refinements are meticulously guided by user interactions and the integration of sparse point cloud data, underpinning the refinement process. The crux of our method lies in an objective function aimed at minimizing the discrepancy between the predicted depth map and the actual depth measurements, encapsulated by the equation:

$$L = \lambda \|\Delta D\|_2^2 + \sum_{i=1}^{n} w_i (D_{x_i}^{\text{pred}} + \Delta D_{x_i} - d_{x_i})^2 \tag{2.53}$$

Here, L denotes the loss function, designed to harmonize the predicted depth $D_{x_i}^{\text{pred}}$ at any given point i with the actual depth d_{x_i}, sourced either from user input or the sparse point cloud. The weights w_i play a pivotal role, accentuating the confidence in data points, especially those emanating from user inputs or the sparse point cloud. The regularization parameter λ aids in maintaining the balance between the model's complexity and its accuracy.

To achieve the refinement of the depth map, we employ the L-BFGS algorithm, an iterative method tailored for minimizing the objective function with respect to ΔD. This algorithmic choice is instrumental in navigating the complexities of depth map refinement, ensuring a methodical and efficient optimization process.

The implementation phase encompasses developing a user interface tailored for depth-related inputs and integrating sparse point cloud data into the depth estimation framework. Given the large-scale nature of depth map refinement, the efficiency of the optimization algorithm is paramount, ensuring that the system remains responsive and effective.

2.5 Deterministic Interpolation Methods

partial analysis often involves various methods to interpolate or estimate values at unknown locations based on known data points. These methods can be broadly classified into three categories: Triangulation Methods, Gridding Methods, and Geostatistical Methods.

2.5.1 Triangulation Methods

Triangulation methods are based on dividing spatial data into triangles (or simplices in higher dimensions) and use these triangles as the basis for interpolation.

Barycentric Interpolation Barycentric interpolation is a variant of linear interpolation that operates on a simplex, such as a triangle in 2D or a tetrahedron in 3D [107]. It leverages the concept of barycentric coordinates, where the position of a point in a simplex is represented

as the center of mass or barycenter of masses placed at the vertices of the simplex. Suppose we have a simplex defined by $n+1$ points in n-dimensional space (denoted $P_0, P_1, \ldots, P_n$), and each point has a corresponding data value $V_0, V_1, \ldots, V_n$. The interpolated value at any point P within the simplex is found by first calculating the barycentric coordinates $\lambda_0, \lambda_1, \ldots, \lambda_n$ for P. These are calculated to satisfy $\lambda_i \geq 0$ for each i, and the sum of all λ_i's is 1. For a point P within the simplex, the interpolated value V at P is calculated as:

$$V(P) = \lambda_0 V_0 + \lambda_1 V_1 + \ldots + \lambda_n V_n \tag{2.54}$$

In a 2D triangle, for instance, the weights λ_i's are calculated based on the relative areas of sub-triangles formed by point P and the vertices of the original triangle.

Planar Interpolation Planar interpolation is a widely used method particularly suited for data that can be effectively represented by planar segments. This technique involves fitting a plane to a set of points and interpolating within that plane. As described in [108], to interpolate the value at a new point $P(x, y)$, one must first identify the triangle to which P belongs. The process of determining which triangle contains the point involves several steps.

First, Delaunay triangulation is performed on the set of input data points. This triangulation divides the space into a set of non-overlapping triangles, ensuring that no point is inside the circumcircle of any triangle in the triangulation. This property of Delaunay triangulation helps in forming well-shaped triangles that are optimal for interpolation.

Next, spatial search algorithms, such as the walking triangle method or point location queries using a spatial index, are used to identify the triangle within the Delaunay triangulation that contains the point $P(x, y)$. These algorithms efficiently navigate the triangulated space to find the candidate triangle that may contain the point.

Once a candidate triangle is identified, the point's barycentric coordinates are computed relative to its vertices. The barycentric coordinates (u, v, w) are calculated, and if they satisfy $u \geq 0$, $v \geq 0$, and $u + v \leq 1$, then the point lies within the triangle. This verification step ensures that the point is correctly located within the identified triangle.

With the containing triangle identified, planar interpolation proceeds by fitting a plane

to the vertices of this triangle. Consider a triangle with vertices $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, and $P_3(x_3, y_3)$, and corresponding values z_1, z_2, and z_3. The plane equation is of the form $z = ax + by + c$. By inserting the known points (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3) into this equation, one can solve a system of linear equations to determine the coefficients (a, b, c). With these coefficients, the value z for any arbitrary point $P(x, y)$ within the triangle can be interpolated using the plane equation.

2.5.2 Gridding Methods

Gridding methods are utilized in various fields, such as geostatistics, meteorology, and GIS, for the purpose of spatial interpolation. These methods involve the creation of a regular grid over an area of interest, estimating values at grid points without relying on the formation of triangles, as seen in triangulation methods [109].

Inverse Distance Weighting (IDW) is a method where values at unknown points are estimated based on the values of known points. In IDW, closer points have more influence than those further away, with this influence diminishing as a function of distance. The basic formula for IDW is expressed as:

$$V(x) = \frac{\sum_{i=1}^{n} \frac{V_i}{d_i^p}}{\sum_{i=1}^{n} \frac{1}{d_i^p}} \tag{2.55}$$

In this formula, $V(x)$ represents the estimated value at the unknown point. The known values from surrounding points are denoted by V_i, and d_i is the distance between the unknown point and each known point. The parameter p is a power parameter that influences how weight decreases with distance, where a higher value of p places more emphasis on nearer points.

Splines, particularly in spatial interpolation, refer to a method for creating a smooth curve through a set of points. A common type of spline used for gridding is the cubic spline, which is formulated as:

$$S(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{2.56}$$

Here, $x_i \leq x < x_{i+1}$ and a_i, b_i, c_i, and d_i are coefficients that are determined based on the known points and boundary conditions.

Radial Basis Functions (RBF) are employed for interpolation by creating a smooth surface through the known points. They are especially useful for irregularly spaced data in multiple dimensions. The basic form of an RBF interpolation is:

$$F(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) \tag{2.57}$$

In this formula, $F(x)$ denotes the interpolated value at the point x. The coefficients λ_i are solved for using the known values. The radial basis function, ϕ, is a function of the distance between the unknown point x and the known point x_i. Common choices for ϕ include Gaussian, Multiquadric, and Inverse Multiquadric functions.

Overall, these gridding methods are advantageous for handling scattered data points and creating a continuous surface or grid, which is useful for visualization, analysis, or further processing. They offer a structured approach that can be more straightforward to implement and interpret in regular, rectangular areas compared to triangulation methods.

2.5.3 Geostatistical Interpolation Methods

Kriging, and particularly Ordinary Kriging, is a geostatistical method extensively used in spatial data interpolation. It operates on the principle that spatial points closer to each other are more likely to be similar than those farther apart, a phenomenon known as spatial correlation. A semivariogram is employed to quantify this spatial correlation [110]. The semivariogram, denoted as $\gamma(h)$ for a lag distance h, is defined as:

$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} [z(x_i) - z(x_i + h)]^2 \tag{2.58}$$

where $z(x_i)$ is the value at location x_i, $N(h)$ represents the number of pairs of points separated by the distance h, and $z(x_i + h)$ is the value at a location h units away from x_i.

Ordinary Kriging predicts the value at an unsampled location $P(x, y)$ as a linear combination of known values from surrounding locations. The predicted value at point P, denoted as $z(P)$, is calculated as a weighted sum of the known values z_i at sampled points, expressed as:

$$z(P) = \sum_{i=1}^{n} \lambda_i \cdot z_i \tag{2.59}$$

Here, λ_i are the weights assigned to the known values z_i, and n is the number of sampled points utilized in the prediction. These weights are computed based on the semivariogram model such that the estimate at P is unbiased and has the minimum variance. This involves solving a system of linear equations, known as the Kriging system, given by:

$$\sum_{j=1}^{n} \lambda_j \gamma(x_i, x_j) = \gamma(x_i, P), \quad i = 1, 2, \ldots, n \tag{2.60}$$

Additionally, the condition for unbiasedness, which requires that the sum of the weights equals one, is:

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{2.61}$$

The goal is to minimize the variance of the prediction error, where the semivariogram model plays a crucial role in determining how the spatial structure of the data influences the weights. This minimization, combined with the unbiasedness condition, ensures that the Kriging estimator does not systematically overestimate or underestimate the true values. Ordinary kriging is widely applied in fields like mining, hydrology, and environmental science for accurate spatial interpolation of data such as mineral concentrations, water quality, or pollution levels.

Delaunay triangulation is known for its robustness and is particularly suitable for irregularly spaced data. This method involves triangulating the data points to create a network of triangles, making sure that no point lies inside the circumcircle of any triangle. The key advantage of Delaunay triangulation is its adaptability to the distribution of data points, which is beneficial for datasets with irregular spacing. However, this method assumes linear variation within the triangles, which can be a significant limitation, especially in areas where the actual data may exhibit nonlinear variation.

IDW is another widely used interpolation method. Its simplicity and computational efficiency make it an attractive choice for quick analyses. IDW estimates the value at an unsampled point based on the weighted average of values from nearby sampled points, where the weights decrease as the distance increases. The performance of IDW is sensitive to the choice of the power parameter, which determines the rate at which weights decrease with distance. The method's effectiveness is also influenced by the density and distribution of data points, and it may not perform well in sparsely populated areas.

Kriging stands out as a statistically rigorous method that accounts for spatial correlation among data points. Unlike simpler methods, Kriging not only interpolates the data but also provides estimates of the uncertainty of these predictions. The method models the spatial structure of the data through a semivariogram, which is essential for capturing how spatial correlation decreases with distance. However, Kriging requires a well-defined semivariogram, which can be challenging to determine, particularly with limited or noisy data. Additionally, due to its computational intensity, Kriging is less suitable for rapid analyses or very large datasets, compared to simpler methods like IDW.

2.6 Quality Assessment Criteria

Our study employs a detailed framework for quality assessment, encompassing three primary categories: predictive error metrics, classification accuracy metrics, and profile analysis metrics. Each category is tailored to evaluate different aspects of our model's performance with precision and depth.

2.6.1 Predictive Error Metrics

This category is central to understanding the accuracy of our predictions. It includes traditional metrics like the Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE). RMSE measures the square root of the average squared differences between predictions and actual values, while MAE calculates the average of the absolute differences

[28]. Additionally, we incorporate the following metrics:

$$RMSE = \sqrt{\frac{1}{T}\sum_{i \in T} ||d_i - d_i^{gt}||^2} \tag{2.62}$$

$$RMSE_{log} = \sqrt{\frac{1}{T}\sum_{i \in T} ||\log(d_i) - \log(d_i^{gt})||^2} \tag{2.63}$$

$$AbsRel = \frac{1}{T}\sum_{i \in T} \frac{|d_i - d_i^{gt}|}{d_i^{gt}} \tag{2.64}$$

$$SqRel = \frac{1}{T}\sum_{i \in T} \frac{||d_i - d_i^{gt}||^2}{d_i^{gt}} \tag{2.65}$$

These additional metrics, $\text{RMSE}_{\log}$ (2.63), AbsRel(2.64), and SqRel(2.65), offer a more nuanced analysis of prediction errors, accounting for logarithmic differences and relative errors. The accuracies are defined as the percentage of d_i subject to a threshold δ:

$$\text{accuracies} = \%\text{ of } d_i \text{ subject to } \max\left(\frac{d_i}{d_i^{gt}}, \frac{d_i^{gt}}{d_i}\right) = \delta \tag{2.66}$$

Here, d_i and d_i^{gt} represent the predicted and ground-truth depths, respectively, and T is the total number of pixels in all evaluated images.

To offer a comprehensive measure of system performance by integrating key dimensions into a single indicator performance index (PI) is used. This index is crucial for evaluating and enhancing the reliability and efficiency of various systems, as it provides a holistic view that incorporates both consistency and accuracy of performance.

To construct the PI, a linear combination of the standard deviation (Std) and the average error (AE) is employed. These two metrics are fundamental in reflecting the system's stability and accuracy. The standard deviation expresses the consistency of the system's performance, while the average error highlights the system's deviation from expected outcomes. This integration offers a balanced perspective on system performance, making the PI a valuable tool for system analysis [111].

The formula for the PI is given by:

$$PI = a \cdot Std + b \cdot AE \tag{2.67}$$

where a and b are coefficients that determine the relative importance of the Standard Deviation and the average error, respectively. These coefficients are adjustable, allowing the Performance Index to be customized to emphasize either consistency or accuracy based on the system's specific requirements and objectives.

Tailoring the PI involves choosing appropriate values for the coefficients a and b, which is crucial for aligning the index with the system's evaluation framework. This customization ensures that the PI remains relevant and targeted to the specific goals of the system assessment.

In contexts where both consistency and accuracy are equally important, setting both a and b to 1 provides a balanced approach. This equal weighting simplifies the PI to a sum of the standard deviation and the average error, offering an effective and straightforward measure of system performance. By adopting this approach, the evaluation ensures that both the system's variability and accuracy are considered simultaneously, providing a comprehensive overview of the system's overall stability and efficiency.

2.6.2 Classification Accuracy Metrics

This category encompasses a set of metrics crucial for evaluating binary classification models. Each metric offers insights into different aspects of model performance [112].

Type I Error (False Positive Rate) This metric measures the proportion of negative instances that are incorrectly classified as positive. It is defined as:

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{2.68}$$

In this equation, FP represents false positives and TN represents true negatives.

Type II Error (False Negative Rate) This represents the proportion of positive instances that are wrongly classified as negative. Mathematically, it is expressed as:

$$\text{False Negative Rate} = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{2.69}$$

Here, FN denotes false negatives and TP denotes true positives.

Intersection over Union (IoU) Commonly used in image segmentation, IoU measures the overlap between the predicted segmentation and the ground truth. It is calculated as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{2.70}$$

The Area of Overlap is the intersection of the predicted and actual segmentations, while the Area of Union is their union.

General Accuracy This is the most straightforward metric, representing the proportion of correctly classified instances (both true positives and true negatives) among all instances:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.71}$$

2.6.3 Profile Analysis Metrics

In this category, the focus is on analyzing the cross-sectional profile properties of the data, where a line is extracted from the depth image, and its characteristics are evaluated.

Correlation Coefficients These coefficients measure the strength and direction of the linear relationship between two variables. The Pearson correlation coefficient, for example, is defined as:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{2.72}$$

where X_i and Y_i are the individual sample points indexed with i, and $\bar{X}$ and $\bar{Y}$ are the mean values of the variables X and Y, respectively.

Earth Mover's Distance (EMD) is a measure of the distance between two probability distributions over a region D. It's defined as the cost of transforming one distribution into the other and is particularly useful in comparing various aspects of profiles. The formal definition involves solving an optimization problem [113]:

$$EMD(P, Q) = \inf_{\gamma \in \Gamma(P,Q)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{2.73}$$

where P and Q are the two distributions, and γ denotes a transport plan between them.

This comprehensive set of metrics ensures a robust and multi-dimensional assessment of the quality of our predictive models, facilitating a thorough understanding of their performance across various dimensions.

2.6.4 SLAM Evaluation Metrics

Absolute Trajectory Error (ATE) ATE evaluates the positional accuracy of the entire estimated trajectory against the ground truth trajectory. It is often computed as the RMSE between corresponding pose points in the estimated and ground truth trajectories. Mathematically, the ATE is given by:

$$\text{ATE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|p_i - q_i\|^2} \tag{2.74}$$

where N represents the number of corresponding pose points in the trajectories, p_i and q_i are the position components of the i-th pose in the estimated and ground truth trajectories, respectively, and $\|p_i - q_i\|$ is the Euclidean distance between the corresponding pose points [114].

Relative Trajectory Error (RTE) RTE measures the discrepancy in the relative motion between estimated and ground truth trajectories. It is evaluated over segments of varying lengths (e.g., 100, 200, ..., 800 meters) to assess the consistency of motion estimation. The RTE is calculated as:

$$\text{RTE} = \sqrt{\frac{1}{M}\sum_{j=1}^{M} \|(p_{i+j} - p_i) - (q_{i+j} - q_i)\|^2} \tag{2.75}$$

Here, M is the number of segments considered for the evaluation, $p_{i+j} - p_i$ and $q_{i+j} - q_i$ represent the relative positions between two poses separated by a certain distance in the estimated and ground truth trajectories, respectively [114].

Relative Rotation Error (RRE) RRE quantifies the angular deviation in the estimated rotations compared to the ground truth. It is crucial for evaluating the rotational accuracy

of the SLAM system. The RRE is computed as:

$$\text{RRE} = \frac{1}{M} \sum_{j=1}^{M} \frac{\|\theta_{i+j} - \theta_i - (\phi_{i+j} - \phi_i)\|}{d_{i,j}} \tag{2.76}$$

In this equation, $\theta_{i+j} - \theta_i$ and $\phi_{i+j} - \phi_i$ are the relative rotations between two poses in the estimated and ground truth trajectories, respectively, $d_{i,j}$ is the translational distance between the poses i and $i + j$, and $\|\theta_{i+j} - \theta_i - (\phi_{i+j} - \phi_i)\|$ measures the absolute difference in rotation between the estimated and ground truth trajectories [114].

2.7 Summary

This chapter meticulously discusses the foundational elements and mathematical principles that underpin the components used in the subsequent chapters. It provides a comprehensive overview of optimization techniques in theory and practice. Deep learning, including its architecture, training processes, and relevance to the study's objectives, is explored in detail. The chapter also covers depth estimation, explaining the algorithms and methodologies used to estimate depth information from various data sources. Interpolation methods are examined, highlighting their significance in enhancing data resolution and accuracy. Lastly, the chapter addresses quality assessment criteria, outlining the standards and metrics used to evaluate the performance and effectiveness of the proposed solutions.

Chapter 3

Model Agnostic Depth Estimation

3.1 Introduction

Dense depth maps are integral to the functionality of autonomous mobile robots, contributing significantly to core processes such as SLAM [115], visual odometry [116], and object detection [117] The integration of DLtechniques has marked a substantial shift in the generation of these maps, mirroring its influence across various technological domains [54].

DL-based methods for depth generation are broadly classified into depth completion and estimation methodologies. Depth completion focuses on enriching sparse depth maps, filling in missing information to create a more comprehensive depth understanding [118, 119]. In contrast, depth estimation methods are tasked with deriving depth information from monocular images, assigning depth values to each pixel based on learned patterns and features [120, 121, 122]. Both strands of methodologies employ supervised or unsupervised learning paradigms, with supervised approaches notably achieving higher accuracy and reliability in various applications [123, 124, 125].

In the realm of supervised learning, depth completion methods predominantly use ground truth or annotated depth (AD) maps as reference points for learning [126, 118]. Supervised depth estimation methods, however, may utilize either AD maps [123, 127] or sparse depth (SD) maps [128] as their training backbone. The choice of training data significantly influences

the learning outcome and the applicability of the methods in various scenarios.

Despite the higher accuracy of supervised depth completion methods, their reliance on a large volume of AD maps poses a challenge. The collection of such data in real-world settings is not only costly but also time-intensive, making it a less feasible option in certain applications [101, 122]. Consequently, depth estimation methods, especially those not requiring extensive annotated data, become more viable in scenarios where acquiring large-scale, accurate training data is impractical. Among depth estimation techniques, those based on SD maps are particularly noteworthy despite their relative inaccuracy compared to AD-based methods. Their advantage lies in the elimination of the need for ground truth or AD, presenting a more accessible solution in diverse environments [105, 99].

However, SD-based methods are not without their challenges. Like many deep neural networks, they face the issue of computational intensity, which can hinder their quick and efficient adaptation to new environments or datasets [129, 61]. This computational burden is a critical factor in determining the practicality of these methods in dynamic and varying operational contexts.

To mitigate these challenges, this paper introduces an inference-time functionally adaptive network, leveraging the concept of functionally adaptive networks to enhance adaptability and efficiency [130, 131]. Specifically, the network employs f-BRS to facilitate adaptability to new environments, which can be crucial in scenarios involving diverse networks and datasets [90, 132]. This approach allows for modifications in the network's activation functions during inference, enhancing its flexibility and responsiveness.

However, f-BRS has inherent limitations, particularly in handling highly distorted outputs, which is a common challenge in depth estimation tasks. To overcome this, a distance-based correction (DBC) module is proposed, aimed at performing an initial coarse refinement before applying f-BRS. This step ensures that the depth maps are sufficiently accurate before further refinement. Additionally, to address the tendency of f-BRS to converge to local optima, the scheme is augmented with a global optimizer, specifically particle swarm optimization [53]. This integration not only enhances the depth estimation accuracy but

also contributes to the overall robustness of the method. The resulting scheme, termed the depth adaptive refinement scheme (DARS), represents a significant advancement in the field of depth estimation, particularly in scenarios where adaptability and computational efficiency are paramount.

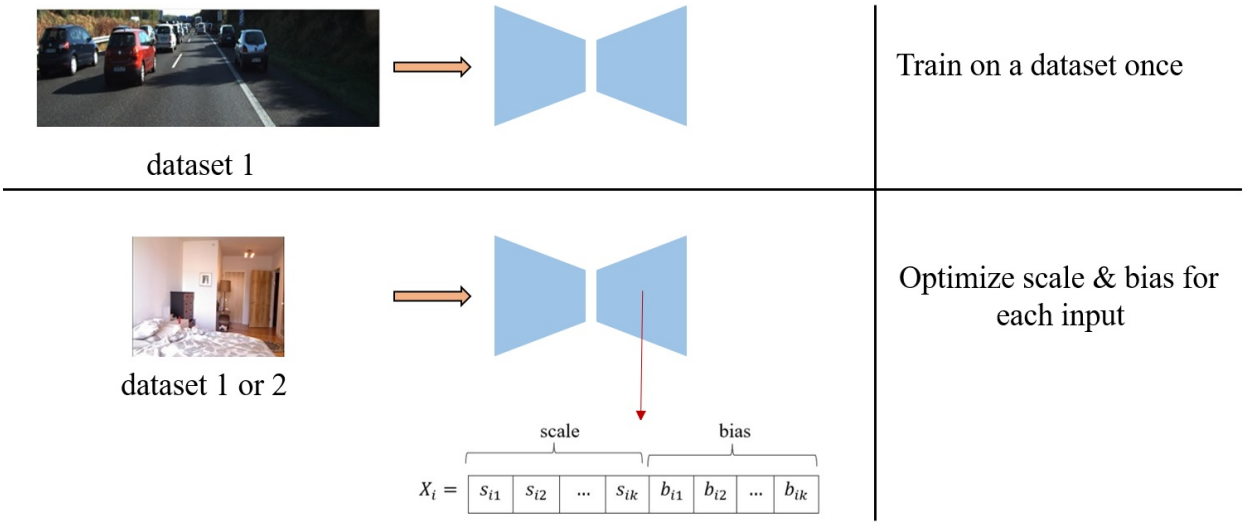


Figure 3.1: Overall performance of the proposed scheme. Keeping the pre-trained weights fixed, channel-wise scales and biases are applied to an intermediate activation map and are optimized to improve the predictions for the input dataset. The input dataset can be identical to the training dataset or a new one.

In summary, our work makes the following contributions:

- A novel inference-time functionally adaptive scheme for monocular depth estimation networks. The proposed scheme needs neither offline data gathering nor offline training, because it uses available pre-trained weights. Note that the weights on one dataset can be used for any other in-domain dataset, while the predicted depth maps are updated through SD maps.

- Using f-BRS for the first time in the field of depth generation.

- A model-agnostic scheme that can be plugged into any baselines. In this article, we selected monodepth2, one of the most widely used for depth estimation, as the baseline.

3.2 RELATED WORK

Depth estimation has been studied for several years. Here, we concentrate on the state-of-the-art unsupervised and supervised methods. In the last part, a brief review of functionally adaptive networks is brought.

3.2.1 Unsupervised Depth Estimation Methods

Unsupervised depth estimation represents a significant stride in the field of computer vision, particularly in the development of depth perception models that do not rely on labeled datasets. These methods principally employ color consistency loss as a learning signal, utilizing either stereo images [100], temporal image sequences [133], or a combination of both [132]. The fundamental idea is to enable a monocular depth model to infer depth information by understanding the changes in pixel values across different views or over time.

A variety of innovative techniques have been introduced to enhance the self-supervision mechanism in these methods. One such technique is the left-right consistency loss, which ensures that the depth predictions are consistent when applied to both the left and right images of a stereo pair [99]. Another advancement is the incorporation of temporal depth consistency, which leverages the temporal relationship between consecutive frames in a video sequence to improve depth estimation [134]. Furthermore, cross-task consistency, which aligns depth estimation with other tasks like optical flow or scene segmentation, has also been explored to enhance the robustness of the learning process [135, 136, 137].

Among the various unsupervised methods, monodepth2 has emerged as a particularly noteworthy model due to its unique combination of techniques and modifications [132]. This method has demonstrated significant improvements in depth estimation accuracy and has influenced subsequent research in the field.

Despite these advancements, unsupervised depth estimation methods still face certain limitations. One of the primary challenges is their application scope, which is often restricted to specific environments. For instance, most existing methods are tailored either for outdoor scenarios [100, 99, 134] or indoor settings [138], but rarely both. This specialization limits their versatility and applicability in diverse real-world situations. Additionally, compared to supervised methods, unsupervised techniques generally exhibit lower accuracy in depth predictions. This limitation is primarily due to the absence of ground truth data, which can guide and refine the learning process in supervised settings [99].

In response to these challenges, recent research has focused on developing hybrid methods that integrate unsupervised learning with minimal supervision or other auxiliary tasks to bridge the gap in accuracy while maintaining the flexibility of unsupervised approaches [139]. These hybrid methods aim to leverage the strengths of both supervised and unsupervised learning paradigms, potentially leading to more robust and versatile depth estimation models suitable for a wider range of applications.

3.2.2 Supervised Depth Estimation Methods

Supervised depth estimation methods have been a foundation in the advancement of depth perception technologies. These methods predominantly fall into two categories: AD-based and SD-based, each with its unique approach and challenges.

AD-based Depth Estimation Methods: AD-based methods require AD maps for training. Contrary to depth completion techniques, the primary input for these methods is images, not depth maps. This approach allows the network to learn depth information directly from visual inputs. AD-based methods, exemplified by AdaBins [123] and BTS [127], focus on minimizing the error between the predicted depth maps and the available AD maps. These methods have shown promising results in accurately estimating depth, benefiting from the detailed and precise depth information provided by AD maps.

However, one significant limitation of AD-based methods is their dependency on AD maps for training. Obtaining such detailed annotated maps can be labor-intensive and costly,

limiting the scalability and practicality of these methods in diverse application scenarios. Furthermore, the need for AD maps restricts the applicability of these methods in environments where such data is unavailable.

SD-Based Depth Estimation Methods: In contrast, SD-based methods leverage SD maps for training. The advantage of these methods lies in the ease of acquiring training data. Modern robotic and mapping systems are capable of simultaneously capturing images and their corresponding SD information, making the data collection process more feasible. SD-based methods utilize these SD maps to enhance the loss functions, improving the training process and depth estimation accuracy [140, 141, 142].

The training process for SD-based methods involves minimizing the discrepancy between the estimated depth map and the SD data. This approach not only simplifies the training data requirements but also introduces an element of self-supervision, leading to these methods often being categorized as semi-supervised [28]. The semi-supervised nature of SD-based methods offers a balanced approach, combining the benefits of supervised learning with the flexibility of unsupervised methods.

Despite their advantages, SD-based methods also face challenges. The primary issue is the inherent sparsity of the depth data, which may not provide sufficient information for accurate depth estimation across all scenarios. Additionally, these methods might struggle in environments where the available SD data is not representative of the entire scene, leading to inaccuracies in depth prediction.

In conclusion, both AD-based and SD-based supervised depth estimation methods offer valuable approaches to depth perception, each with its strengths and limitations. AD-based methods excel in accuracy but are hindered by the need for extensive annotated data, while SD-based methods offer greater flexibility in data requirements but may face challenges in depth estimation accuracy and comprehensiveness. The choice between these methods largely depends on the specific requirements and constraints of the application at hand.

3.2.3 Functionally Adaptive Neural Networks

Functionally adaptive neural networks have gained prominence for their ability to adapt dynamically to new inputs and changing environments [143, 144]. These networks are capable of modifying their structure or parameters, which is crucial for applications that require flexibility and responsiveness.

Adaptation Techniques in Neural Networks

Weight Modification: The adaptation in neural networks can be achieved through weight modification, which involves optimizing the weights for new inputs. This process typically employs backpropagation and gradient descent methods, expressed as:

$$\Delta w = -\eta \frac{\partial \mathcal{L}}{\partial w} \tag{3.1}$$

where Δw is the change in weights, η represents the learning rate, and $\frac{\partial \mathcal{L}}{\partial w}$ is the gradient of the loss function $\mathcal{L}$ with respect to the weights w.

Recent advancements in this domain have been noteworthy. For instance, the work by [145] introduced an innovative approach to weight modification that leverages advanced optimization techniques, resulting in faster convergence and improved performance on complex tasks. Additionally, another significant contribution by [146] demonstrated a novel adaptive learning rate algorithm that dynamically adjusts the learning rate based on network performance, further enhancing the efficiency of weight modification in neural networks.

These state-of-the-art approaches underscore the ongoing evolution in the field of neural network adaptation, highlighting how sophisticated techniques in weight modification continue to push the boundaries of what these powerful computational models can achieve.

Functional Adaptation: Another method of adaptation is functional adaptation, which involves altering the activation functions' slope and shape through additional parameters, as seen in PReLU:

$$PReLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{3.2}$$

In this case, α is a learnable parameter that adapts during the training process.
Compared to PReLU, TAAFs [147] provide a more generalized and flexible approach by incorporating four adaptive parameters that allow for horizontal and vertical scaling and translation of any inner activation function. TAAFs [147] extend beyond the simple adaptability of PReLU, offering a comprehensive method to dynamically adjust activation functions and covering a broader range of adaptation possibilities. This flexibility enables TAAFs to model both adaptive and non-adaptive activation functions, serving as a robust tool for enhancing neural network performance across diverse applications and contexts.

Applying this concept to the Softplus activation function, the Parametric Softplus (PSoftplus) illustrates the adaptability of TAAFs. PSoftplus, defined by two fixed parameters for scaling and translation, can be understood within the TAAF framework. This demonstrates how TAAFs can be adapted to include functions such as Softplus, further improving neural networks' adaptability and performance by utilizing TAAFs' capacity to dynamically tailor activation functions [148].

Network Adaptation

Training Stage Adaptation: During the training stage, adaptive neural networks adjust their parameters based on the training dataset [130]. This stage is critical for the initial learning and calibration of the network.

Inference Stage Adaptation: During the inference stage of real-time applications such as interactive segmentation or SLAM, it is essential to incorporate adaptation techniques. This is particularly important due to the presence of sparse ground truth data that is generated on the fly. Effective adaptation strategies enable these applications to operate in real time while providing accurate and reliable results. [144].

Application-Specific Adaptation

Neural networks can be adapted for specific applications. Two primary modes are single-image Adaptation and sequence-image adaptation. Both approaches have unique characteristics

and are suited for different types of tasks. A network can be adapted to either a sequence of images or a single image. In the context of single-image adaptation, the primary advantage lies in optimizing the prediction for a particular image or an individual object. After this optimization, the adaptation is not retained for subsequent images. This approach, denoted in the literature as [144], can be particularly advantageous in scenarios where the scenes are subject to significant variations.

f-BRS in Functional Adaptation

A notable example of functional adaptation is the f-BRS [90]. f-BRS, as discussed in section 2.4.5, is a method specifically designed to adjust the network's activations in response to user interactions or other real-time inputs during the inference stage. It backpropagates through the network, refining the output and intermediate feature representations to ensure precision and contextual relevance. This scheme is particularly effective in single-image adaptation scenarios, where immediate adjustments are crucial for accuracy.

In conclusion, functionally adaptive neural networks, particularly with the incorporation of methods like f-BRS, represent a significant advancement in the field of AI. These networks demonstrate a remarkable ability to evolve and adapt, offering tailored solutions for a wide range of dynamic applications and environments.

3.3 Operational Framework

Here, we delve into how foundational theories are put into action. This section is about bridging the gap between abstract concepts and their tangible applications, particularly focusing on how the operational framework leverages techniques such as f-BRS, PSO, and interpolation. We will outline how each component functions within the system, illustrating their roles and synergies. This section aims to provide a clear and accessible explanation of how these theoretical elements are practically employed to achieve the desired outcomes in real-world scenarios.

3.3.1 f-BRS

f-BRS, as discussed in [90], is an optimization scheme that updates activation responses while network weights are fixed. This is done by applying channel-wise scale and bias parameters on activation responses of an intermediate layer in the network, which results in a nonlinear effect on the final output. Given an input image x, the output of the intermediate layer can be denoted as $H(x)$. Then the network head g outputs $\hat{g}(x, s, b) = g(s.H(x) + b)$ where . denotes channel-wise multiplication. Since the scale and bias parameters (p) are learnable, the following optimization problem can be formulated:

$$f(\hat{g}(x, p + \Delta p), l) \rightarrow \min_{\Delta p}. \tag{3.3}$$

where f, Δp, and l are, respectively, the cost function, corrections applied on the bias and scale parameters and labels. This optimization problem is solved in the original version using the L-BFGS optimizer. However, PSO is used in this study since L-BFGS is trapped in local optima due to its local search nature.

3.3.2 PSO-oriented f-BRS

PSO is a population-based stochastic optimization technique, which is explained in section 2.2.3. PSO has two main components, particles and a loss function, which need to be specifically defined for each application. In this application, the goal of which is the estimation of layers' biases and scales, each particle has the potential of solving the problem; this means it must contain all the arguments needed for the problem in question. Given f-BRS for an intermediate layer with a k-channel activation response, each particle X_i comprises $2k$ parameters shared equally between the scale and the bias (see Figure 3.1). These are parameters that PSO tries to find their optimum values based on our objective function, and the way discussed in section 2.2.3.

3.3.3 Delaunay-based Interpolation (DBI)

The first step of the interpolation is to conduct triangulation. Considering that there are many different triangulations for a given point set, we should obtain an optimal triangulation method, avoiding poorly shaped triangles. Delaunay triangulation method has proved to be the most robust and widely-used triangulation approach. This method connects all the neighboring points in a Voronoi diagram to obtain triangulation [149].

To find the value of any new point by interpolation, its corresponding triangle in which it lies should be identified, and subsequently, planar interpolation (section 2.5.1) should be done (Figure 3.2). This interpolation can be used to estimate the correction for each point provided that the left side of the equation, in equation **??**, z, is replaced with the correction Δz. As a relevant example to this study, this correction can be the difference between predicted and target depth values.

3.4 PROPOSED METHOD

3.4.1 DARS

One of the problems with the reviewed depth estimation methods, including monodepth2 [132] as our baseline, is that they need to be retrained whenever the dataset is changed. To address

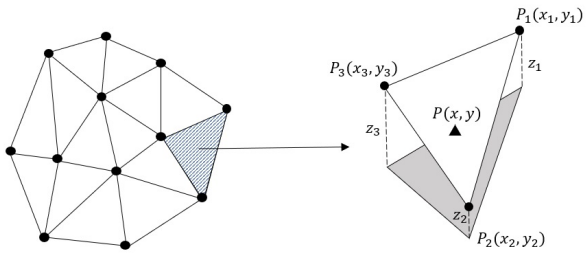


Figure 3.2: Delaunay-based interpolation on a set of points. First, a Delaunay triangulation is done on the points. Then, a plane is fitted to each triangle, and finally, the value for points on each of them is obtained based on the fitted plane.

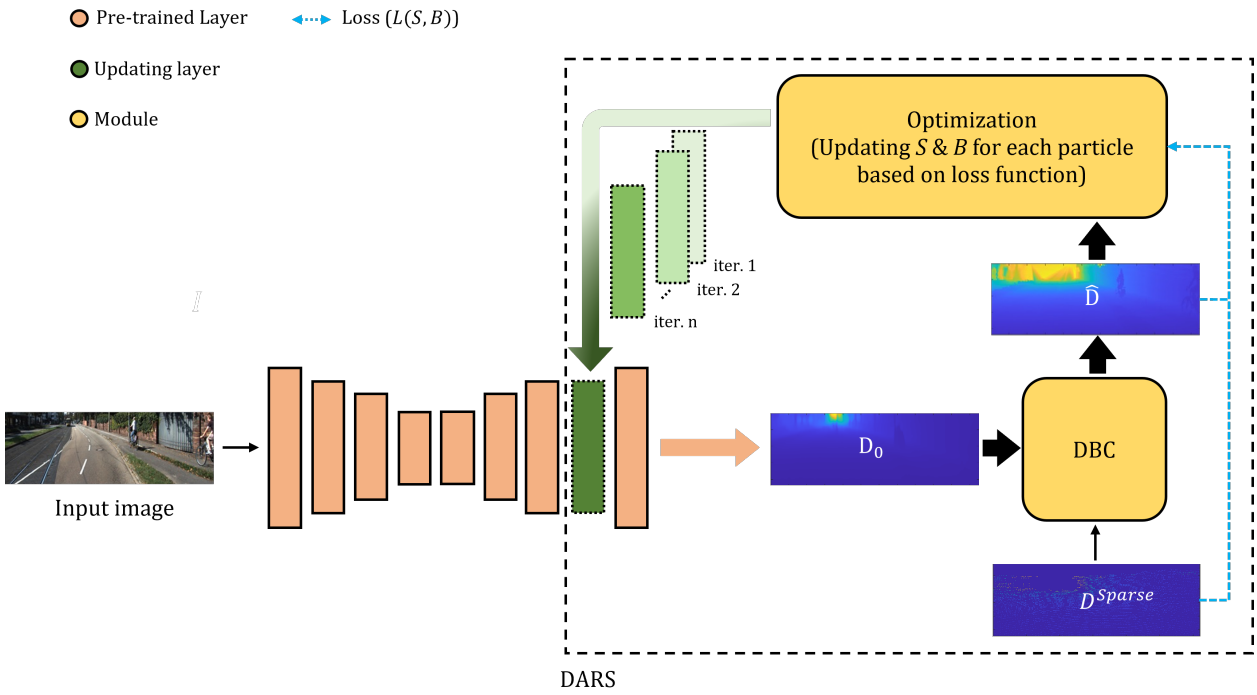


Figure 3.3: The double-stage adaptive refinement scheme (DARS). DARS refines depth maps estimated by baselines through optimizing (green arrows) an intermediate set of feature maps during inference (red arrows). Hence, given a pretrained baseline with fixed weights and an RGB input image, an initial depth D_0^d is estimated. In the first stage (correction), a Sliced Delaunay Correction (SDC) module corrects D_0^d with the guidance of an SD map D^s. Afterward, an optimizer module (PSO) tries to update the intermediate feature maps to minimize the distance between the dense corrected depth $\hat{D}^d$ and D^s. As the scheme can be used for any deep architecture, the baseline (here, monodepth2) is illustrated minimally.

this problem, a novel monocular depth estimation method based on the fusion of f-BRS and monodepth2 is proposed (Figure 3.3). The presence of f-BRS enables the network to function properly when the data changes. This is to say that available weights for the pre-trained network (monodepth2), coming from a specific dataset, can be used for other datasets. This significantly reduces the computational burden needed for retraining. Nonetheless, f-BRS suffers from two problems which are addressed by correction and optimization modules in the proposed method.

The first problem is related to the nature of f-BRS, dealt with in the correction module. As implied in [90], f-BRS can do small refinement due to its simple parameters, biases, and scales. However, a major refinement is needed to fine-tune our unsupervised baseline, specifically for a new dataset. Accordingly, a module is leveraged to make a rough improvement in each iteration. As a result of this, there would be a product with marginal distortions (D_0) having the potential to be improved by f-BRS. For this purpose, in this article, a distance-based correction (DBC) is introduced. In DBC, the SD map (D^{sparse}) is first divided into three overlapped slices, which can also represent regular segments based on the distance from the LiDAR scanner (Figure 3.4). Afterwards, a DBI is applied to each segment to calculate each pixel's correction (section 3.3.3). For the pixels in overlapped areas (Figure 3.4), the average of two values coming from two adjacent segments is considered as the final depth. Thanks to this stage, a corrected depth ($\hat{D}$) is generated, yet with marginal distortions.

The second problem is associated with the default optimizer of f-BRS, coped with in the optimization module. f-BRS originally uses the L-BFGS optimization algorithm. The problem of this optimizer, stemming from its local gradient-based nature, is that it is trapped in local optima. To overcome this problem, L-BFGS is replaced with PSO. As can be seen in Figure 3.3, f-BRS is applied to the result of DBC in each iteration. Thus, PSO iteratively updates scale (S) and bias (B) parameters in each particle based on the below loss:

$$L(S,B) = \sqrt{\frac{1}{T}\sum_{i\in T} ||log(\hat{D}) - log(D^{sparse})||^2} \tag{3.4}$$

where T is the total number of pixels with depth values in D^{sparse}.

3.4.2 PanoDARS

Building on the principles outlined in the DARS, PanoDARS is introduced to apply to panoramic images. As discussed in [32], in the SLAM application, PanoDARS enhances the number of ORB visual features augmented with depth, particularly in the overlapping region (OR_t), see Figure 3.6, between the panoramic images and LiDAR point clouds.

Unlike the original DARS, which was optimized for perspective images, PanoDARS adapts models pre-trained on images with perspective geometry for use in panoramic contexts. This adaptation is crucial due to the lack of a standard dataset for panoramic depth estimation and the challenges posed by self-supervised methods in panoramic geometry. Therefore, the pre-trained weights of Monodepth2 on the KITTI dataset for monocular perspective images serve as our baseline for inference.

PanoDARS omits the slicing approach used in DARS during the correction phase due to the differing sparsity patterns of LiDAR data and the shape of the overlapping region in this dataset (see Figure 3.5). This is evident in the blue rainbow area, which sparsely covers a small portion of the images. Depth estimation in PanoDARS is confined to regions of $\mathbf{I}_t$ where SD data is available ($OR_t = {}^{l}\varphi_t \cup {}^{c}\varphi_t$), as illustrated in Figure 3.6.

The densification method used in DARS, as illustrated in Figure 3.4, is not well-suited for rainbow densification. This is attributable to several factors, discussed in the following. Firstly, the overlapped method attempts to maintain global consistency by averaging data across the overlapping regions. This approach, while beneficial in areas with a good distribution of data points, can become problematic in regions where the data is sparse. The averaging process inherent in the overlapped method could inadvertently smooth out significant local variations that are distinctly captured by the few sparse LiDAR points available. These local features, which are crucial for an accurate representation of the terrain, might be lost or diluted when the interpolation process extends over larger areas to stitch them together in a globally coherent manner. As a result, the model generated using this method could fail to represent the subtleties of the terrain, leading to a less reliable map of the area.

Moreover, the use of an independent approach for each section, as opposed to the

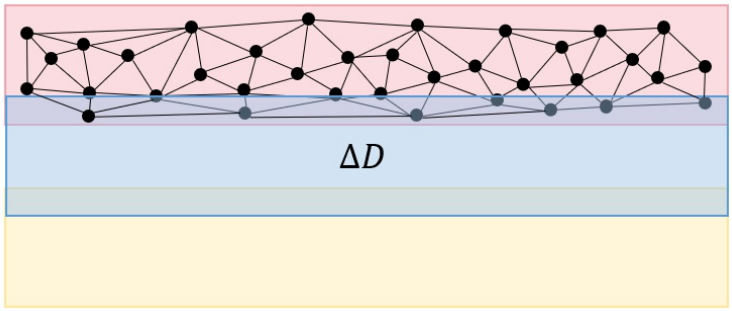


Figure 3.4: Distance-based correction (DBC). Each SD map is first divided into three slices with overlap, and then the correction value (ΔD) is interpolated using DBI in each of them independently. In the overlapped areas, the average of the values coming from the two slices is taken into account.

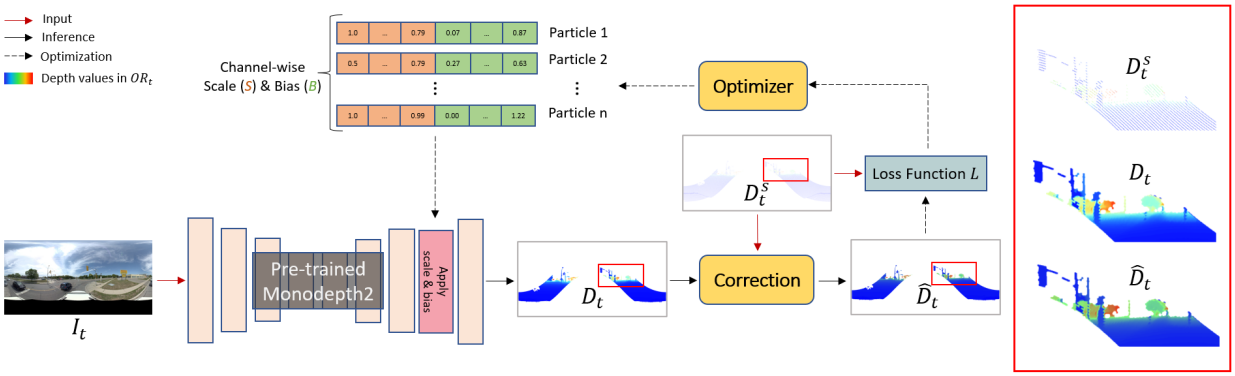


Figure 3.5: PanoDARS schematic architecture.

overlapped method, allows the interpolation to adapt to the local data density. In areas where LiDAR data is sparse, the independent approach can generate a coarser mesh that aligns more closely with the available data points, avoiding the introduction of potentially misleading information between widely spaced points. Conversely, in areas with more abundant data, it can create a denser mesh that captures the terrain's nuances in greater detail. This adaptability is key in creating an accurate digital representation of the landscape, especially in heterogeneous environments where data point density varies significantly.

Lastly, the nature of sparse data requires careful statistical or probabilistic treatment to infer the characteristics of the unobserved terrain. Methods that take into account the probability distribution of the terrain's features can make more informed guesses about the areas between sparse points, leading to a more accurate interpolation. With its tendency to average across boundaries, the overlapped method may not be as conducive to such statistical treatments. By blending data from different areas, the unique statistical properties of the sparse points can be overshadowed by the process of merging, which can homogenize the data and obscure the true nature of the underlying features that the sparse points represent.

In essence, the overlapped interpolation method is not well-suited for cases like the "blue rainbow" area with sparse LiDAR points because it prioritizes global consistency at the expense of local accuracy, does not adapt well to varying data densities, and may not fully leverage statistical methods needed to interpret sparse data effectively. The result is a less accurate and potentially misleading model of the terrain, which fails to capture the critical details that sparse data points can offer.

The correction stage begins with the calculation of a correction value for each valid pixel in the SD map. This value is obtained by comparing it with the corresponding predicted depth as DARS. Then, unlike DARS, DBI is utilized to create corrections for unknown points. Note that estimating depth for these points can lead to a densified map. To have this map the correction$\mathbf{D}_t$, yields the corrected depth map $\hat{\mathbf{D}}_t$.

In the optimization stage, PanoDARS refines the predicted depth by applying learnable auxiliary parameters to the intermediate features of the baseline. This involves splitting the

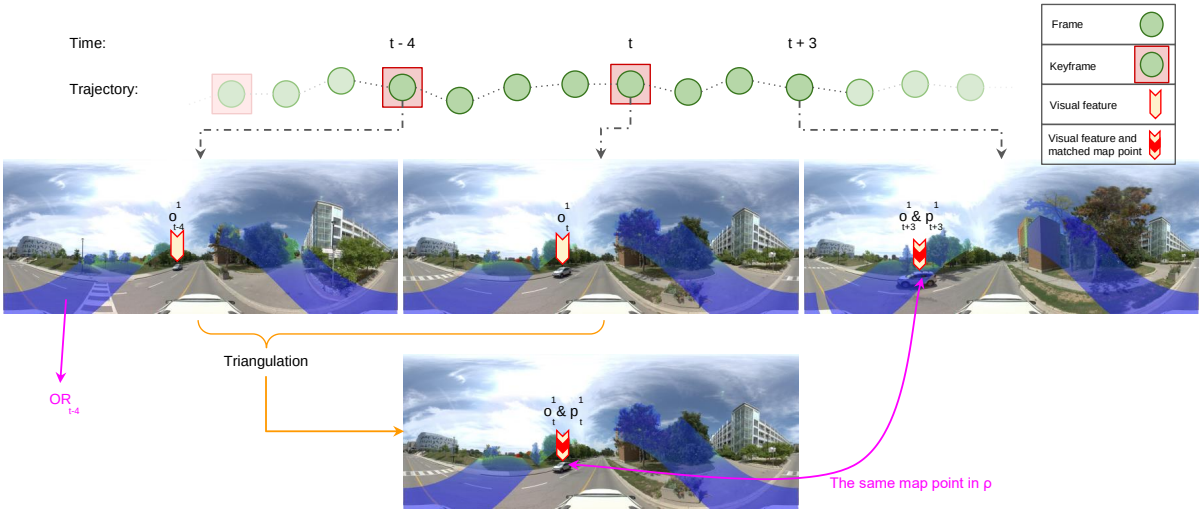


Figure 3.6: A map point p_t^1 is created at time t by triangulating visual features of times $t-4$ and t. This triangulation is because these two visual features are not in OR_{t-4} and OR_t. In F_{t+3}, o_{t+3}^1 points to the middle of the parked car's rooftop and matches with the previously created p_t^1 at time t which is now called p_{t+3}^1. At time $t+3$, o_{t+3}^1 is inside OR_{t+3}, which means the depth data is available. This enables the system to estimate the map points' 3D position more precisely through the depth coming from the depth estimation method [150].

pre-trained baseline into two components: a body G and a head H, with scales $\mathbf{S}$ and biases $\mathbf{B}$ applied to $G(\mathbf{I}_t)$. The optimization challenge is then formulated as minimizing the loss function L with respect to the auxiliary parameter adjustments $\Delta\mathbf{X}$:

$$L = \sqrt{\frac{1}{|^{l}\varphi_t|} \sum^{^{l}\varphi_t} \left\| \log(\hat{d}_t) - \log(d_t^s) \right\|} \tag{3.5}$$

This approach enables the effective application of PanoDARS for panoramic images, leveraging the strengths of the underlying baseline while addressing the unique challenges of panoramic depth estimation.

3.5 EXPERIMENTS

3.5.1 Datasets

KITTI

The KITTI dataset [151], a cornerstone in autonomous driving and computer vision research, was created with a meticulous approach to ensure accuracy and realism in its data. The dataset includes high-resolution RGB images and depth maps, characterized by a unique composition process and ground truth generation.

The depth maps in the KITTI dataset were primarily acquired using a Velodyne HDL-64E rotating 3D laser scanner, which has a field of view of 360° horizontally and 26.8° vertically, with a range of up to 120 meters. This scanner, capable of collecting approximately 1.3 million points per second with a 0.09-degree angular resolution and 2 cm distance accuracy, was instrumental in capturing detailed depth information of the scenes. The Velodyne scanner's data is stored as floating point binaries, with each point comprising its (x, y, z) coordinates and an additional reflectance value (r). The average size of each file/frame is about 1.9 MB, corresponding to roughly 120,000 3D points and reflectance values.

For ground truth generation, annotations for each dynamic object within the camera's field of view were provided. These annotations took the form of 3D bounding box tracklets

represented in Velodyne coordinates. The dataset includes various classes like cars, vans, trucks, pedestrians, cyclists, and more.

An important step in the creation of the KITTI dataset was the calibration and registration of the Velodyne laser scanner with respect to the reference camera coordinate system. This process began with the initialization of the rigid body transformation, followed by an optimization process. This optimization was based on the Euclidean distance of 50 manually selected correspondences and a robust measure on the disparity error with respect to the top-performing stereo methods in the KITTI stereo benchmark. The optimization employed Metropolis-Hastings sampling, ensuring high accuracy in aligning the Velodyne data with the camera imagery.

In conclusion, the creation of the KITTI dataset involved a comprehensive and detailed process, from the sophisticated data acquisition using high-end sensors to the careful calibration and annotation for ground truth generation. This rigorous approach has made the KITTI dataset a highly valuable resource for advancing technologies in autonomous driving and computer vision.

Figure 3.7 presents a sample from the KITTI dataset, illustrating the typical quality and characteristics of the images and depth maps. This visual representation aids in understanding the dataset's composition and the type of data researchers and algorithms are dealing with.

NYUv2

The NYU Depth V2 (NYUv2) dataset, introduced by Silberman et al. [152] in their seminal 2012 study, represents a significant advancement in the field of computer vision, specifically in indoor scene understanding. This dataset is pivotal for research focusing on depth perception, object recognition, and scene segmentation in indoor environments.

NYUv2 comprises approximately 120,000 pairs of RGB and depth images, each with a resolution of 640×480 pixels. These images were captured using a Microsoft Kinect sensor, which simultaneously records both color and depth information. The Kinect's depth sensor employs infrared technology to measure the distance of objects from the camera, thereby

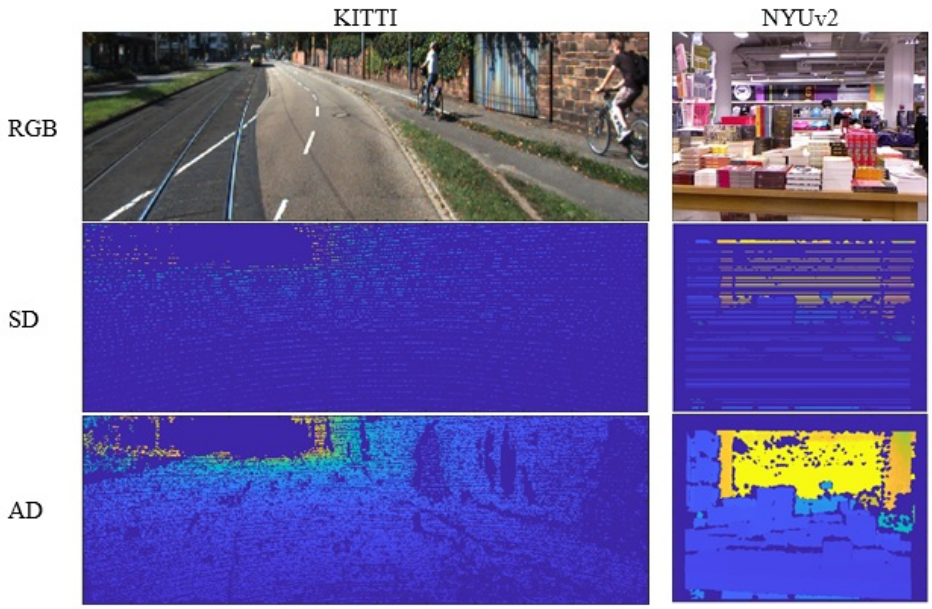


Figure 3.7: A sample of used datasets.

generating depth maps that correspond to the RGB images.

The dataset encompasses data collected from 464 different indoor scenes, including a variety of settings such as homes, offices, and public spaces. The diversity of these scenes is crucial, as it ensures the dataset covers a wide array of indoor environments. This variety makes NYUv2 an invaluable tool for developing and evaluating algorithms that require an understanding of different indoor contexts.

A unique aspect of the NYUv2 dataset is that the RGB and depth pairs were acquired as video sequences. This approach not only captures static scenes but also records dynamic changes within these environments over time. Such video sequences provide additional challenges for computer vision algorithms, particularly in understanding temporal consistency and changes in indoor scenes.

For standardized evaluation and algorithm development, the NYUv2 dataset is divided into an official train/test split. This split consists of 249 scenes for training and 215 scenes for testing. The training set offers a broad spectrum of data, enabling algorithms to learn from varied indoor settings and conditions. Conversely, the testing set, comprising scenes

distinct from those in the training set, serves as an unbiased benchmark for evaluating the algorithms' performance in new and unseen environments.

This structured organization of the NYUv2 dataset into training and testing sets is essential for the fair comparison of different algorithms. It allows researchers to understand the capabilities and limitations of various approaches to indoor scene analysis. Furthermore, insights gained from training and testing on this dataset have direct implications for real-world applications, including robotic navigation, augmented reality, and smart home systems.

In summary, the NYU Depth V2 dataset is a foundational resource in the field of computer vision. It provides a comprehensive platform for developing, testing, and benchmarking algorithms in indoor scene understanding, thereby driving progress and innovation in this research area. Sample data for the NYUv2 dataset, including the synthetic SD map, are illustrated in Figure 3.7.

YUTO MMS Dataset

The dataset encompasses four sequences with a combined length of over 18.9 kilometers. These sequences were recorded on August 12, 2020, and June 21, 2019, using a specialized vehicle equipped with an array of sensors, including a panoramic camera, a tilted LiDAR, a GPS, and an Inertial Measurement Unit (IMU). The data capture occurred at two distinct locations: the York University Keele Campus in Toronto and the Teledyne Optech headquarters in Vaughan (depicted in Fig 3.8). The dataset, which is currently in the submission process for publication, will be made publicly available upon the official release of its accompanying research paper. This dataset represents an extensive collaborative effort within AUSMLab published in [150].

The YUTO MMS Dataset offers unique challenges, particularly for visual SLAM systems, due to its varied landscapes that include both natural and urban settings. The data were collected under typical daily conditions, capturing busy roads bustling with cars, buses, pedestrians, and cyclists. The dataset also features multiple loops, numerous turns, traffic lights, and stop signs, which introduce complexities due to variable speeds and acceleration.

This dataset provides an excellent opportunity to test the effectiveness of visual SLAM algorithms in dynamic urban scenarios.

While recent SLAM datasets have contributed significantly to the field, none have explicitly addressed the specific challenges introduced by a 3D LiDAR sensor. Our dataset presents not only challenging scenarios for camera and LiDAR sensors but also focuses on a unique challenge associated with the use of LiDAR. Specifically, the dataset employs a tilted LiDAR configuration, where the sensor's spinning axis is inclined, unlike the typical upright position.

The tilted LiDAR configuration offers several advantages for SLAM applications. It allows for effective detection and capture of features at higher elevations, such as traffic signs, lights, poles, billboards, and tall buildings. These features, which are mostly static, provide sTable tracking points, crucial for accurate localization and mapping in SLAM algorithms. The lower heights, dominated by dynamic objects like vehicles and pedestrians, present challenges, but the presence of static high features significantly enhances the SLAM system's robustness and accuracy in urban environments.

The unique setup of the tilted LiDAR sensor introduces specific challenges for both LiDAR-centric and visual LiDAR SLAM methodologies. In LiDAR-centric SLAM, object tracking is difficult due to the limited time objects spend in the narrow LiDAR coverage area. The angled setup complicates point matching and geometric modeling for LiDAR SLAM, leading to challenges in feature tracking and sparse coverage at lower heights. Visual-LiDAR SLAM systems also face difficulties due to the limited and complex overlapping region between the camera and the tilted LiDAR sensor, and often, the topmost LiDAR beams fail to detect objects due to their absence, resulting in sparse data in the upper part of the LiDAR range.

To address these challenges, Teledyne Optech employed the Maverick MMS. This system is designed for mapping in environments with tall structures and combines the Ladybug 5 panoramic camera with the Velodyne HDL-32E LiDAR. The LiDAR's axis is set at a 45-degree angle relative to the camera's optical axis, enabling comprehensive capture of both visual and LiDAR data for a complete environmental understanding. Ground truth accuracy is achieved through RTK positioning using GPS signals.

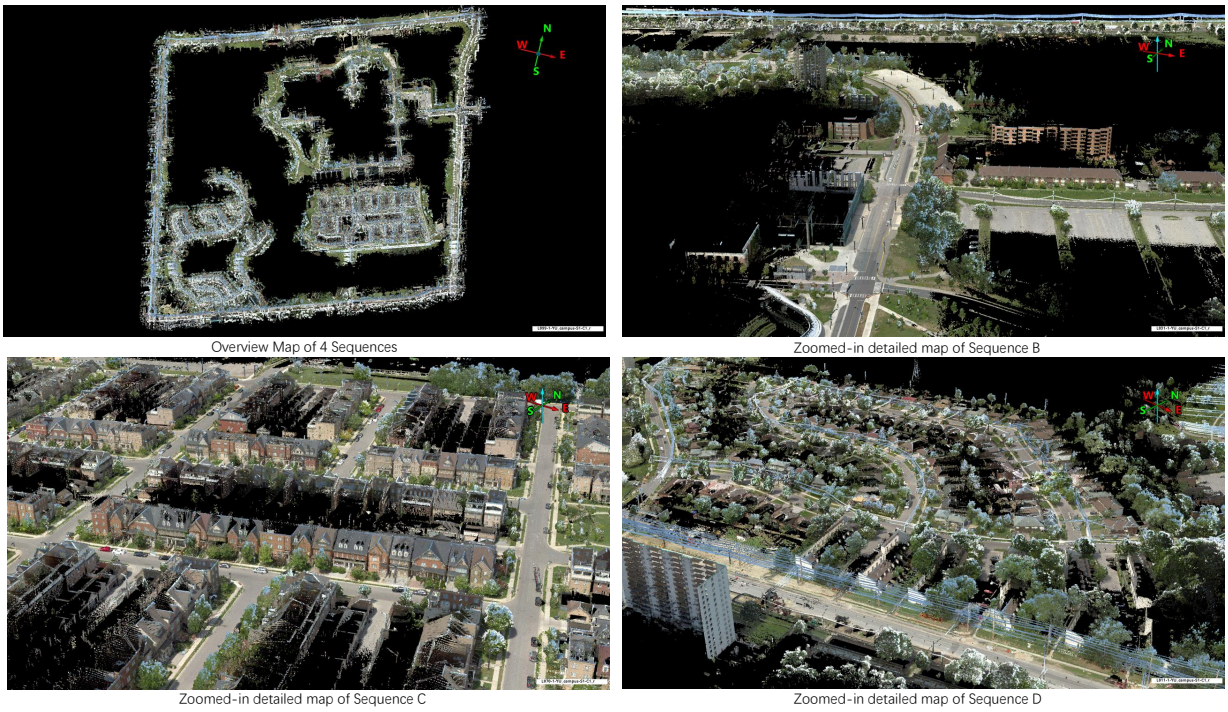


Figure 3.8: Color textured PC of YUTO MMS dataset in York University campus [150].

3.5.2 Discussion and Results

This section is divided into two parts. The first part discusses the effectiveness of the proposed methods on two well-known benchmarks, KITTI and NYUv2. It includes ablation studies that examine different aspects of the proposed method and their effectiveness. This is followed by a comparison with state-of-the-art methods. The availability of ground truth for data sets makes this analysis possible. This part is titled DARS Performance. On the other hand, the second section examines the efficiency of the proposed method through its application, which is visual SLAM. In this section, we discuss how the depth generated by the proposed method, PanoDARS, is used in a SLAM application. We compare its performance with traditional methods to demonstrate its effectiveness.

DARS Performance

Ablation Studies: Given that our dataset comprises two main parts, correction, and optimization, two methods need to be introduced to show the efficiency of DARS. The first one offers the capability of the correction part, while the second one indicates the efficacy of the optimization part. The first method, which has been discussed in the section 2.5.1, is called DBC. To provide the second method, we just changed the optimizer of DARS from PSO to L-BFGS, which has been used in the original version of f-BRS. In this method, named DARS (L), the correction module has been involved because, as discussed earlier, f-BRS cannot play a significant role without that.

Apart from those methods, two other methods, baseline, and DBI, have been presented (see Table 3.1). The former is defined to show the accuracy of the unsupervised version of monodepth2 [132] as our baseline. Unlike its well-known version, brought in Table 3.3, this version does not use a post-process median scaling coming from AD maps, and as a result, it suffers from a lack of accuracy due to scale ambiguity. The latter, which is DBI on the whole depth, is introduced to show the efficacy of distance-based segmentation in our DBC. For the sake of brevity, these two methods have just been discussed for KITTI.

Table 3.1: Ablation study for KITTI.

Method	Lower is better				Higher is better		
	AbsRel	SqRel	RMSE	$\text{RMSE}_{\log}$	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
baseline	0.996	15.398	19.324	5.715	0.000	0.000	0.000
DBI	0.864	18.253	16.888	3.149	0.183	0.330	0.447
DBC	0.046	0.185	1.676	0.091	0.976	0.991	0.995
DARS-L	0.046	0.185	1.676	0.091	0.976	0.991	0.995
DARS	**0.024**	**0.137**	**1.440**	**0.071**	**0.985**	**0.993**	**0.996**

As can be seen from Table 3.1, our proposed correction (DBC) has led to prominent

results compared to DBI. This implies that the distance-based approach has been entirely successful in error correction. Two reasons can be brought why this approach works properly. The first reason comes from the fact that depth is a spatial product. In spatial products like depth, adjacent pixels show similar errors, while distant ones might have completely different behavior. The second reason is related to the error pattern of points collected by the laser scanner. It means that more distance to the laser scanner leads to less accuracy. In better words, the error pattern in the common laser scanners is correlated with the distance, and points at the end of the scene have much more error than those near the scanner.

According to Table 3.1 and Table 3.2, DBC and DARS-L have led to an identical performance. This means that L-BFGS cannot improve the results because, unlike PSO, it does not have the capability for global search. In better words, it has been trapped in the local optimum provided by DBC. Therefore, for that identical performance and for the sake of conciseness, just the results of the DBC have been brought in Figure 3.9 and Figure 3.10.

As is clear from error patterns in Figure 3.9, related to KITTI and Figure 3.10 pertaining to NYUv2, the introduction of PSO has led to considerable improvements. Similarly, the quantitative evaluation, Table 3.1 and Table 3.2, demonstrates a remarkably better performance by PSO compared with L-BFGS.

Table 3.2: Ablation study for NYUv2.

Method	Lower is better			Higher is better		
	AbsRel	RMSE	$\text{RMSE}_{\log}$	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
DBC	0.018	0.766	0.747	0.972	0.974	0.975
DARS-L	0.018	0.766	0.747	0.972	0.974	0.975
DARS	**0.017**	**0.109**	**0.044**	**0.993**	**0.996**	**0.999**

Comparison with SOTA: As is clear from Table 3.3, DARS outperforms competing methods in terms of almost all assessment criteria except for $\delta_{1.25^2}$ and $\delta_{1.25^3}$. From the perspective of these two criteria, the performance of our method is not as good as that of

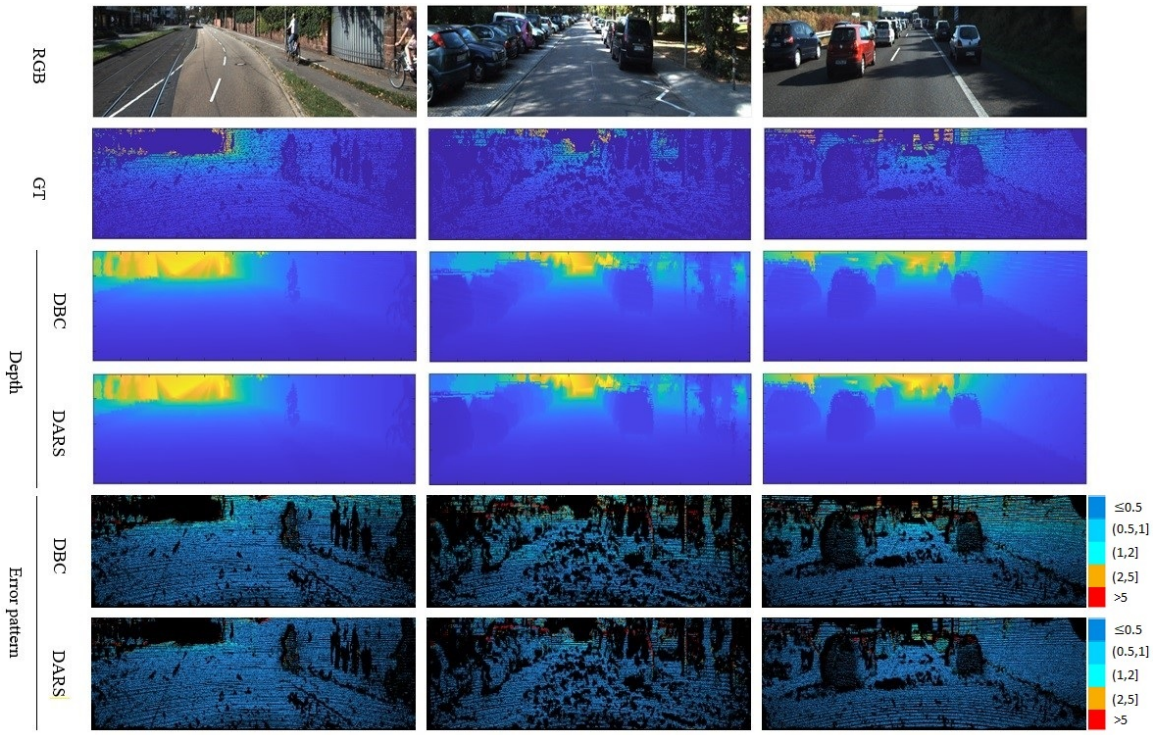


Figure 3.9: Visual results related to ablation study of KITTI dataset. Numbers on the right side of error patterns are in meters.

our second-place rival. This slight deviation cannot be considered as a drawback because DARS has led to better performance in terms of $\delta_{1.25}$, which is the primary criterion for accuracy assessment. Regarding the second dataset, NYUv2, DARS has outperformed the competing methods in terms of all criteria according to Table 3.4. This Table indicates how the proposed method has successfully adapted to an unseen in-domain dataset. Note that, unlike DARS, the other methods in Table 3.4 have been trained on NYUv2.

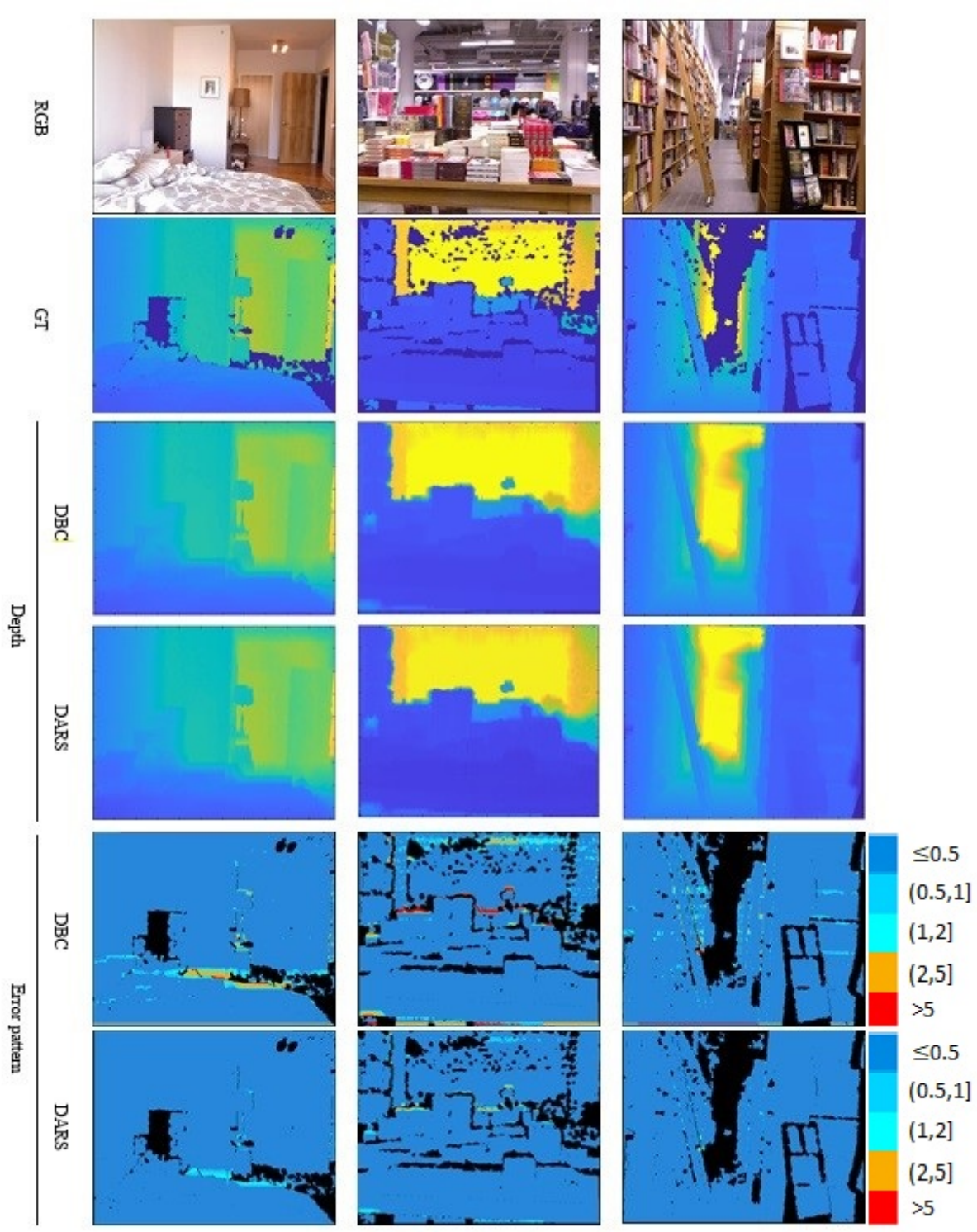


Figure 3.10: Visual results related to ablation study of NYUv2 dataset. Numbers on the right side of error patterns are in meters.

Table 3.3: Comparative study for KITTI.

Method	Lower is better				Higher is better		
	AbsRel	SqRel	RMSE	$RMSE_{log}$	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
[153]	0.120	0.789	4.755	0.177	0.856	0.961	0.987
[135]	0.132	0.994	5.240	0.193	0.833	0.953	0.985
[99]	0.114	0.898	4.935	0.206	0.861	0.949	0.976
[132]	0.090	0.545	3.942	0.137	0.914	0.983	0.995
[154]	0.090	0.424	3.419	0.133	0.916	0.984	0.996
[155]	0.060	0.231	2.642	0.094	0.958	0.994	0.999
[123]	0.058	0.190	2.360	0.088	0.964	**0.995**	**0.999**
DARS	**0.024**	**0.137**	**1.442**	**0.071**	**0.985**	0.993	0.996

Table 3.4: Comparative study for NYUv2.

Method	Lower is better			Higher is better		
	AbsRel	RMSE	$RMSE_{log}$	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$
[125]	0.158	0.641	-	0.769	0.950	0.988
[127]	0.110	0.392	0.047	0.885	0.978	0.994
[155]	0.107	0.373	0.046	0.893	0.985	0.997
[123]	0.103	0.364	**0.044**	0.903	0.984	0.997
DARS	**0.017**	**0.109**	**0.044**	**0.993**	**0.996**	**0.999**

PanoDARS Performance

In this section, as mentioned before, we discuss the effectiveness of the proposed depth estimator through its application. Specifically, this estimator is incorporated into the HDPV-slam system [150], and tested on the yuto (discussed in section 3.5.1) dataset. Our initial experiments concentrate on one of the two components of the system: the Depth Estimation module. This focus is due to its relevance to this research. Following this, we examine the accuracy of the system for SLAM applications.

Depth Estimation: The evaluation of the Depth Estimation module poses certain challenges due to the absence of ground truth information for assessing its correctness. To provide some quantitative measurements, we calculated the RMSE, explained in section 2.6.1, of the depth estimates obtained from the module, considering points where LiDAR data is available (see Figure 3.11).

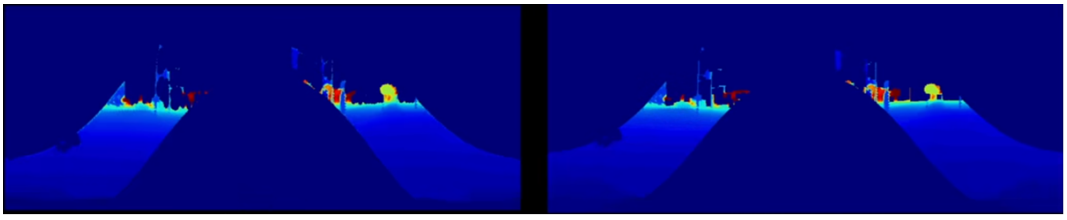


Figure 3.11: Comparison of depth maps generated by Bi-interpolation (left) and PanoDARS (right). The Bi-interpolation method shows deficiencies in correctly perceiving and representing the depth of objects such as trees and traffic lights. In contrast, the PanoDARS method provides a more accurate and perceptually correct depth representation. The improved accuracy of PanoDARS is evident in the clear and well-formed depth depiction of various objects, highlighting its effectiveness over Bi-interpolation in generating reliable depth maps (reproduced from [150]).

Figure 3.11 offers a qualitative comparison of a random scene between the Depth Estimation module (PanoDARS) and the bi-interpolation method employed in the base work, RPV-SLAM [156]. In comparison between the depth maps generated by the PanoDARS module and the previous bi-interpolation method, a notable difference is evident, particularly

when examining a magnified portion of the two depth maps. Notably, the magnified section of our depth map, shown in Figure 3.11, exhibits sharper details and a better alignment with the corresponding real RGB image, showcasing the enhanced accuracy and visual quality achieved by our method. On the other hand, the depth map generated by the bi-interpolation method appears blurry and lacks the sharpness observed in our results.

Moreover, the bi-interpolation method introduces noticeable noise in the depth map, as highlighted by the presence of red circles in the comparison (Figure 3.11). In contrast, our proposed module exhibits a higher level of noise reduction and produces more reliable and smoother depth estimates. The absence of noise artifacts in our results further supports the superiority of PanoDARS in generating clean and accurate depth maps.

Table 3.5: RMSE (m) of the two depth estimation methods on LiDAR points

Sequence	Bi-Interpolation	PanoDARS
A	1.01	0.01
B	0.97	0.00
C	0.76	0.01
D	1.20	0.01

Depth Assessment via SLAM Trajectory Estimation: HDPV-SLAM, which operates using PanoDARS (refer to Appendix A.5), was compared with RPV-SLAM, a system based on Bi-Interpolation (refer Appendix A.3.2). Another reason for selecting RPV-SLAM as a benchmark is that it is the only method employing the same sensors as our approach. The assessment techniques elaborated upon in section 2.6.4 were applied to all the comparative analyses. Table 5.2 presents the total number of frames and keyframes in each sequence of the dataset, which are identified as A, B, C, and D.

We will conduct an in-depth comparison between two different SLAM systems to discuss the efficacy of PanoDARS. HDPV-SLAM utilizes PanoDARS for depth estimation, while RPV-SLAM utilizes Bi-Interpolation. We will evaluate the two techniques across three crucial metrics: ATE, RTE, and RRE. Analyzing the ATE from Figure 3.12, it is evident that

Table 3.6: Keyframe conversion rate and frame statistics.

	Number of frames	Average number of keyframes	Percentage
Sequence A	717	192.2	26.80%
Sequence B	8382	2306.4	27.51%
Sequence C	10778	2576.2	23.90%
Sequence D	4500	1117.8	24.84%

HDPV-SLAM consistently outperforms RPV-SLAM across all sequences, demonstrating its superior capability in accurate localization. Particularly in Sequence B, the margin by which HDPV-SLAM surpasses RPV-SLAM is notable, emphasizing the robustness of PanoDARS in achieving precise positioning.

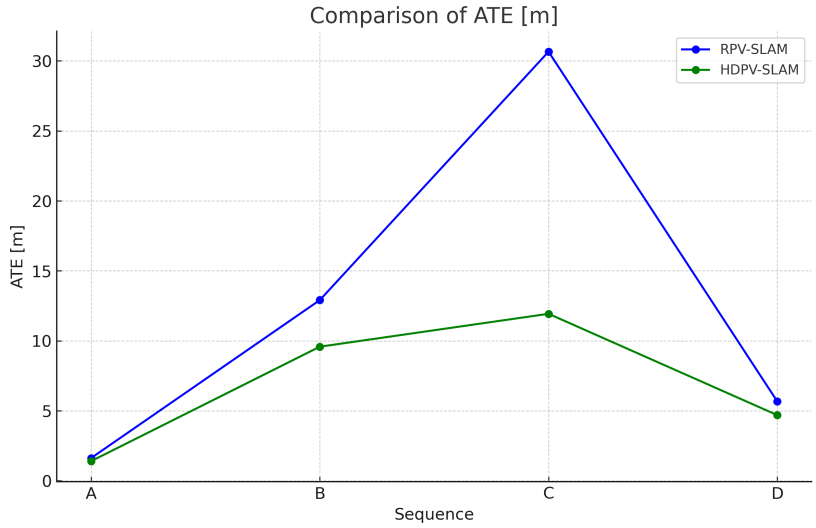


Figure 3.12: Comparative analysis of ATE for HDPV-SLAM and RPV-SLAM across distinct sequences [150].

Turning our attention to the RTE, depicted in Figure 3.13, HDPV-SLAM again exhibits superior performance in all sequences. This indicates a more reliable relative positioning, with Sequence D showing a remarkable difference in favor of HDPV-SLAM. Such consistent

outperformance in RTE accentuates the efficacy of PanoDARS in maintaining accurate relative translation estimates over the Bi-Interpolation method.

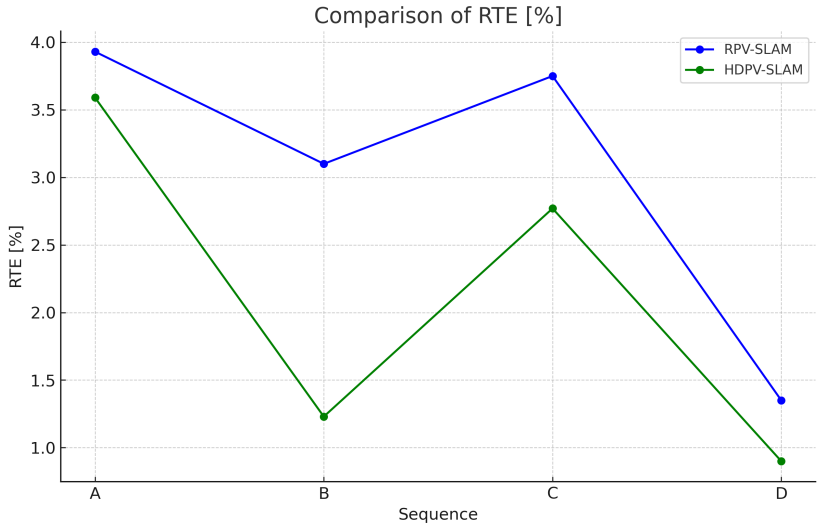


Figure 3.13: RTE percentage comparison between HDPV-SLAM and RPV-SLAM across distinct sequences [150].

Lastly, the comparison of RRE in Figure 3.14 reinforces the superior orientation accuracy of HDPV-SLAM. The method shows enhanced performance, particularly in Sequence C, which underscores the robustness of PanoDARS in ensuring precise rotational estimates. This suggests that HDPV-SLAM's orientation estimation capabilities are more reliable and stable than RPV-SLAM's.

In conclusion, the comprehensive comparison across ATE, RTE, and RRE underscores the superiority of HDPV-SLAM over RPV-SLAM. The PanoDARS approach, employed by HDPV-SLAM, evidently ensures more accurate, consistent, and reliable localization and orientation estimations than the Bi-Interpolation method utilized in RPV-SLAM. Hence, in the context of these metrics, HDPV-SLAM has been deemed the better-performing system.

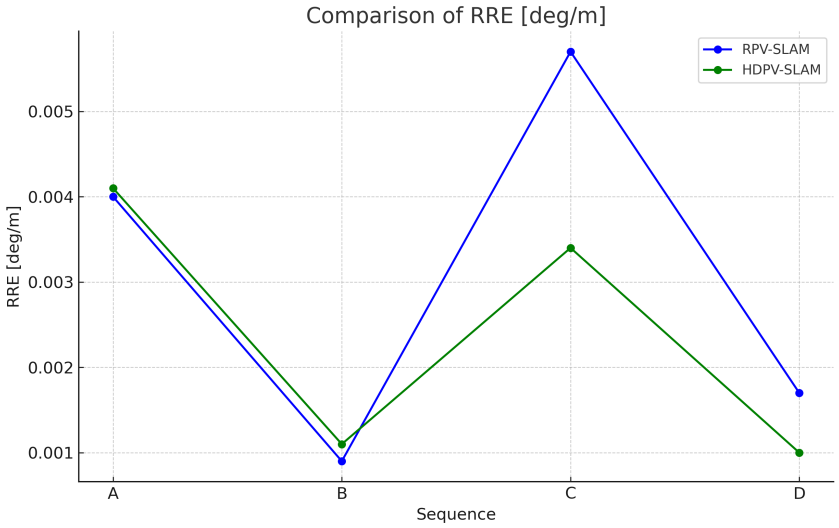


Figure 3.14: Evaluation of RRE for HDPV-SLAM and RPV-SLAM across distinct sequences [150].

3.6 Summary

In this chapter, we have methodically examined the challenge of depth estimation in dynamic environments and its computational demands within deep learning frameworks. Our proposed solution, the Depth Adaptive Refinement Scheme (DARS), offers an elegant, in-domain adaptive approach that enhances depth estimation during inference, eliminating the need for retraining on new datasets.

DARS, synergizing with a pre-trained monodepth2 network, introduces adaptive refinements via channel-wise scaling and biasing—hallmarks of a functionally adaptive neural network. It encompasses a dual-module architecture: the correction module leverages novel distance-based interpolation for broad improvements, while the optimization module refines these estimations to a higher degree of accuracy using a particle swarm optimizer to calibrate the feature backpropagating refinement scheme (f-BRS).

Our empirical investigations, conducted on KITTI and NYUv2 datasets featuring perspective geometry, demonstrate that the correction module alone provides a commendable

baseline. However, when supplemented with the optimized f-BRS, the system's performance excels, setting new benchmarks in accuracy and outperforming rival models.

Furthermore, the versatility of DARS is showcased through PanoDARS, an extension tailored for the third dataset characterized by panoramic geometry. PanoDARS reaffirms the adaptability of our method, yielding promising results in a diverse set of conditions. The application of PanoDARS to panoramic data highlights the scheme's robustness and its potential in a broader spectrum of scenarios, beyond the conventional datasets.

Conclusively, the chapter underscores the efficacy of DARS and PanoDARS as advanced, scalable solutions for depth estimation. These methods have proven their merit not only on standard benchmarks but also in less conventional, panoramic environments, paving the way for future applications in the ever-evolving landscape of visual perception.

Chapter 4

Geospatially informed Single Task Autoencoder for DTM Reconstruction

4.1 Introduction

DTMs are a crucial component of geospatial analysis, as they provide foundational information about the earth's topography without the interference of artificial obstructions. The applications of DTMs are broad and varied, ranging from flood risk assessment [157] to urban planning [158] and environmental conservation [159]. As such, they represent an essential tool for disaster management and sustainable development initiatives.

DTMs have traditionally been created using geodetic measurements, which are transformed into a topographic map featuring contour lines. These contour lines are then digitized and gridded to convert the hardcopy map into a digital format called a DTM. However, modern data capture technologies such as LIDAR and remote sensing have significantly improved the efficiency and precision of DTM generation [160].

With remote sensing, DTMs can be created from aerial or satellite imagery using the stereoscopic imaging concept. This involves capturing two or more images of the same area from slightly different angles, similar to how human eyes perceive depth. Once these images are analyzed and intersected, they construct three-dimensional representations of the earth's

surface [161], known as a DSM, which can be converted to DTM. On the other hand, a LiDAR point cloud, which contains x, y, and z positional data, is composed of ground points, which reflect the terrain, and non-ground points, which represent structures such as buildings and trees. Combining these points forms a DSM [162], which can subsequently be converted into a DTM.

The process of extracting DTMs from DSMs typically involves one of four primary groups of methods: slope-based, morphology-based, interpolation-based, and segmentation-based [163]. Slope-based methods, which involve filtering based on a predefined slope threshold, are effective in terrains with significant slope variations but are not well-suited for rugged or discontinuous terrain [164]. In contrast, morphology-based methods rely on morphological operators to smooth out irregularities and emphasize significant elements, which aids in isolating and removing objects like buildings and vegetation that are not part of the natural terrain. The accuracy of these methods depends significantly on the size of the structural element [165]. Interpolation-based methods, including Triangulated Irregular Networks (TINs) [166] and grid-based techniques like Kriging and Inverse Distance Weighting (IDW), as well as adaptive methods, play a pivotal role in estimating the height of unmeasured areas based on the elevation of nearby measured locations. However, these methods are limited to interpolation and do not support extrapolation. This means that unknown points that are not surrounded by known points remain without estimable information. Lastly, segmentation-based filters, such as Cloth Simulation Filtering (CSF), are essential for classifying point clouds into ground and non-ground points. However, this approach often requires additional methodologies, particularly interpolation-based ones, for refinement and gap-filling. The effectiveness of each method is dependent on the specific terrain characteristics and the intended use of the model [167, 168]. Therefore, it is essential to choose the appropriate method based on the specific requirements of the project.

The strategies mentioned earlier do not suffice in addressing intricate terrains comprising mountainous regions, dense low-height foliage, and buildings encompassed by trees. These challenging environments demand innovative techniques that effectively navigate complex

topography and vegetation. Consequently, there has been a shift towards adopting deep learning technologies, particularly those based on CNNs, which have proven to be highly effective in addressing these challenges [169]. DL-based techniques can be broadly classified into point-based [170], voxel-based [171], and depth image-based [164] approaches, each with its unique advantages and challenges [172].

Point-based methods, which directly manipulate the original geometric configurations of point clouds, offer fidelity to the structural complexities of the data. However, these methods face limitations due to their reliance on the extent of sampling, which can restrict their effectiveness in diverse or expansive environments [173]. The aforementioned task highlights a fundamental challenge in handling large datasets and optimizing computational efficiency, a prevalent theme in discussions surrounding point cloud processing methodologies.

To overcome some of these limitations, voxel-based methods use a 3D grid representation that allows the application of 2D CNNs within 3D spaces. This approach, while mitigating some of the computational burdens associated with point-based methods, still needs significant memory and processing costs. Furthermore, the task of determining the most effective combination of inputs for these methods introduces an additional layer of complexity to their implementation.

An alternative approach is a depth image-based deep learning, which shows promise in addressing scalability and efficiency challenges inherent in point and voxel-based methods. This method projects point clouds onto a 2D plane, transforming 3D data into a format that 2D CNN architectures can handle. However, this conversion process is not without its drawbacks, as the simplification from 3D to 2D can lead to a loss of critical geometric information, particularly in settings characterized by complex structures, such as forested areas [172]. However, two key features of image-based methods have captured the interest of researchers, adding to the already complex landscape of point cloud processing. Firstly, the inherently organized nature of input data in image-based methods aligns seamlessly with the requirements of convolutional operators, which stand in contrast to the unstructured format of Point Clouds (PCs). It is important to note that methods relying on PC often

encounter similar issues to those faced by image-based approaches. This is because they typically produce a raster output, and most of their processes, except for the initial stage, are similar to voxel or depth-image-based approaches [170]. Additionally, these methods are more computationally demanding. According to the literature, PC-based methods use either a PointNet network [170] or rasterization [174] to convert 3D point clouds into raster format. After this step, the process becomes similar to image-based methods. For instance, in [174], a two-stage method has been proposed, where the first stage involves rasterizing point clouds before feeding them into a generative network like image-based ones. The above-mentioned challenges related to PC-based methods led us to focus on image-based methods in our research.

This chapter addresses the challenges associated with DTM reconstruction models that rely on DSMs as their input, as discussed in [175]. We introduce a geospatially-informed autoencoder designed to enhance the reconstruction of DTMs indirectly. This method is termed "geospatially informed" because it integrates nDSM, a form of geospatial information, as a constraint within the network. By incorporating nDSM, the autoencoder leverages topographic data to normalize background impressions and increase resistance to topographical variations and slopes. This approach ensures that the network's architecture inherently supports and optimizes the spatial characteristics of the input data, thereby improving the robustness and accuracy of terrain reconstruction. These challenges arise from the need for data processing, ranging from post-processing [169] to sensitivity towards the distribution of training and test datasets [174]. To address these challenges, physical constraints are incorporated into our autoencoder. The proposed autoencoder is specifically designed for DTM generation and introduces an innovative DL framework that generates DTMs through nDSM. The network itself updates the nDSM, which is then used to generate the DTM via a physical relationship that expresses the DTM as a subtraction of the network's input, DSM, from the nDSM. This approach reduces the sensitivity of the framework to the earth's topography, which is often a major obstacle to making these methods more generalizable. To make it more physically equipped, the traditional skip connections, as concatenative ones,

are replaced with subtractive ones in our network architecture. This modification enables the network to be better equipped to handle physical variations and, thus, improve the overall performance of the framework.

4.2 Related work

Deep learning techniques have improved the accuracy of DTM extraction. This review summarizes studies that integrate topographic knowledge into deep learning and develop new neural network architectures for terrain modeling. In the most relevant task, [169] presented a method for generating DTMs directly from DSMs without the need for traditional filtering methods to remove non-ground pixels. Their approach employs a Hybrid Deep Convolutional Neural Network (HDCNN) that merges the U-net architecture with residual networks, enhanced by a multi-scale fusion strategy for DTM generation. This method excels in complex scenes, outperforming both deep learning-based filters and reference algorithms, particularly in challenging environments.

In a related study, [175] explored the efficacy of the pix2pix model, a conditional GAN framework, for converting DSMs to DTMs in built-up areas without additional data or extensive parameter tuning. This approach simplifies the DTM generation process and demonstrates the model's versatility across varied topographies and building characteristics, highlighting its potential for flood risk assessment at the building level. Further extending their research, [175] applied the pix2pix model to coastal urban areas of Japan, achieving a high spatial resolution DTM with minimal RMSE, underscoring the model's capability in handling high and densely built environments. [174] have recently tackled the challenge of DTMs directly from ALS point clouds. To achieve this, they put together a large dataset for training purposes and introduced a new method called DeepTerRa, which is based on deep learning and rasterization. The authors' work is significant because it not only provides a valuable dataset, but also proposes a unified framework for DTM extraction. The proposed method showcases submetric error levels, indicating its potential for end-to-end solutions.

In this category, lately, [176] developed an end-to-end deep learning approach for DTM generation from DSMs, utilizing an EfficientNet-based architecture within a UNet framework. This method, despite its relatively lower parameter count, efficiently discriminates non-ground pixels and retains detailed landscape features, offering a significant step forward in preserving the anthropogenic geomorphology of landscapes.

DTMs are generated by removing non-ground areas from Digital DSMs and subsequently filling the resultant gaps. This process of gap-filling has been a focal point of numerous research studies due to its importance in creating accurate representations of the Earth's terrain. Among these, [177] investigated the use of a Wasserstein Generative Adversarial Network (WGAN) with a fully convolutional architecture and a contextual attention mechanism for filling voids in Digital Elevation Models (DEMs). Utilizing GeoTIFF data from various regions in Norway, provided by the Norwegian Mapping Authority, the study demonstrated the model's capability to generate semantically plausible data for DEM inpainting. This research showcases the adaptation of image inpainting methodologies to DEM void filling, offering a proof of concept for the application of deep generative models in enhancing remote sensing data.

Further advancing the field, [178] introduced a multi-attention GAN specifically designed for addressing voids in DEMs, which commonly occur due to instrumentation artifacts and ground occlusion. Their model integrates a multiscale feature fusion generation network for preliminary void filling, followed by a multi-attention network for the recovery of detailed terrain features, and employs a channel-spatial cropping attention mechanism to improve network performance. The discriminator's convolution layers are enhanced with spectral normalization, and the model's optimization incorporates a combined loss function that includes both reconstruction and adversarial losses.

Additionally, [179] presented an innovative approach for DTM data void filling by integrating topographic knowledge into a Conditional Generative Adversarial Network (CGAN), resulting in the Topographic Knowledge-Constrained Conditional Generative Adversarial Network (TKCGAN). This model significantly improves the accuracy of elevation and surface

slope in reconstructed DTMs by incorporating topographic constraints into the loss functions.

4.3 Methodology

4.3.1 Problem Formulation

This study introduces an autoencoder architecture that incorporates an encoder and a decoder designed to predict DTM from DSM, as illustrated in Figure 4.1. The encoder is tasked with converting the input DSM into a latent representation within a lower-dimensional space. The DSM is described as 2.5-dimensional geospatial data, where each pixel is characterized by its image coordinates (x, y) in the pixel grid, along with a positive scalar value z denoting elevation. This relationship can be articulated as a mapping from $(x, y) \in \mathbb{R}^2$ to $z \in \mathbb{R}_+$, where $\mathbb{R}_+$ symbolizes the set of positive real numbers. The encoder generates a latent representation, denoted by $\mathbf{Z}$, existing in a 3D space represented as $\mathbb{R}^{d \times p \times q}$. Here, d indicates the depth of the feature space, while p and q represent the reduced spatial dimensions. During training, the encoder's parameters or weights, θ_{enc} (Equation 5.2), are optimized to efficiently capture the significant features of the DSM in the latent space.

$$\mathbf{Z} = f_{\text{enc}}(\text{DSM}; \theta_{\text{enc}}) \tag{4.1}$$

Regarding the decoder, it is tasked with reconstructing the estimated DTM values, denoted by D_{est}. This process involves optimizing the regression decoder parameters, θ_{reg}, as detailed in Equation 5.3:

$$D_{\text{est}} = f_{\text{reg}}(\mathbf{Z}; \theta_{\text{reg}}) \tag{4.2}$$

4.3.2 Regression Task

The regression autoencoder, as can be seen from Figure 4.2, is equipped with subtractive skip connections to reconstruct $\mathbf{D}_{\text{est}}$. In detail, Each layer l in the encoder performs a

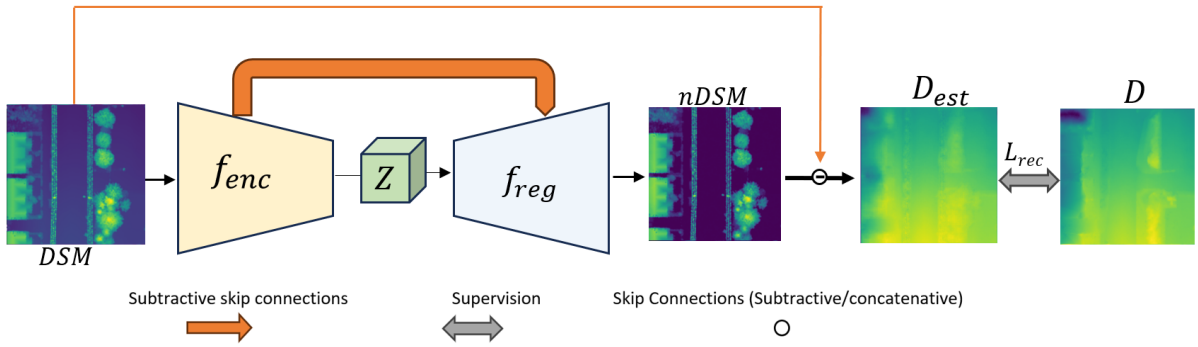


Figure 4.1: Architecture of the proposed method.

transformation represented by the equation 5.14.

$$h_i = f^i_{\text{enc}}(h_{i-1}), \tag{4.3}$$

where $h_0 = \text{DSM}$ and $h_{\text{end}} = Z$, as depicted in Fig 4.2, and f^i_{enc} is the transformation function at the ith layer. On the other hand, the decoder's regression branch transforms the latent representation Z into the nDSM. The transformation at each decoder layer of the regression branch i can be represented as:

$$\hat{h}_i = f^i_{\text{reg}}(\hat{h}_{i-1}), \tag{4.4}$$

where $\hat{h}_0 = Z$ is the latent representation, and $f^i_{\text{regression}}$ is the corresponding layer's transformation function.

In the regression task, we carry out layer-wise operations like subtraction and concatenation. To ensure clarity and precision, we require a formal structure for these operations. The output at the l-th layer of the encoder is denoted by h_l, and its corresponding decoder output is denoted by $\tilde{h}_l$. The operations are outlined as follows: the subtraction operation at layer l is defined by $S_{\text{sub},l} = h_l - \tilde{h}_l$, or the concatenation operation at layer l, denoted by $\oplus$, is articulated as $C_{\text{concat},l} = h_l \oplus \tilde{h}_l$.

This process acts as a filter that highlights the high-frequency elements of the DSM. It produces a high-frequency component represented by S^l_{sub} in the case of subtractive skip connections. This component can be considered as the difference between DSM and DTM and

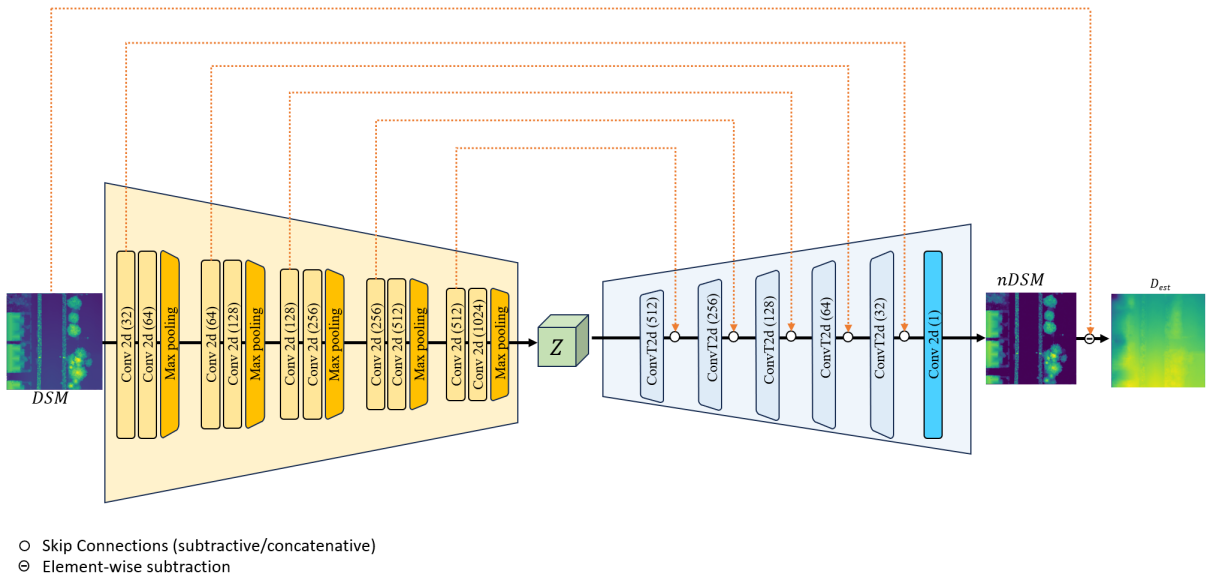


Figure 4.2: Detailed architecture of the regression task.

can be geospatially equivalent to nDSM. This comparison arises because the regression's loss affects the decoder outputs or $\tilde{h}_l$, which measures the deviation between the estimated DTM and the true one. As a result of this fact, the decoders have features that aim to capture the essential qualities of DTM. In contrast, concatenative skip connections strive to provide informed decoder features for their respective details, viewed as high frequency. Finally, in both cases of skip connections, the regressed DTM, D_{est}, is obtained by subtracting nDSM from DSM.

The loss function of this task is designed to minimize the difference between the predicted elevation values and the actual DSM values, facilitating the model's ability to accurately capture the underlying terrain elevation by adjusting the model parameters. The objective of the regression is to estimate D_{est} by optimizing the parameters θ_{enc} and θ_{reg} through the minimization of the following loss function:

$$L_{rec}(\theta_{\text{enc}}, \theta_{\text{reg}}) = \sum_{i=1}^{N} \ell(\hat{D}_i, D_i), \tag{4.5}$$

where ℓ represents a loss metric (here, mean absolute error), $\hat{D}_i$ are the estimated elevation values from the regression model, D_i are the actual target DTM values, and N signifies the

number of samples.

4.4 Experiments

4.4.1 Characteristics of Datasets

The datasets chosen for evaluating the proposed network's performance can be categorized into three groups, each of which has been widely adopted as a benchmark for DL-based DTM reconstruction study. Our study aims to comprehensively analyze these datasets. The insight gained through this analysis can not only help us evaluate the performance of our proposed method better but also lay the foundation for future research.

USGS

The present study aims to deliver a comprehensive insight into the USGS datasets [180]. USGS dataset contains four main subsets with geographical and structural information, tabulated in Table 4.1. It is worth noting that SU itself has been divided into three parts: SUI, SUII, and SUIII (Figure 4.5). Similarly, the KA subset has been separated into two main parts: KAI and KAII (Figure 4.6). However, KW (Figure 4.8) and RT (Figure 4.7) are not divided into smaller parts.

Table 4.1: Structural characteristics of USGS dataset, where each subset comprises three semantic classes: non-ground, ground, and noise.

Dataset	**Year**	**Coverage [km^2]**	**Semantics**	**Density (pts/m^2)**	**Topography Type**
SU	2014	45	3	11.93	City, industrial, residential
KA	2005	114	3	2.42	Dense forests, steep mountainous terrain
KW	2020	50	3	7.89	Watershed, landscape response to debris flows
RT	2018	45	3	48.46	Mission River area, post-Hurricane landscape

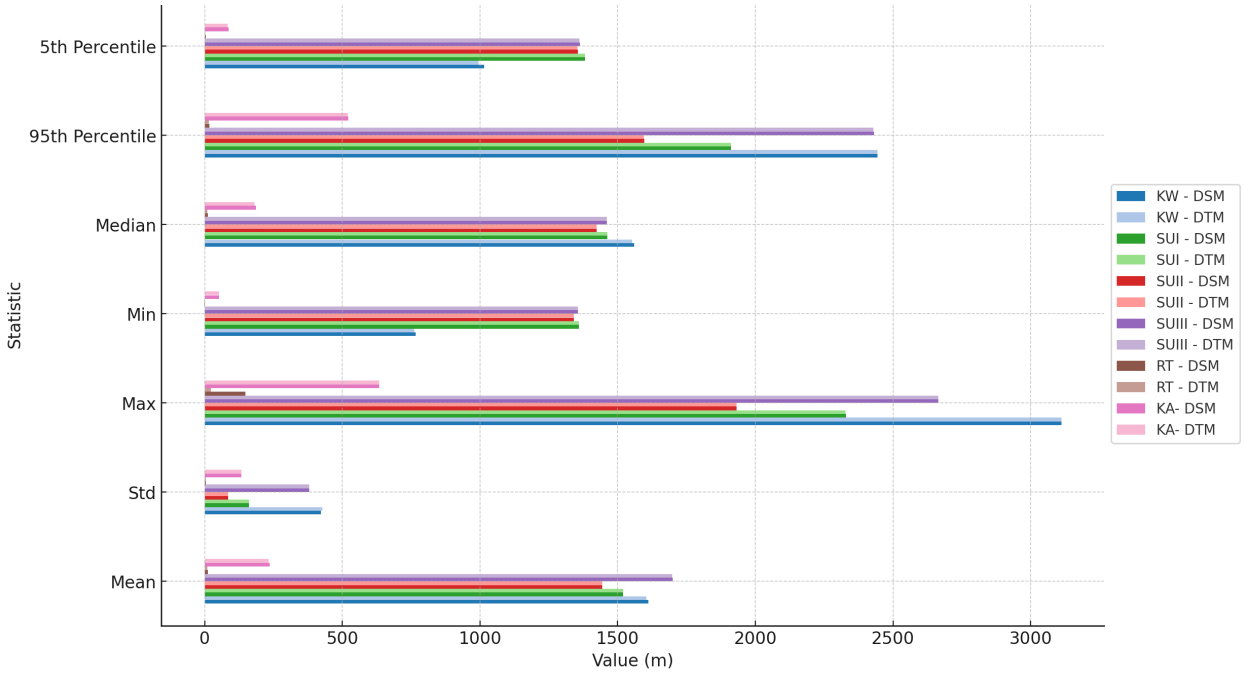


Figure 4.3: illustration of statistical features in the USGS dataset.

Table 4.2: Quantitative analysis of statistical features in the USGS dataset (unit: meter).

Dataset	Mean	Std	Max	Min	Median	95th Percentile	5th Percentile
KW - DSM	1611.45	421.81	3111.84	767.34	1559.19	2443.53	1015.90
KW - DTM	1603.88	428.09	3111.84	760.25	1553.54	2443.53	994.72
SUI - DSM	1519.59	160.94	2329.18	1359.07	1463.09	1912.02	1382.29
SUI - DTM	1518.69	160.97	2328.03	1359.07	1461.85	1911.14	1381.52
SUII - DSM	1445.03	86.06	1932.76	1340.83	1424.46	1596.04	1355.61
SUII - DTM	1443.89	86.26	1931.05	1340.82	1423.11	1595.36	1354.51
SUIII - DSM	1699.62	380.30	2664.46	1356.26	1460.83	2430.71	1362.73
SUIII - DTM	1699.03	380.35	2664.46	1356.26	1459.84	2429.83	1362.14
RT - DSM	11.12	4.02	147.50	-0.99	11.34	17.84	3.72
RT - DTM	9.73	3.78	23.52	-0.16	10.27	15.46	3.28
KA- DSM	236.53	132.58	634.49	52.50	187.49	520.96	87.25
KA- DTM	233.09	134.12	634.31	52.43	182.14	520.54	83.27

Figure 4.3, and Table 4.2 present the statistical characteristics of the USGS dataset. In the SUI region, the DSM and DTM datasets show minimal differences in mean and median values, with a mean of around 1519 m for both datasets. This indicates a consistent elevation profile across the region. The maximum values for DSM and DTM are nearly identical, suggesting that the highest elevation points are captured similarly in both models. The 5th and 95th percentile values are very close, indicating a less varied elevation range compared to the KW region.

For the SUII region, there's a slight decrease in mean elevation values compared to SUI, with the DSM and DTM means around 1445 m and 1443 m, respectively. The standard deviations are lower here, indicating a more uniform elevation profile. The maximum elevation is significantly lower than in the KW and SUI regions, which could suggest a less rugged terrain.

The SUIII region presents higher mean and median values than the previous regions, with means around 1699 m. The high standard deviation indicates a wider range of elevation differences, possibly due to more varied terrain. The maximum elevation is between those of KW and SUI, suggesting a mix of terrain features.

The KA region shows moderate mean elevation values (236.53 m for DSM and 233.09 m for DTM) compared to the other regions. The standard deviations are higher than in RT but lower than in the mountainous regions, indicating a moderate variation in elevation. To provide a better understanding of the position of the network's input (i.e., DSM) to the network's output (i.e., DTM), the fitted normal distribution for each subset/region has been illustrated in Figure 4.4 as well.

For the KW region, the DSM and DTM mean values are closely aligned at 1611.45 m and 1603.88 m, respectively, suggesting a relatively consistent elevation profile between the surface and terrain models. However, the standard deviation is slightly higher for the DTM, indicating more variation in terrain elevation compared to the surface. Notably, both models share the same maximum value of 3111.84 m, pointing to a significant elevation feature present in both datasets. The 95th percentile values are identical for both models, reinforcing the

presence of high-elevation features. The RT region stands out with significantly lower mean values (11.12 m for DSM and 9.73 m for DTM), indicating a very flat region. The negative minimum value in the DSM dataset might represent an artifact or, less likely, a below-sea-level feature. The maximum values are notably lower than in other regions, emphasizing the flatness of the area.

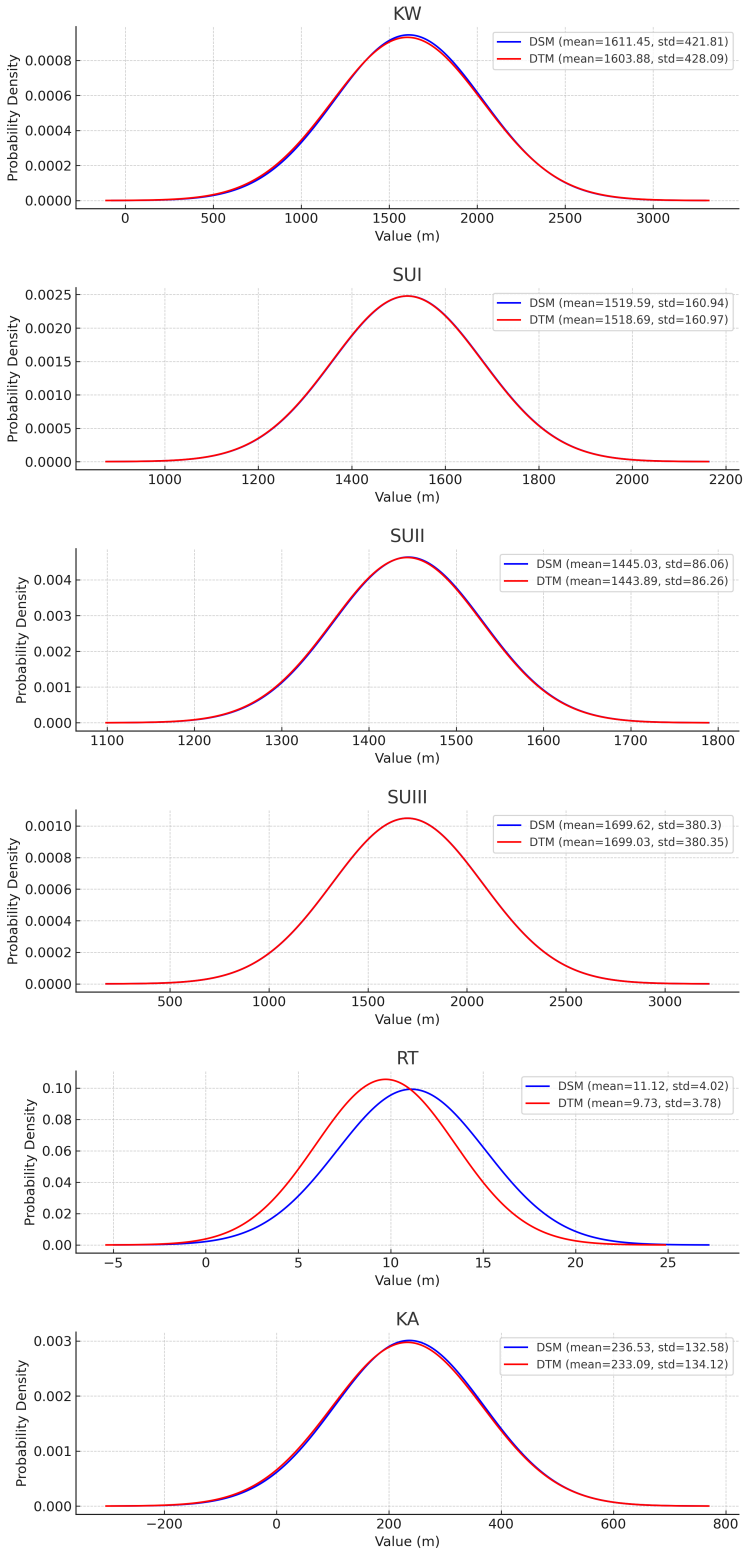
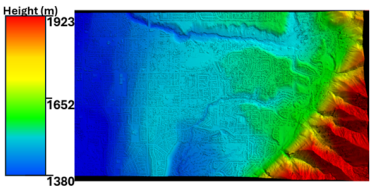
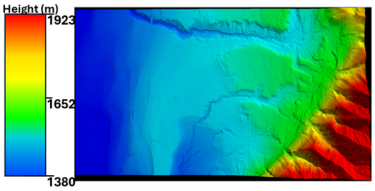


Figure 4.4: Normal distribution fits for USGS.

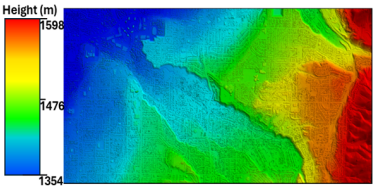
These subsets, in total, reveal a rich tapestry of elevation profiles across different regions, from flat terrains in RT to rugged landscapes in KW. The DSM and DTM models consistently capture the elevation characteristics of each region, with slight variations that might be attributed to the models' inherent differences in representing surface and terrain features.



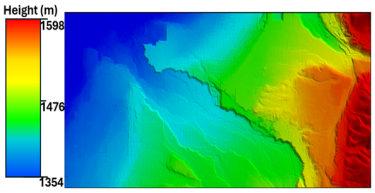
(a) DSM of SUI



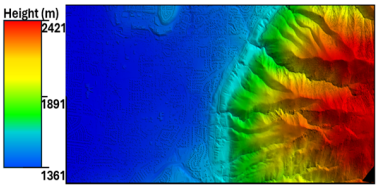
(b) DTM of SUI



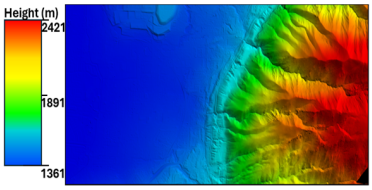
(c) DSM of SUII



(d) DTM of SUII

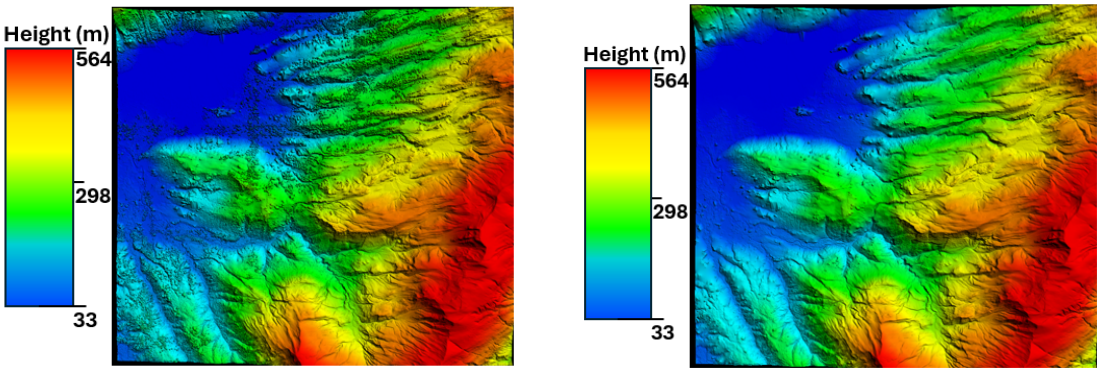


(e) DSM of SUIII



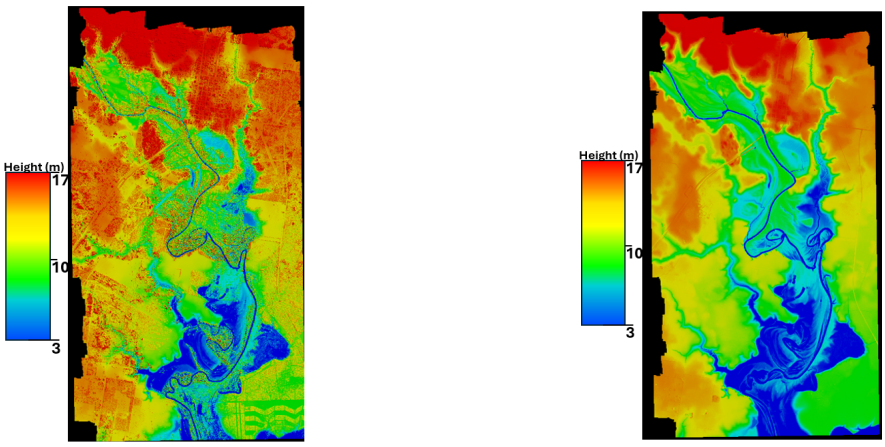
(f) DTM of SUIII

Figure 4.5: Topographic detail of SU region samples.



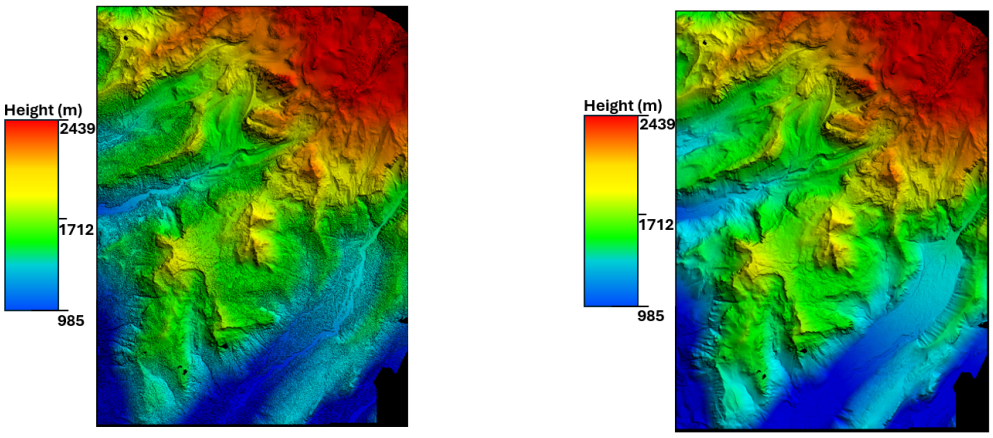
(a) DSM of KA (b) DTM of KA

Figure 4.6: Topographic detail of a KA region sample.



(a) DSM of RT (b) DTM of RT

Figure 4.7: Topographic detail of an RT region sample.



(a) DSM of KW (b) DTM of KW

Figure 4.8: Topographic detail of a KW region sample.

The OpenGF Dataset

The OpenGF dataset is a comprehensive collection of ground-annotated ALS point clouds, meticulously chosen and processed to support advanced ground filtering methodologies in diverse terrains. Its creation involved selecting only high-quality ground annotations from open-access ALS point clouds, addressing inconsistencies often observed in classification quality [181]. The dataset encompasses point clouds that represent four prime terrain types, each with its unique characteristics and challenges. These terrains include Metropolia, Small Cities, Villages, and Mountains, each typified by distinct ground and vegetation features (Table 4.3). Among these datasets, three tiles (Figure 4.11) of that, namely, S4, S8, and S9, are employed in this study, the feature of them brought in Figure 4.9 and Table 4.3.

The S4 datasets exhibit the lowest variability among the presented datasets, as indicated by their standard deviations (4.76 for DSM and 2.96 for DTM). This low variability is beneficial for achieving consistent and reliable measurements. The mean values of DSM (359.56) and DTM (356.78) are relatively close, with the DSM dataset having a slightly higher average. Both datasets' maximum and minimum values are close, indicating a consistent range of values.

The S8 datasets show significant variability, evident in their standard deviations (46.41 for

Table 4.3: Structural characteristics of OpenGF dataset, where each subset comprises three semantic classes: non-ground, ground, and noise.

Dataset	**Year**	**Coverage [km^2]**	**Semantics**	**Density (pts/m^2)**	**Topography Type**
OpenGF (S1)	2022	42	3	11.51	Metropolis with large roofs
OpenGF (S2)	2022	42	3	11.51	Metropolis with dense roofs
OpenGF (S3)	2022	42	3	11.51	Small city with flat ground
OpenGF (S4)	2022	42	3	11.51	Small city with local undulating ground
OpenGF (S5)	2022	42	3	11.51	Small city with rugged ground
OpenGF (S6)	2022	42	3	11.51	Village with scattered roofs
OpenGF (S7)	2022	42	3	11.51	Mountain with gentle slopes and dense vegetation
OpenGF (S8)	2022	42	3	11.51	Mountain with steep slopes and sparse vegetation
OpenGF (S9)	2022	42	3	11.51	Mountain with steep slopes and dense vegetation

Table 4.4: Statistical features in the three regions of OpenGF dataset (unit: meter).

Dataset	Mean	Std	Max	Min	Median	95th Percentile	5th Percentile
S9 - DSM	638.30	33.60	753.49	531.59	639.38	690.90	582.38
S9 - DTM	630.69	33.88	733.79	523.03	632.43	683.97	573.62
S4 - DSM	359.56	4.76	392.04	349.37	359.27	367.93	352.68
S4 - DTM	356.78	2.96	366.74	349.31	356.54	360.93	351.64
S8 - DSM	365.45	46.41	451.40	235.42	365.58	436.77	286.63
S8 - DTM	360.37	49.87	451.37	224.83	360.90	436.18	274.72

DSM and 49.87 for DTM). This high variability suggests a diverse set of measurements, which could indicate the heterogeneous nature of the samples or the measurement process itself. The means of the DSM (365.45) and DTM (360.37) datasets are relatively close. However, the DTM dataset shows a slightly lower mean. The ranges of the datasets are broad, emphasizing the spread of values in both cases. Interestingly, the maximum values are nearly identical (451.40 for DSM and 451.37 for DTM). However, the minimum value for DTM (224.83) is significantly lower than that for DSM (235.42).

For the S9 datasets, both DSM and DTM present similar statistical profiles with slight differences. The mean value of the DSM dataset is 638.30, slightly higher than the DTM's

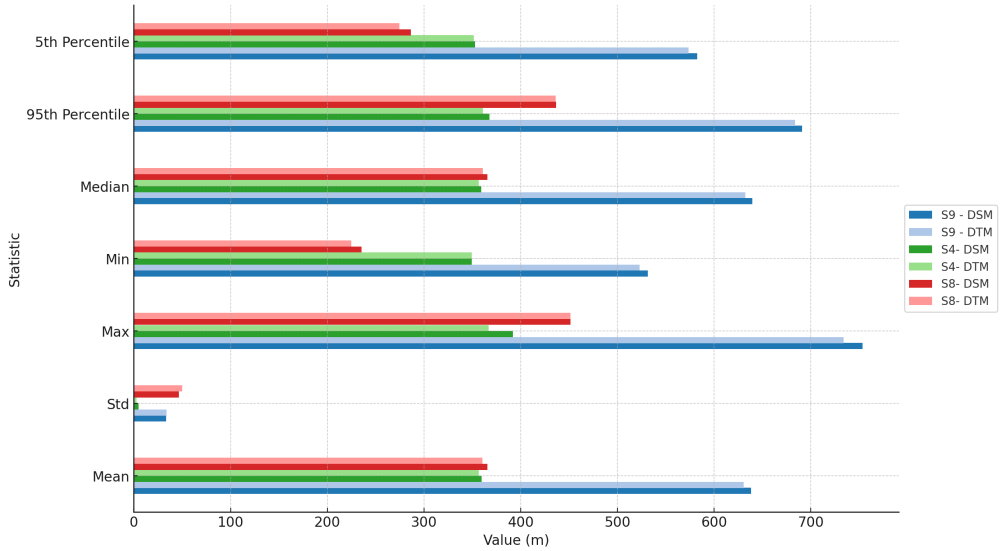


Figure 4.9: Qualitative illustration of statistical features in three regions of OpenGF dataset.

630.69, indicating a marginally higher average measurement in the DSM dataset. The standard deviation for DSM (33.60) and DTM (33.88) are nearly identical, suggesting a similar level of variability within both datasets. As expected, the maximum value is higher in the DSM dataset (753.49) than in the DTM dataset (733.79). Conversely, the minimum value is slightly lower in the DTM dataset, showing a broader range in the DSM dataset. In Figure 4.10, the normal distribution that was fitted for each subset/region has been depicted to help comprehend the position of the network's input (DSM) with regards to the network's output (DTM).

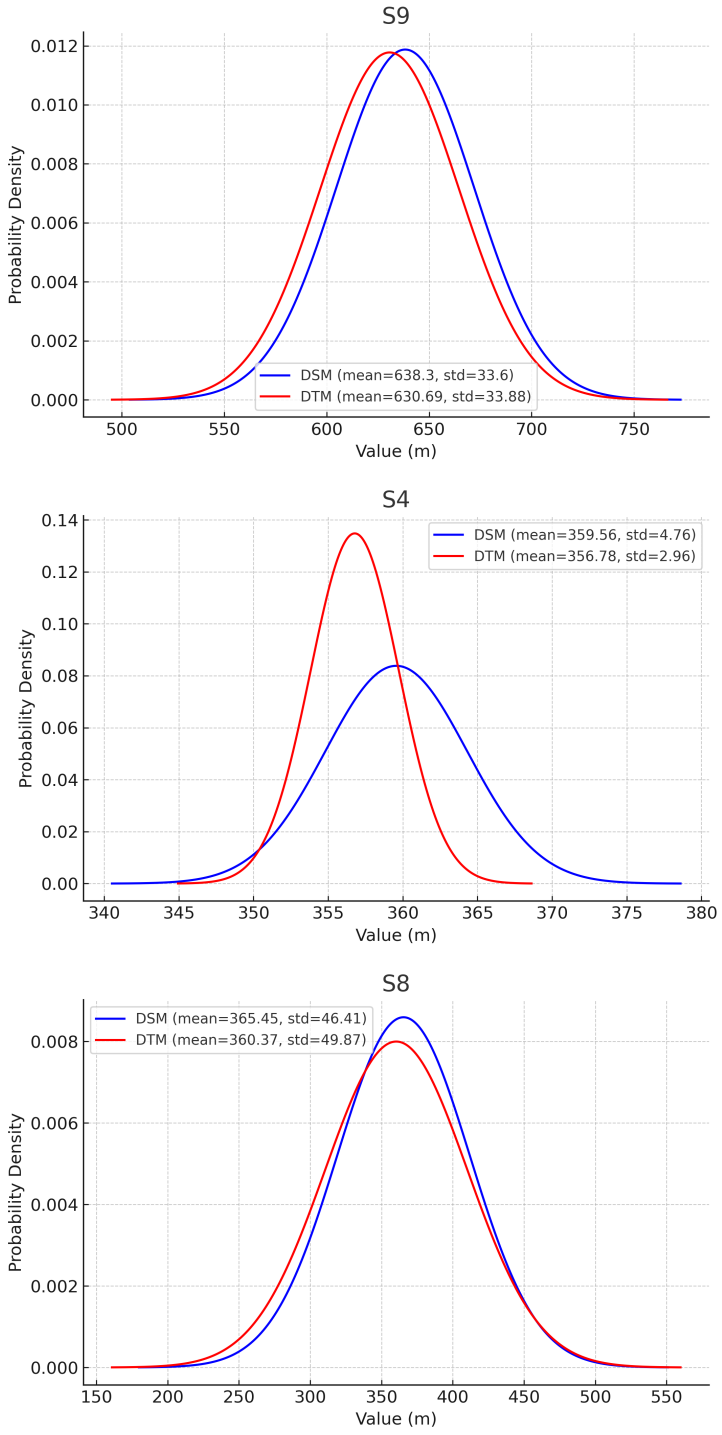
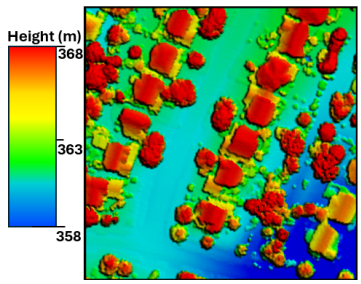
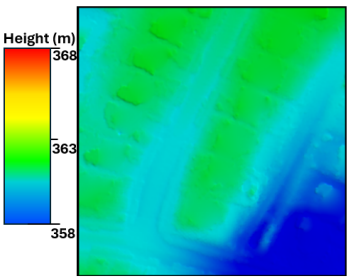


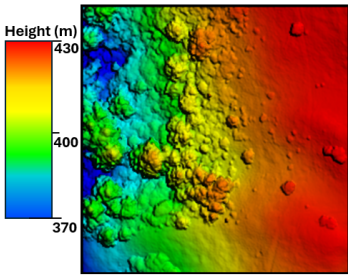
Figure 4.10: Normal distribution fits for three regions of the OpenGF dataset.



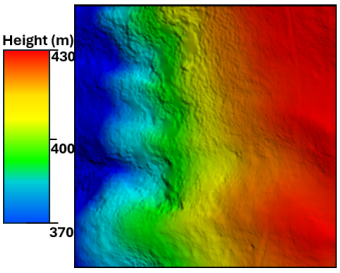
(a) DSM of S4



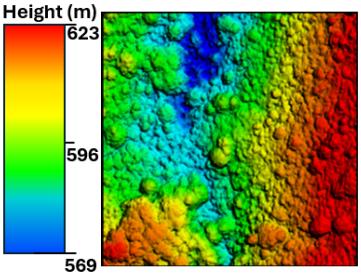
(b) DTM of S4



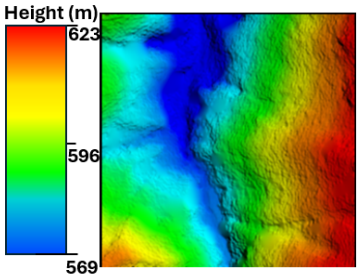
(c) DSM of S8



(d) DTM of S8



(e) DSM of S9



(f) DTM of S9

Figure 4.11: Topographic detail of three samples related to OpenGF.

ALS2DTM Datasets

The ALS2DTM project employs two distinct subsets of ALS point clouds, known as the DALES and NB datasets (see Figure 4.13). These datasets are integral to the development and evaluation of algorithms for generating DTM from ALS data [174].

The DALES dataset is designed for ongoing research in ALS point clouds and enriched with reference DTM data, thus proving valuable for training CNNs for DTM generation. It

represents a uniform terrain type, aiding in the development of models capable of effectively processing ALS point cloud data.

In contrast, the NB dataset, collected from the New Brunswick region, encompasses a wider range of elevations, including urban, rural, forested, and mountainous areas. This diversity introduces additional complexity, crucial for assessing the robustness and adaptability of DTM generation algorithms. The ALS point clouds in the NB dataset are meticulously produced and processed, ensuring high data integrity and quality.

The characteristics of these datasets are encapsulated in Table 4.5, while their semantic information is brought in Table 4.6.

Table 4.5: Structural characteristics of ALS2DTM dataset.

Dataset	**Year**	**Coverage [km^2]**	**Semantics**	**Density (pts/m^2)**	**Topography Type**
DALES	2020	10	8	50.5	Urban area
NB	2022	42	11	27.45	Urban and rural areas, forested and mountainous

Table 4.6: Semantic information of DALES and NB subsets.

Subset	**Class Details**
DALES	ground (1), vegetation (2), cars (3), trucks (4), power lines (5), fences (6), poles (7), buildings (8)
NB	unclassified (1), ground (2), low vegetation (3), medium vegetation (4), high vegetation (5), buildings (6), low point - noise (7), reserved – model keypoint (8), high noise (18)

For the DALES dataset, we observe that the DSM statistics exhibit a slightly higher mean (56.51) compared to the DTM and last-return, which have means of 53.14 and 54.68, respectively. This indicates that, on average, the surface elevation values in the DSM are slightly higher than those in the DTM and last-return datasets. The standard deviation is fairly consistent across all three, suggesting similar variability in elevation values. The

maximum value is significantly higher in the DSM (196.89) compared to the DTM, yet it is almost equal to the last-return maximum. The DSM shows a slightly less negative minimum value, which suggests the presence of fewer extremely low elevation values. The minimum values in both cases are within a comparable range. The medians are close, with the DSM's median being the highest, which aligns with its higher mean. In terms of the 95th and 5th percentiles, the DSM again shows higher values, suggesting that while the bulk of its data is similar to the other two datasets, it has higher extreme values (Table 4.7).

In contrast, the NB dataset shows a different trend. The DSM's mean (144.38) is notably higher than that of the DTM (136.77) and the last return (137.49), indicating a greater average elevation in the DSM data. The standard deviations are quite high across all datasets, indicating a wide range of elevation values, with the DSM showing the highest variability. The maximum elevation values are again highest for the DSM, reinforcing the idea that it captures higher elevation points more frequently than the DTM and last-return. The minimum values show that the DSM dataset contains the most extremely low elevation value (-18.02), which could be an outlier or indicate a wider range of elevation data captured. The median values are relatively close, with the DSM having the highest, aligning with its higher mean and suggesting a higher central tendency in elevation. The 95th and 5th percentiles for the DSM are also the highest among the three datasets, indicating that its elevation values are skewed higher both at the upper and lower ends of the dataset (Table 4.7).

Table 4.7: Statistical analyses of DALES and NB subsets (unit: meter).

Dataset	Mean	Std	Max	Min	Median	95th Percentile	5th Percentile
DALES - DSM	56.51	32.04	196.89	-3.86	62.99	98.46	2.31
DALES - DTM	53.14	31.22	101.81	-2.91	59.63	93.66	1.43
DALES - last-return	54.68	31.73	196.85	-4.50	60.85	95.94	1.85
NB - DTM	136.77	129.64	382.97	-0.21	62.81	340.77	3.27
NB - DSM	144.38	133.79	399.92	-18.02	68.74	352.26	4.32
NB - last-return	137.49	129.72	396.05	-32.65	63.52	341.47	3.44

To have a better understanding of each of the two, the fitted normal distributions of them

are illustrated in Figure 4.12.

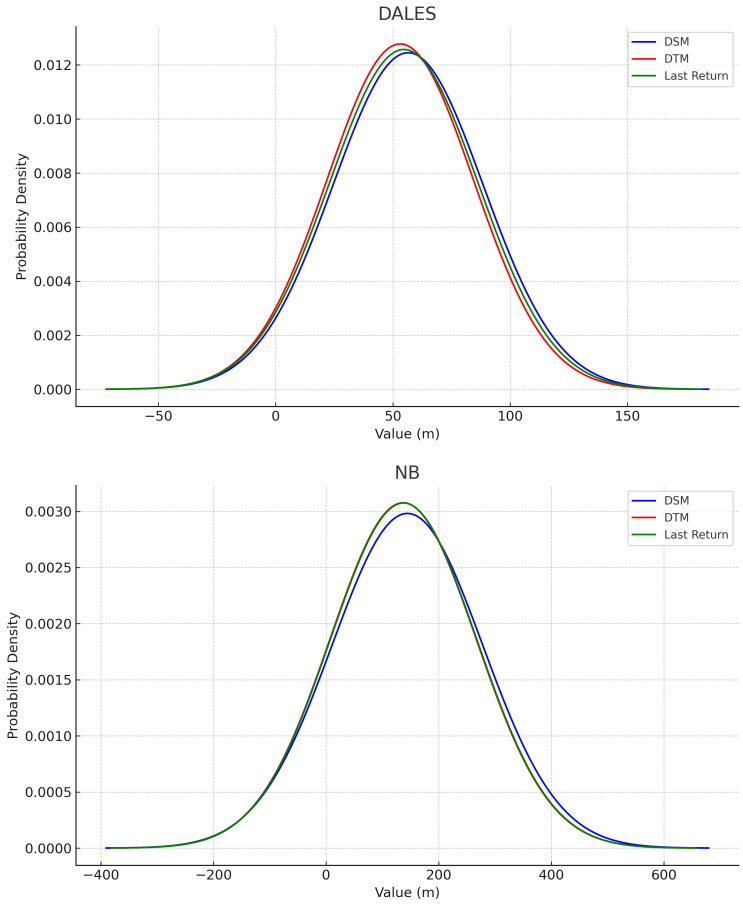
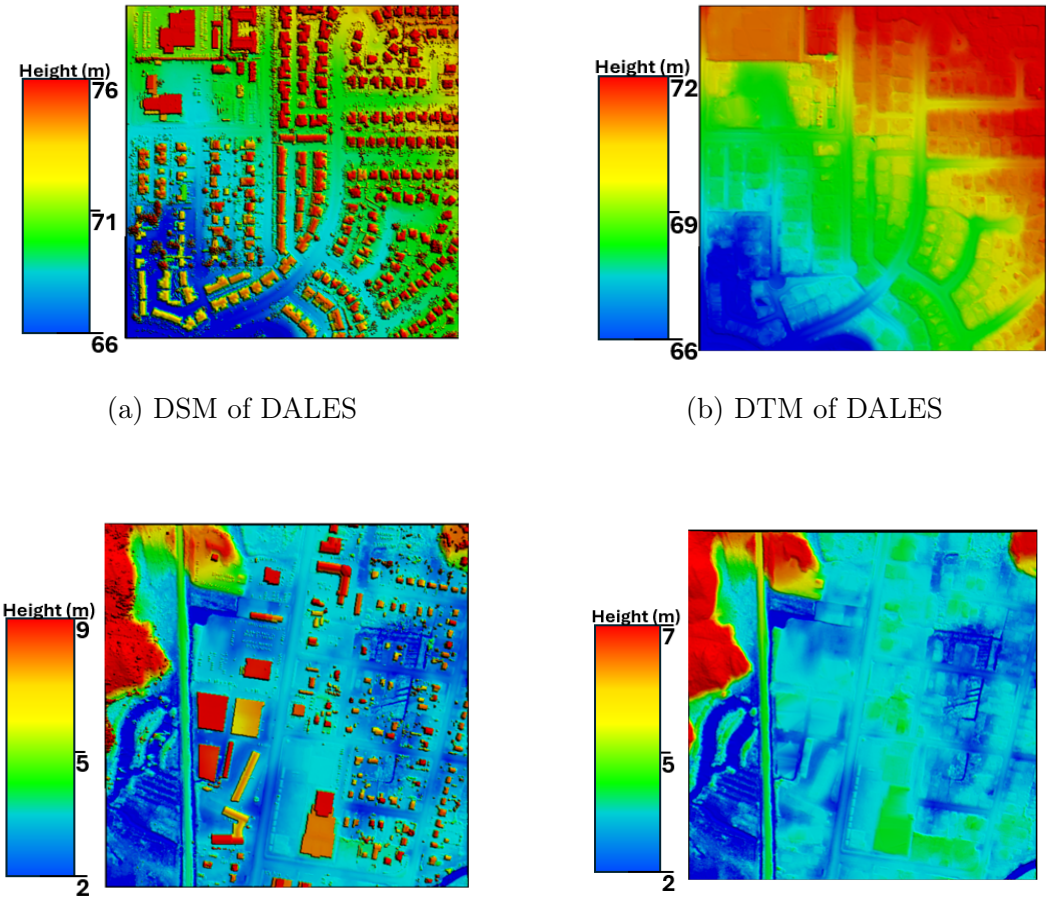


Figure 4.12: Normal distribution fits for ALS2DTM.



(a) DSM of DALES

(b) DTM of DALES

(c) DSM of NB

(d) DTM of NB

Figure 4.13: Topographic detail of two samples related to ALS2DTM dataset.

Consolidated Analysis of datasets

In order to gain a comprehensive understanding of the relationship between different datasets, a thorough examination of all datasets is carried out collectively. To facilitate this analysis, Figures 4.14 and 4.15 are presented, which graphically illustrate the distribution of DSMs and DTMs of USGS and ALS2DTM datasets, respectively. Figure 4.16 provides a detailed summary of the mean value of DSM and DTM of all datasets, along with their respective standard deviations. To facilitate a better understanding of the distinction between DSM and DTM, Figure 4.17 illustrates this difference. Additionally, the distribution of RT and

KW datasets, which are different from others, are separately depicted in Figure 4.18. These illustrations provide a clear and concise overview of the datasets' characteristics, enabling researchers to draw meaningful conclusions.

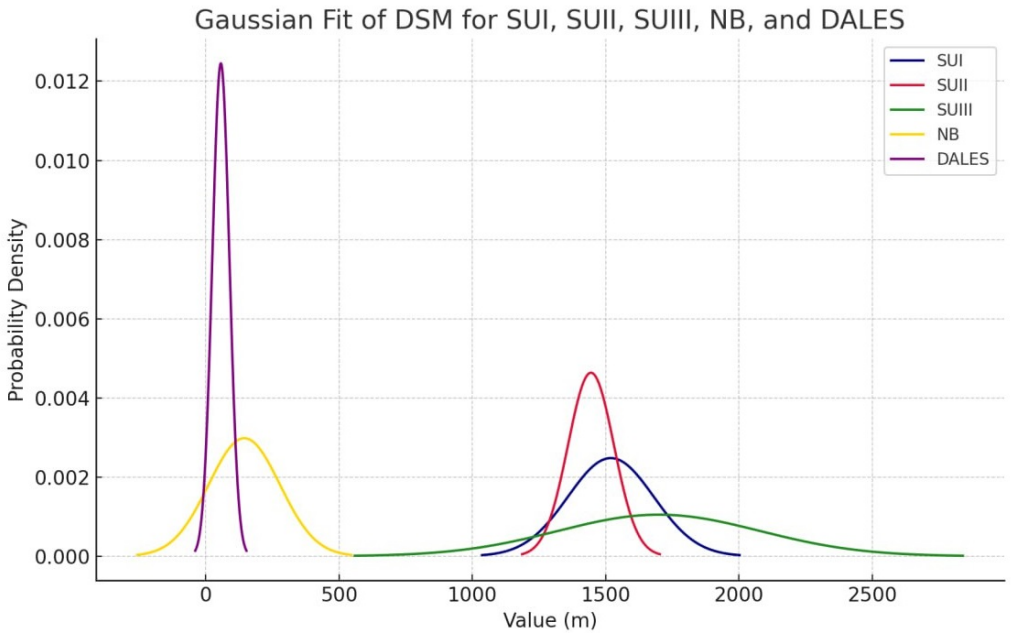


Figure 4.14: DSMs' distribution of USGS and ALS2DTM datasets.

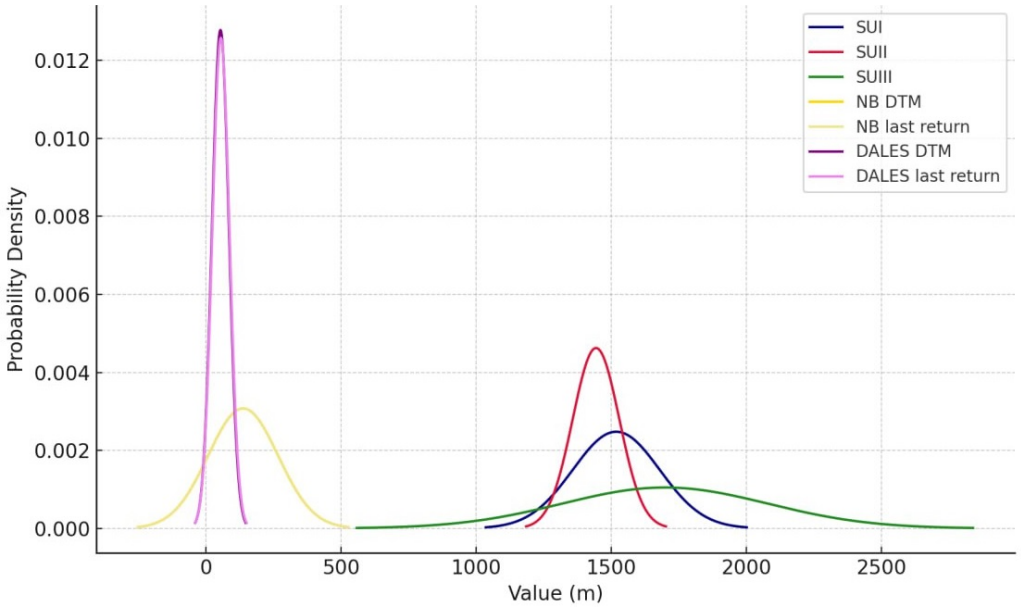


Figure 4.15: DTMs' distribution of USGS and ALS2DTM datasets.

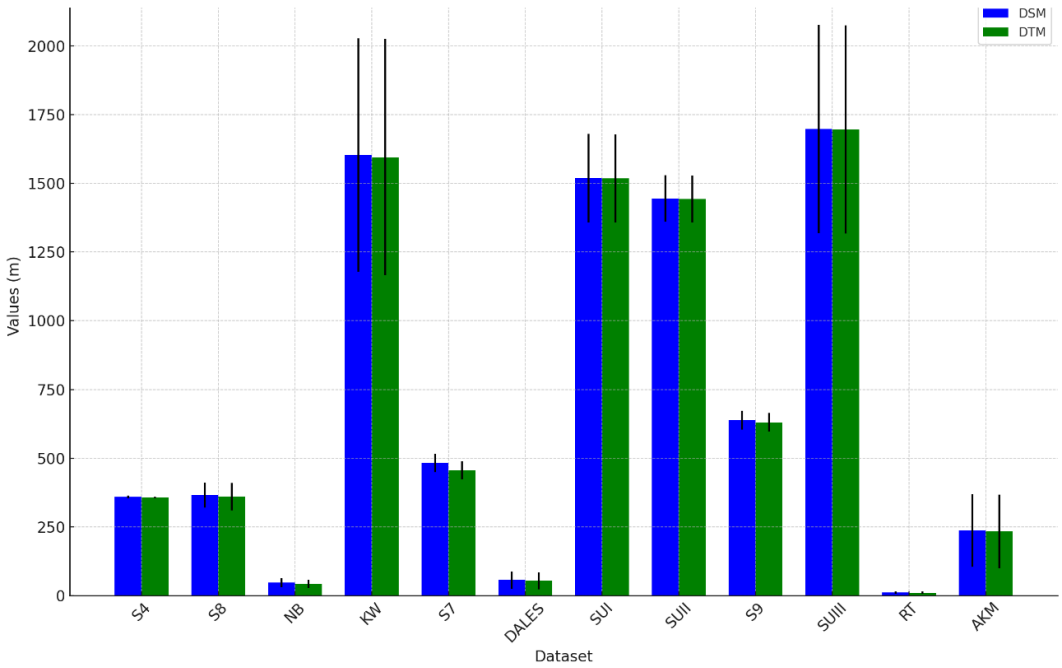


Figure 4.16: Comparison of DSM and DTM values across datasets.

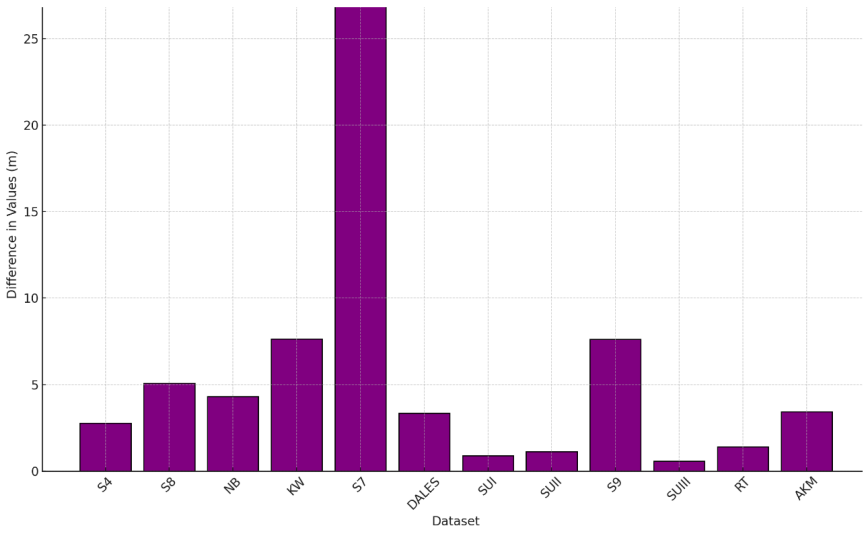


Figure 4.17: Difference between DSM and DTM values across datasets.

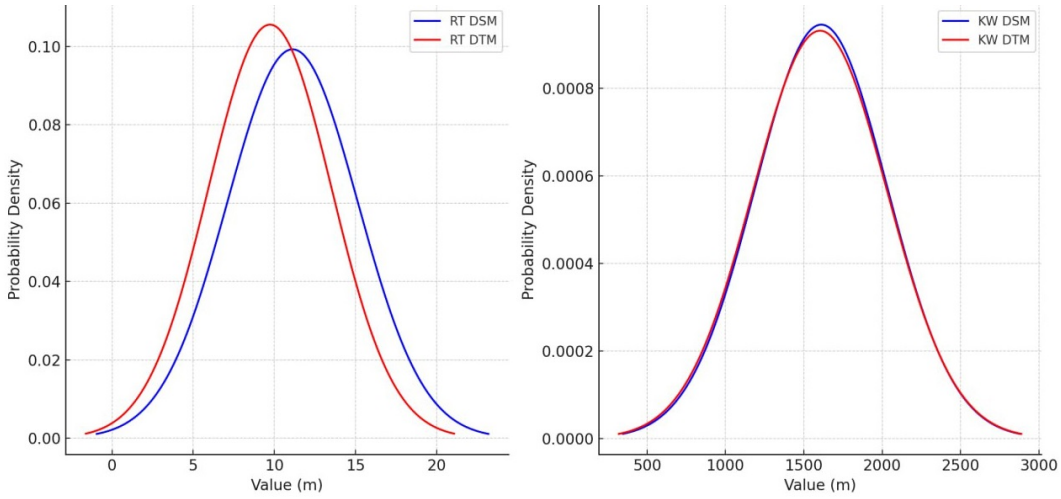


Figure 4.18: Distribution of RT and KW Regions.

Since the most relevant state-of-the-art method, HDCNN [169], has focused on ZHL, referred to as blind testing, here we also discuss the proper datasets for this purpose, which is the foundation of Chapter 5, where our proposed methods will be compared with several state-of-the-art methods. For the training dataset in chapter 5, the objective is to identify a training dataset that shares attributes similar to those of the target datasets. Among the

available options, SUI stands out as a strong candidate for the inter-domain case, where SUII, SUIII, RT, DALES, and NB are employed during the inference time. This is because it covers the same land cover as the datasets under consideration. On the other hand, the KA dataset has a distinct statistical profile, making it a better training choice for a different subset of datasets such as KW, S8, and S9. These datasets, especially KW, with their higher mean and variability, and S8 and S9, with their unique elevation ranges, could benefit from a model trained on KA due to the shared attributes of elevation variance.

4.4.2 Experimental Setup

To establish a comprehensive benchmark encompassing various terrain features, ranging from flat plains to steep mountains and from urban to forested areas, a balanced dataset is compiled for this chapter, where the efficacy of basic modules of the proposed method will be assessed. By doing so, we can also ensure that all elevation ranges presented in Figure 4.16 are covered. To make our comprehensive dataset, from each region/subset, 1000 images are randomly selected as a training without replacement, and 200 images are randomly selected as a test. This selection provides us with a comprehensive benchmark, which can be used to evaluate the performance of the proposed method. This dataset, which is a 'large mixed dataset,' has been called LMD from now on. Not only is LMD used in this chapter, but it is also utilized in the ablation study of the chapter 5.

In the preparation of our data, we implemented several key steps to ensure optimal training and results. Initially, DSMs and their corresponding DTMs were divided into images with a 50% overlap, each having a size of 256×256 pixels. These images were then localized and subsequently normalized to mitigate the effects of domain shift [169]. We then applied horizontal and vertical flips as part of our data augmentation strategy to enhance the diversity and robustness of the training set. We then incorporated horizontal and vertical flips into our Data Augmentation strategy to enhance the diversity and robustness of the training set.

The model training parameters were meticulously set, starting with a learning rate of 10E-2, gradually reducing to 10E-5 every 70 iterations throughout 250 epochs. Finally,

during inference, the weights derived from the training dataset were applied to 20% of the comprehensive benchmark, considered as a test.

4.4.3 Ablation Study

To show the efficacy of the geospatially-informed proposed method, five different ablation studies along with our network are considered, the details of which have been tabulated in Table 4.8. As shown in this table, we designed an ablation study of the proposed network by changing three modules based on the autoencoder (AE) baseline: with skip-connection or without skip-connection; with subtractive or concatenative skip-connection; with direct DTM reconstruction (i.e., directly regressing DTM from DSM input) or with indirect DTM reconstruction (i.e., indirectly regressing DTM through subtracting nDSM from DSM input). This combinatorial network architect provides six different networks, including CNETD (Concatenative NETwork with Direct DTM regression), CNETI (Concatenative NETwork with Indirect DTM regression), SUBNETD (SUBtractive NETwork with Direct DTM regression), SUBNETI (SUBtractive NETwork with Indirect DTM regression), TAED (Traditional/Typical Autoencoder with Direct DTM regression), and TAEI (Traditional Autoencoder with Indirect DTM regression), for evaluating their performance.

Table 4.8: Summary of ablation studies.

Skip Connections	**DTM Reconstruction**	**Modules**	**Description**
Concatenattion	Direct	CNETD	AE equipped with concatenative skip connections that generates the DTM directly, without any intermediate steps affecting the generation process.
Concatenation	Indirect	CNETI	The CNET leads to the generation of the DTM through an indirect process, which involves nDSM.
Subtraction	Direct	SUBNETD	This case refers to the scenario where the SUBNET generates the DTM directly.
Subtraction	Indirect	SUBNETI	This is our proposed method, named SB-SUBNET in chapter 5.
No Skip-Connection	Direct	TAED	This scenario involves the TAE generating the DTM directly, showcasing a straightforward application of the TAE's capabilities.
No Skip-Connection	Indirect	TAEI	In this case, the TAE generates the DTM indirectly via nDSM.

4.4.4 Discussion

As can be seen from Figure 4.19 SUBNETI emerges as the standout method, particularly excelling in minimizing RMSE, MAE, MSE, and ABSREL values (See section 2.6.1). This superiority can be attributed to its algorithmic efficiency in handling the absolute differences between the output and the target, as highlighted by the provided equations. The ABSREL metric, calculated as the mean of the absolute differences divided by the target, is significantly lower for SUBNETI, indicating its superior performance in relative error reduction. This is crucial because it demonstrates SUBNETI's ability to maintain accuracy proportionally across varying scales of target values, making it especially reliable in applications where precision relative to the magnitude of the measurement is critical. Conversely, TAED and

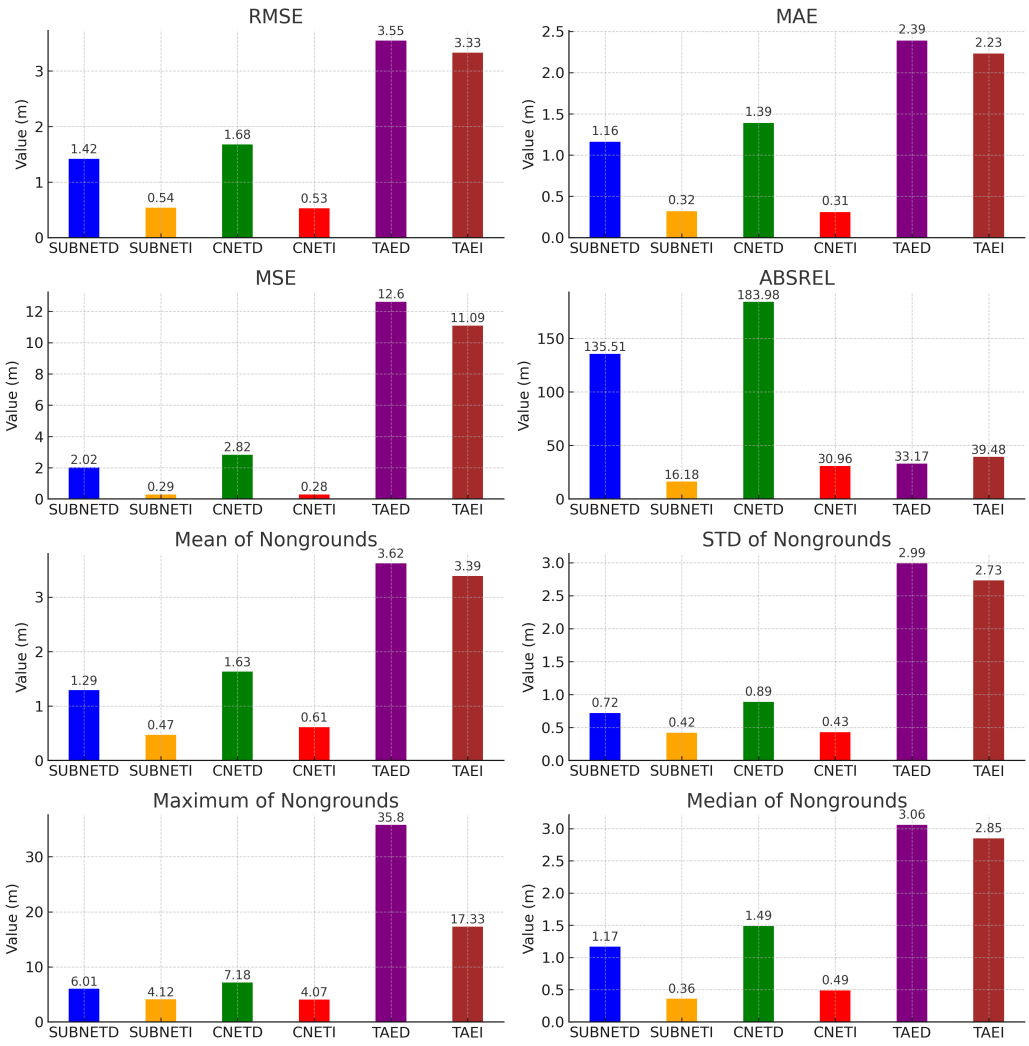


Figure 4.19: Comparative study of different versions of autoencoders, along with the proposed method, in terms of statistical and predictive metrics.

TAEI show poorer performance across these metrics, with particularly high values in RMSE, MSE, and ABSREL. The elevated MSE and RMSE values suggest a greater variance in their errors, and the high ABSREL values indicate a pronounced relative error, which could be due to less effective handling of the absolute differences between the output and the target. This inefficiency is further exacerbated in complex scenarios where maintaining a proportional relationship between errors and target values is essential for accuracy.

Regarding nonground area statistics, SUBNETI again leads with the lowest Mean, STD, and very competitive Median values. These statistics are critical for evaluating the methods' ability to closely approximate the ground truth in nonground areas, with a particular emphasis on consistency and reliability as reflected by the STD and Median values. The superior performance of SUBNETI in these criteria suggests a methodological advantage in minimizing variations and biases in nonground area estimations, likely due to more sophisticated processing of spatial data and an effective balancing of sensitivity to outliers, as evidenced by its optimal maximum of nonground and minimum of nonground values.

CNETI follows SUBNETI in performance, offering a balanced profile that, while not achieving the lowest values in every criterion, demonstrates robustness across both ground and nonground area analyses. This suggests that CNETI, similar to SUBNETI, employs effective strategies for error minimization and consistency in its estimations but may have slight limitations in either its handling of outliers or its adaptability to varying scales of target values, as indicated by its ABSREL performance.

On the other end, TAED's performance is notably less effective, especially in nonground area statistics, where it records the highest values across Mean, STD, and Maximum. This indicates a significant deviation from the ground truth, possibly due to the method's lower sensitivity to fine-grained spatial variations or a propensity to be influenced by outliers, as suggested by its high maximum of nonground values. Such characteristics might stem from algorithmic constraints or less optimized processing techniques for spatial data, leading to increased variability and less accuracy in representing nonground areas.

The analysis underscores the importance of method selection based on specific needs and

contexts. For applications requiring high relative accuracy and consistency in both ground and nonground areas, SUBNETI is clearly the preferred choice. However, for contexts where a balance between performance and possibly other considerations, such as computational efficiency or ease of implementation, is needed, CNETI presents a viable alternative. The lesser performance of TAED and TAEI, particularly in nonground area statistics, highlights the challenges in optimizing for both accuracy and consistency, emphasizing the need for further development or application-specific adjustments to these methods.

Following a comparative analysis of various states of our proposed method, this report compares SUBNETI with its competitor, CNETI. In order to determine the stability of the two methods, a comparison was made between them. To achieve this, a 5-fold cross-validation was conducted for both methods, and the results were documented in Table 4.9.

Table 4.9: Comparative performance analysis of the proposed method and its leading competitor, CNETI, across statistical and predictive metrics (unit: meter).

Metric	CNETI Avg	CNETI Std	CNETI PI	SUBNETI Avg	SUBNETI Std	SUBNETI PI	Efficacy
RMSE	0.533	0.043	0.576	0.538	0.028	0.566	SUBNETI
MAE	0.334	0.035	0.369	0.334	0.021	0.355	SUBNETI
MSE	1.513	1.298	2.811	1.375	0.798	2.173	SUBNETI
ABSREL	13.671	10.462	24.133	10.430	6.026	16.456	SUBNETI
Mean	0.496	0.082	0.578	0.469	0.011	0.480	SUBNETI
Std	0.451	0.037	0.488	0.454	0.023	0.477	SUBNETI
Max	4.155	0.204	4.359	4.178	0.213	4.391	CNETI
Median	0.371	0.082	0.453	0.344	0.011	0.355	SUBNETI

The stability of a system can be measured by its Std and the PI values, which is a linear combination of Std and the average error (see section 2.6.1). A lower value of Std and PI indicates a more reliable performance. When it comes to stability, SUBNETI outperforms other systems, as it generally exhibits lower Std values for metrics including RMSE, MAE, MSE, ABSREL, Mean, and Median.

Efficiency can be inferred from the MSE and RMSE values, metrics that quantify the error magnitude in predictions. SUBNETI and CNETI show comparable performance in

terms of MAE. However, SUBNETI edges slightly in RMSE, suggesting it is more efficient at minimizing squared errors, a crucial aspect when high error magnitudes are significantly penalized. The lower PI values for SUBNETI across most metrics further support its efficiency, indicating it performs well on average and does so with more excellent reliability.

Accuracy is often assessed through metrics like RMSE, MAE, and ABSREL, which directly measure error magnitude and relative error. SUBNETI's marginally better RMSE and significantly better ABSREL values suggest it is more accurate in capturing the actual signal with fewer and less severe errors, particularly in relative terms. This is essential when accuracy relative to the magnitude of the target values is critical, as in many Geospatial applications where the range of expected values can vary widely. While SUBNETI and CNETI have almost similar maximum values, the former yields more favorable outcomes overall. This is primarily due to its higher median, mean, and PI values, which indicate a more balanced and consistent performance. Consequently, SUBNETI is deemed to provide better and more reliable results compared to CNETI.

Subsequent analysis entails evaluating each dataset's performance by comparing the RMSE, maximum error, and distribution of median error between the two. The RMSE comparison plot, Figure 4.20, immediately highlights the superior performance of SUBNETI across all datasets. This superiority is particularly noteworthy in datasets with high variability, such as S8 and KW, where the standard deviation in both DSM and DTM is significant. The ability of SUBNETI to achieve lower RMSE values in these contexts suggests it is better equipped to handle the complexity introduced by terrain variability. This is crucial in applications where accurate ground-level estimation is paramount, as it indicates a consistent ability to closely match the ground truth DTM, even in challenging environments.

When considering the maximum error comparison plot alongside the dataset characteristics, Figure 4.21, we gain insight into each method's ability to manage outliers and extreme values. For example, the S8 dataset, characterized by its high DSM and DTM standard deviations, presents a challenging scenario for DTM estimation. The plot reveals that SUBNETI generally produces lower maximum errors compared to CNETI, indicating its robustness

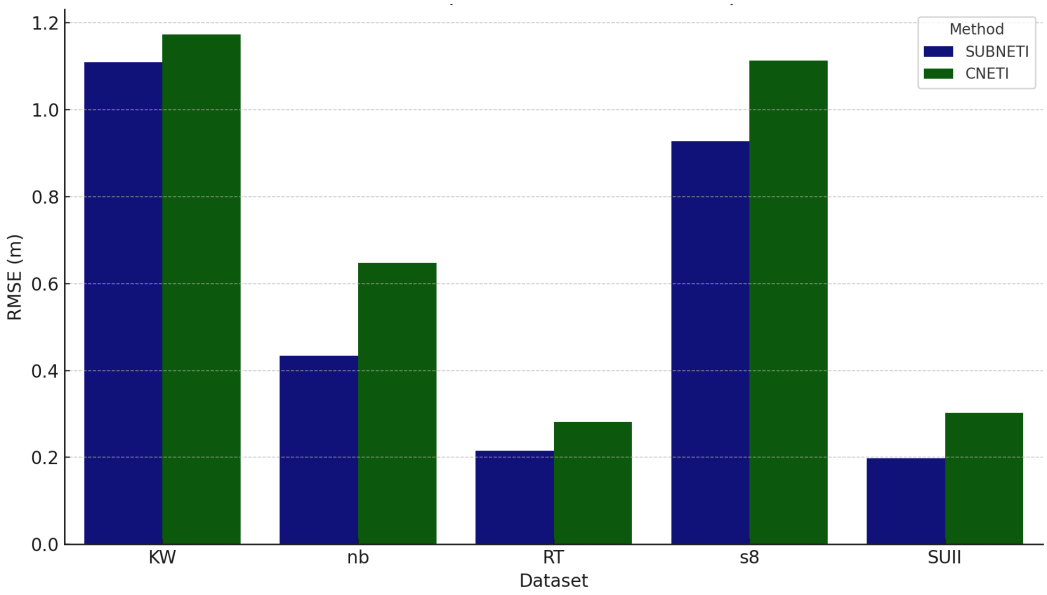


Figure 4.20: Visual inspection of the proposed method and its leading competitor, CNETI, focused on RMSE.

against extreme terrain variations. This aspect of performance is crucial for applications where even rare, extreme errors can have significant consequences, affirming the value of SUBNETI's approach in maintaining accuracy across various conditions.

The median error along with std, brought in Table 4.9, provide a representation of the variability and central tendency of errors for both methods. In datasets like RT and SUII, which exhibit lower variability in their DSM and DTM characteristics, SUBNETI not only maintains lower median errors but also demonstrates a tighter distribution of errors, as evidenced by the smaller interquartile ranges and the standard deviation represented by the vertical dashed lines. This tighter error distribution underscores SUBNETI's precision and reliability, showcasing its effectiveness in providing consistent DTM estimations across different terrains.

Integrating the above-mentioned results with the DSM and DTM characteristics, highlighted in Figure 4.16, allows for a more comprehensive assessment of the methods. The variability and complexity captured in the dataset characteristics illuminate why SUBNETI's performance is particularly commendable. Its ability to handle datasets with high variability

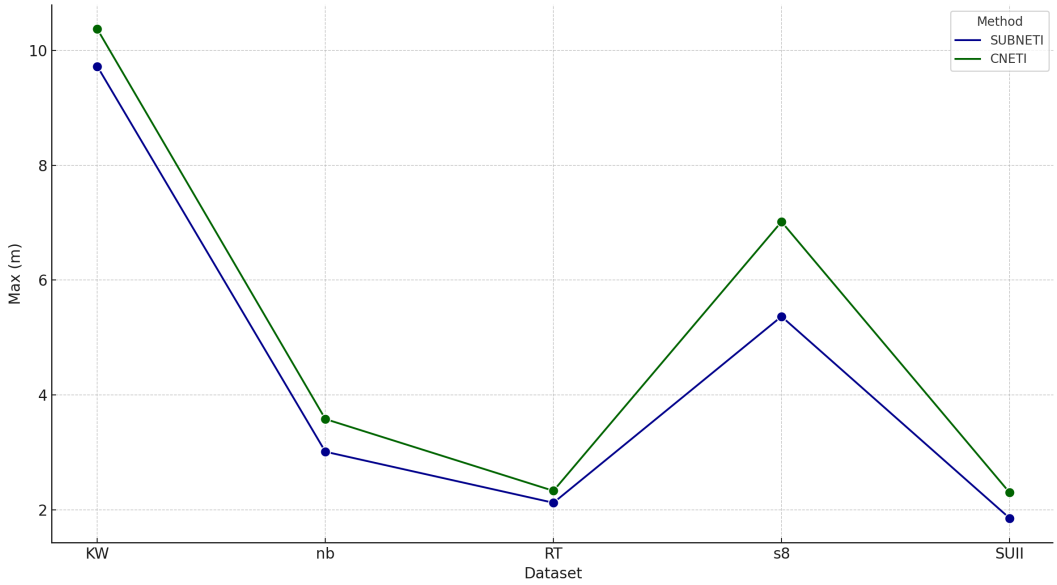


Figure 4.21: Visual inspection of the proposed method and its leading competitor, CNETI, focused on maximum error.

(e.g., S8 and KW) while still minimizing RMSE and maximum errors highlights its robustness and adaptability. Conversely, its precision in datasets with lower variability (e.g., RT and SUII) showcases its reliability and consistency, making it a versatile tool for DTM estimation. Moreover, the plots serve as a visual testament to these findings, with the RMSE and Max (m) comparisons directly reflecting SUBNETI's superior performance across a range of conditions and the box plots for Median (m) with Std Deviation offering a granular view of the methods' error distributions.

4.4.5 Summary

This chapter concludes by detailing the development and evaluation of a new autoencoder, designed to efficiently generate DTMs from DSMs in a geospatially-informed manner. The research's foundation is based on a comparative analysis between the proposed method, SUBNETI, and several rivals, which aim to highlight the effectiveness of the subtractive concept in DTM extraction. A series of ablation studies were conducted to achieve this goal. The first finding of experimental results that draws one's attention is the fact that the indirect

cases for all the methods at hand lead to better results than the direct case. This means generating nDSM as a part of the network and then estimating DTM from that using a physical equation can lead to better results. Experimental results have also definitively underscored the superior capabilities of SUBNETI in terms of accuracy, efficiency, and stability across diverse terrains and conditions, especially compared to CNETI. The only difference of CNETI with the proposed method was its skip connections which are concatenative compared to subtractive. The comparative stability, efficiency, and accuracy analyses between SUBNETI and CNETI, particularly within the context of a 5-fold cross-validation, reveal SUBNETI's consistent outperformance. It is the more stable and reliable method, demonstrating lower standard deviation and performance index values across critical metrics. This consistency is paramount in geospatial applications where precision and reliability in DTM estimation can significantly impact environmental monitoring, urban planning, and disaster management strategies. Furthermore, the dataset-specific performance analysis illuminates SUBNETI's adeptness at managing complex terrain variability. Its ability to consistently achieve lower RMSE and maximum error rates across datasets with high variability, such as S8 and KW, showcases its robustness and adaptability. Additionally, the precision of SUBNETI in datasets with lower variability, like RT and SUII, through tighter error distributions and lower median errors, affirms its reliability and effectiveness in delivering accurate DTM estimations across a spectrum of environmental conditions.

While SUBNETI is an effective tool for producing impressive results, it does have a major drawback when used as a regressor. It modifies all the pixels in DSM, including those that are associated with the ground and obtained from LiDAR, which are highly precise, leading to a reduction in their accuracy. However, DTM is a type of DSM that requires regression of non-ground points. To address this problem, the network needs to consider ground segmentation simultaneously to just regress non-ground points.

Chapter 5

Geospatially informed Multi Task Autoencoder for DTM Reconstruction

5.1 Introduction

Depth completion is a complex process that involves reconstructing comprehensive depth information from partial or obscured depth maps or images. The integration of deep learning into this field has brought about a paradigm shift, offering robust methodologies for the intricate analysis and processing of complex depth data [182]. This advancement is particularly transformative in applications that require high precision in depth perception, such as autonomous vehicle navigation [183], where the accuracy of depth information is crucial for ensuring the safety and efficacy of navigation systems.

Beyond autonomous navigation applications, depth completion also plays an important role in refining geospatial data, particularly in the derivation of DTMs from DSMs [164]. The essence of this process is the intelligent interpolation of terrain data, where the removal of non-terrain elements—like buildings and foliage from DSMs—leaves voids that must be filled. The strategies deployed for depth completion fall into two categories: blind and non-blind, each offering a distinct pathway to the holistic reconstruction of depth information.

Blind depth completion involves scenarios where the locations of missing data within

depth images or maps are unknown a priori [184]. Algorithms under this classification are engineered to autonomously detect missing data, often through advanced machine-learning techniques that analyze patterns and anomalies in the data [185]. The complexity of blind depth completion lies in developing a system capable of not only detecting but also filling these gaps, a process that often involves sophisticated models trained to understand and interpret the depth data's intricacies [185].

Contrastively, non-blind depth completion operates with a priori knowledge of the gaps in the depth data, typically identified through segmentation techniques or other analytical methods [186]. With the areas of missing data already defined, the focus shifts to accurately filling these gaps, utilizing the contextual information provided by the surrounding data. This predefined knowledge allows for a more targeted approach, where the algorithm can directly address the identified areas, often leading to more efficient and precise depth reconstruction. The distinction between blind and non-blind depth completion methods is crucial, as it dictates the approach and complexity of the algorithmic solution.

Non-blind methods benefit from having the initial task of identifying gaps already accomplished, allowing the algorithms to focus exclusively on data reconstruction [186]. However, in blind depth completion, the challenge is twofold: the algorithm must first identify areas lacking depth information and then fill them in a manner consistent with the surrounding data. This often necessitates an end-to-end DL approach, where the DL model is trained to handle both tasks simultaneously [185]. Although non-blind methods may seem attractive at first glance, they are not feasible in most applications because the areas with gaps are not known in advance. More importantly, in certain situations, such as DSM filtering, the gap recognition process is reliant on depth estimation. This creates a complex issue where both factors need to be considered together. It is similar to the "egg-or-chicken" problem, where it is difficult to determine the cause-and-effect relationship. Accordingly, in this chapter, an attempt is made to present a novel method for blind task execution in DTM generation.

DTMs, as depicted in Figure 5.1, offer a sophisticated 2.5-dimensional visualization of the earth's bare surface. These models are invaluable across a diverse range of fields, from

geomatics to environmental sciences, enabling critical applications such as flood modeling [187, 188], biomass estimation [189], and infrastructure planning [190].

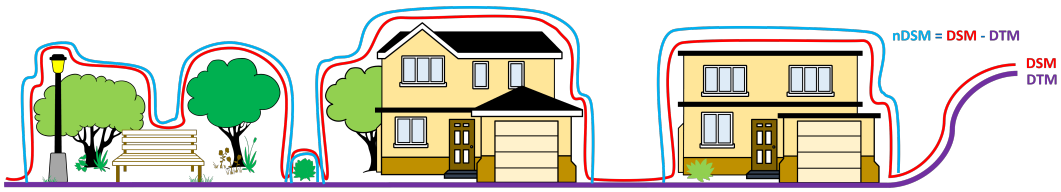


Figure 5.1: A scheme of three different surface models.

DTMs are usually extracted from DSMs through filtering methods. DSMs either come directly from aerial/terrestrial laser scanners [191] or indirectly from multi-image reconstruction [192]. This transformation from 3D point clouds to a 2.5D raster representation is an important process that preserves crucial topographical information, thereby maintaining the integrity of the original data. This conversion process, known as DSM image generation, presents two primary advantages: firstly, DSM images offer a more structured and consistent data framework, which significantly enhances the application of image processing techniques and advanced methods like convolutional neural networks for ground filtering [193]. Secondly, this conversion markedly reduces the computational intensity required to process extensive 3D point clouds, a critical factor in large-scale environmental analysis [194].

The efficiency and structured nature of DSMs, resulting from this conversion, set the stage for the next critical step in terrain analysis: the generation of DTMs. The methods employed in DTM generation vary widely and can be broadly categorized into unsupervised and supervised techniques, each offering unique approaches to terrain extraction. Unsupervised methods, which are conventionally referred to as traditional methods, encompass slope-based, morphology-based, interpolation-based, segmentation-based, and statistics-based methods. These methods rely on pre-defined rules and algorithms to extract terrain information from the input data. In contrast, supervised methods, which are based on artificial intelligence, utilize machine learning algorithms to recognize terrain features and generate DTMs. The selection of an appropriate method depends on the specific application and the characteristics of the input data.

Slope-based methods are a critical component of unsupervised techniques, utilizing a slope threshold to differentiate terrain features, as depicted by the equation:

$$\text{Feature} = \begin{cases} \text{High Slope}, & \text{if Slope} > \text{Threshold} \\ \text{Low Slope}, & \text{otherwise} \end{cases} \tag{5.1}$$

These methods demonstrated their success in areas with significant changes in slope [164]. However, they may not perform well in rugged and complex terrains [163]. On the other hand, morphology-based methods utilize morphological operations to enhance the terrain features and eliminate non-ground elements. These methods have proven to be effective in various landscapes but depend on the selected structural elements [165].

Interpolation-based methods address the challenge of estimating terrain heights in unsampled areas, employing techniques like TINs [166], Kriging, and IDW, known for their interpolation strengths yet limited in extrapolation capabilities. Segmentation-based methods, vital in unsupervised settings, classify point clouds into the ground and non-ground points, often necessitating further interpolation to refine the DTM [167, 168]. Lastly, statistics-based filters assume that ground points follow a certain statistical distribution, identifying non-ground points by their deviation from this expected distribution [195]. These methods are particularly adept in environments where ground points exhibit consistent statistical characteristics, offering a unique approach to ground filtering compared to more geometrically focused methods [172].

These conventional methods, despite their effectiveness, encounter challenges in complex environments such as mountainous regions or areas with dense, low-height vegetation, as previously discussed. To address these limitations, recent advancements have turned towards DL methodologies, particularly CNN-based approaches. These deep learning-based methods are primarily categorized into segmentation-based [196] and regression-based methods. To address this problem, some researchers, especially those who work directly with 3D point clouds, compare just segmented points as ground with their corresponding ground truth [197]. However, it is important to note that such methods do not provide a smoothed DTM, as

their focus is solely on separating ground points from other features. Therefore, while these techniques may be useful for ground point identification, they do not account for the nuances of terrain and may not be suitable for more complex analyses that require a more refined DTM. It is crucial to consider the limitations of these methods when applying them in research and business settings. Moreover, a critical aspect often overlooked is the high correlation between segmentation and height estimation, which necessitates a joint consideration.

In contrast, regression-based methods, such as GANs including pix-to-pix frameworks, referenced in [175, 174], and multi-resolution approaches like those proposed by [169], offer an alternative pathway, where terrain surfaces are estimated through regression using DL networks. Despite their advantages, these methods, due to the lack of explicit classification information, tend to significantly alter ground points. Such an approach can result in unnecessary modifications of ground points that have been accurately captured using costly laser technology. Consequently, developing methods that maintain the integrity of ground values in DTM estimation becomes crucial.

To effectively address these issues, this chapter introduces a novel geospatially informed multi-task autoencoder architecture, named Double Branch SUBtractive NETwork (DB-SUBNET), featuring a joint encoder and dual decoders. geospatially informed DTM reconstruction embodies the application of domain-specific knowledge—principles derived from physical geography and environmental science to inform the neural network's design. This approach employs the DSM - nDSM = DTM relationship, which is grounded in the physical reality that DSM encapsulates both the earth's topography and above-ground objects. The network's architecture incorporates this physical awareness to enhance the model's predictive abilities, aligning them with the physical characteristics of the terrain it seeks to replicate. This methodology is not only innovative, but it is also vital for ensuring the accuracy and reliability of DTMs in various applications. The proposed design effectively addresses the challenges of terrain segmentation and gap-filling in a unified manner by leveraging a double-branch decoder, whose branches are segmentation and regression ones. The regression branch works to rebuild DTM by filling in gaps, which are also referred to as nongrounds. This is

done by leveraging physical relationships and subtractive skip connections. The segmentation branch is responsible for detecting these gaps and is dynamically updated based on the features derived from the regression branch. The key contributions and observations of our study are:

- Introducing subtractive skip connections embedding domain-specific knowledge in our network, making it a geospatially informed model that reflects the underlying terrain process.
- Employing a dual-branch network, our approach innovatively combines segmentation and regression. This method ensures accurate non-ground element estimation for DTM generation while preserving the valuable ground point data from the DSM.
- Adopting a bi-task learning approach, our method acknowledges the high correlation between elevation mapping and segmentation, as discussed in [198, 199]. This integrated strategy leads to improved accuracy in both DTM generation and segmentation tasks.
- Validating our algorithm against three state-of-the-art benchmarks in 10 varied areas, we significantly enhanced our research's robustness, showcasing the algorithm's adaptability and marking a notable contribution.

5.2 Related Work

MTL represents a significant paradigm shift in machine learning, characterized by the simultaneous solving of multiple learning tasks while sharing a common representation [200]. This approach is advantageous over single-task learning in several aspects. By enabling models to leverage shared information among related tasks, MTL enhances learning efficiency and prediction accuracy. This is especially valuable in scenarios with limited data, as it reduces the risk of overfitting on individual tasks. Furthermore, MTL promotes the development of more robust models by facilitating a more generalized representation, thus enabling better performance in complex scenarios where single-task models may not suffice.

In the field of computer vision, depth estimation and completion are pivotal applications of MTL frameworks, demonstrating the inherent synergy in jointly learning-related tasks [201]. Depth estimation entails the process of predicting the distance between objects and the observer, a critical component for numerous applications such as autonomous driving, robotic navigation, and augmented reality. On the other hand, depth completion involves the refinement of depth maps, where the goal is to fill in gaps or rectify uncertain depth information, often derived from sparse or noisy sensor data. These tasks are frequently integrated within MTL frameworks alongside other related computer vision tasks, such as semantic segmentation or object detection. Semantic segmentation categorizes each pixel in an image into a predefined class, which, when combined with depth estimation, provides a more detailed understanding of the scene. Object detection identifies and locates objects within an image, and when coupled with depth information, it allows for a three-dimensional understanding of object positions and sizes. The integration of depth estimation and completion with tasks like semantic segmentation or object detection in MTL frameworks leverages the contextual information from one task to enhance the performance of others. For example, understanding the semantics of a scene can significantly enhance depth estimation accuracy. Objects or environmental features with known dimensions or typical distance ranges provide valuable cues for depth estimation algorithms. Such contextual understanding can lead to more robust and accurate depth predictions, as the model can infer depth based on known characteristics of similar objects or environments observed in the training data [202]. Moreover, the use of MTL approaches in this domain encourages the sharing of features and representations across tasks, which can lead to more efficient and generalizable models. By learning tasks concurrently, the model can exploit commonalities and correlations between tasks, improving the overall performance compared to learning each task in isolation.

Significant contributions in this domain have emerged where depth estimation and segmentation are integrated using MTL. For example, [203] leverages depth completion as an auxiliary task to align features across domains for LiDAR semantic segmentation. This approach has shown notable improvements in multi-modal Unsupervised Domain Adaptation

(UDA) benchmarks, underscoring the potential of MTL in enhancing feature alignment and performance in complex scenarios. In this category, the work by [204] introduces a novel method for 3D instance segmentation that targets volumetric scene representations obtained with depth sensors or multi-view stereo methods. By learning shape information and instance boundaries, this method addresses the critical challenge of instance segmentation, demonstrating the effectiveness of MTL in extracting and utilizing geometric information for improved scene understanding. Further contributing to this field [205] introduced a joint depth and segmentation prediction network within the Multi-view Stereo (MVS) framework, designed to interpret 3D scenes from a few images. Unlike previous methods that are either monocular or require a complete dense scan, this approach utilizes the Segment Anything Model (SAM) to provide robust semantic features that guide the creation of more accurate cost volumes for MVS. [206], have introduced a joint model that explores the relationship between semantic segmentation and depth completion. By employing boundary features as constraints, this model enhances the performance of both tasks, showcasing the benefits of integrating depth and semantic information in a unified framework. This approach not only improves accuracy but also provides a more nuanced understanding of scene geometry and semantics. Moreover, [207], which jointly performs depth prediction and semantic scene segmentation, exemplifies the power of MTL in creating temporally consistent predictions. Utilizing a recurrent network with skip connections and adversarial training, this method ensures the coherence and reliability of the predictions, which is essential for dynamic and evolving environments.

These studies collectively highlight the significance of integration strategies in MTL, where shared features or constraints are leveraged to bolster the accuracy and consistency of predictions across tasks. The synthesis of depth and semantic information not only enriches the scene understanding but is also pivotal for real-time applications such as navigation and interaction in complex environments.

5.3 Methodology

5.3.1 Problem formulation

This study introduces an autoencoder architecture that incorporates a joint encoder and two distinct decoders designed to predict DTM from DSM, as illustrated in Figure 5.2. The encoder is tasked with converting the input DSM into a latent representation within a lower-dimensional space. The DSM is described as 2.5-dimensional geospatial data, where each pixel is characterized by its image coordinates (x, y) in the pixel grid, along with a positive scalar value z denoting elevation. This relationship can be articulated as a mapping from $\{(x, y) \in \mathbb{R}^2 \mid x \in \mathbb{N}, y \in \mathbb{N}\}$ to $z \in \mathbb{R}_+$, where $\mathbb{R}_+$ symbolizes the set of positive real numbers, and $\mathbb{N}$ is the set of positive integers, excluding zero.

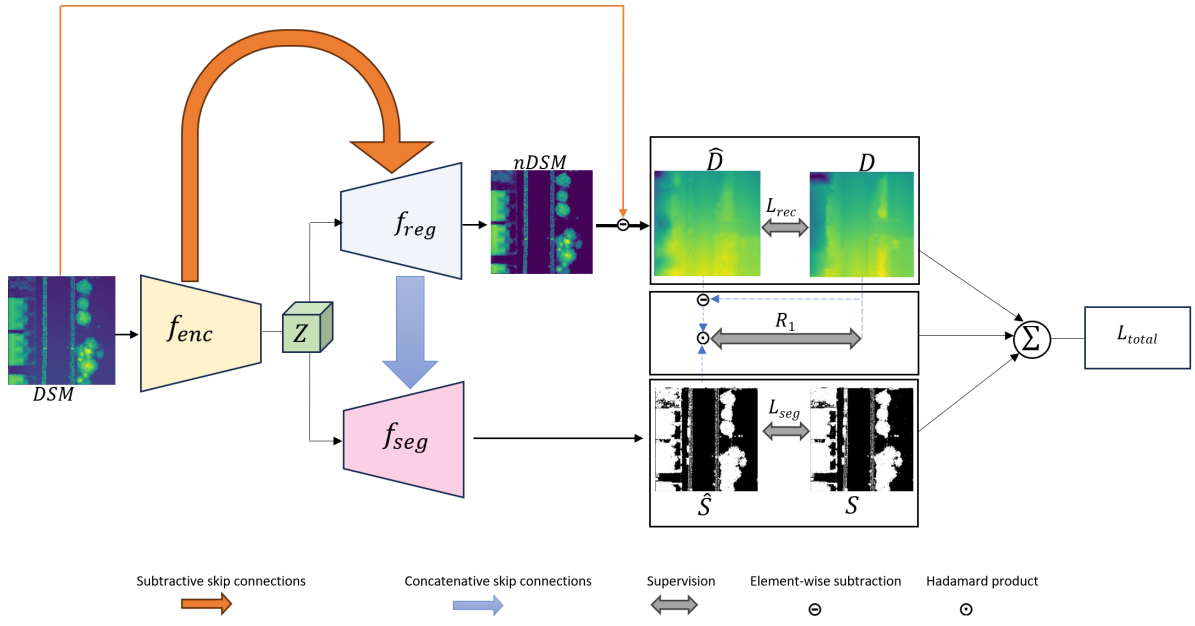


Figure 5.2: Architecture of the proposed method.

The encoder function f_{enc} maps the input DSM into a lower-dimensional latent space, creating a compressed representation denoted by the matrix $\mathbf{Z}$. This representation is

described by the dimensions d, p, and q, where d stands for the number of features (depth), and p and q represent the reduced spatial dimensions of the data in latent space, constituting a tensor in $\mathbb{R}^{d\times p\times q}$. The encoding process optimizes the encoder parameters θ_{enc}, efficiently encapsulating the DSM's significant topographical features (Equation 5.2).

$$\mathbf{Z} = f_{\text{enc}}(\text{DSM}; \theta_{\text{enc}}) \tag{5.2}$$

The first decoder in our dual-decoder setup, which can be referred to as the regression decoder, employs the function f_{reg} to reconstruct the DTM from the latent representation $\mathbf{Z}$. It does so by adjusting its parameters θ_{reg} to predict the DTM values $\hat{D}$ (Equation 5.3).

$$\hat{D} = f_{\text{reg}}(\mathbf{Z}; \theta_{\text{reg}}) \tag{5.3}$$

The second decoder, known as the segmentation decoder, utilizes the function f_{seg} to generate a binary segmentation map $\hat{S}$. In this map, a value of 1 indicates areas classified as non-ground, such as vegetation or structures. This decoder adjusts its parameters θ_{seg} for the task (Equation 5.4).

$$\hat{S} = f_{\text{seg}}(\mathbf{Z}; \theta_{\text{seg}}) \tag{5.4}$$

The output of the segmentation decoder is passed through a sigmoid activation function σ, see Section 2.3.3, ensuring the output values fall within the range [0,1], thereby defining the binary classification for each pixel in the DSM (Equation 5.5).

$$\hat{S} = \sigma(f_{\text{seg}}(\mathbf{Z}; \theta_{\text{seg}})) \tag{5.5}$$

5.3.2 Regression Task

The regression autoencoder, as can be seen from Figure 5.3, is equipped with subtractive skip connections to reconstruct $\hat{D}$. In detail, Each layer l in the encoder performs a transformation

represented by the Equation 5.6.

$$h_i = f^i_{\text{enc}}(h_{i-1}), \tag{5.6}$$

where $h_0 = \text{DSM}$ and $h_{\text{end}} = Z$, as depicted in Figure 5.3, and f^i_{enc} is the transformation function at the ith layer. On the other hand, the decoder's regression branch transforms the latent representation Z into the nDSM. The transformation at each decoder layer of the regression branch i can be represented as

$$\hat{h}_i = f^i_{\text{reg}}(\hat{h}_{i-1}), \tag{5.7}$$

where $\hat{h}_0 = Z$ is the latent representation, and $f^i_{\text{regression}}$ is the corresponding layer's transformation function.

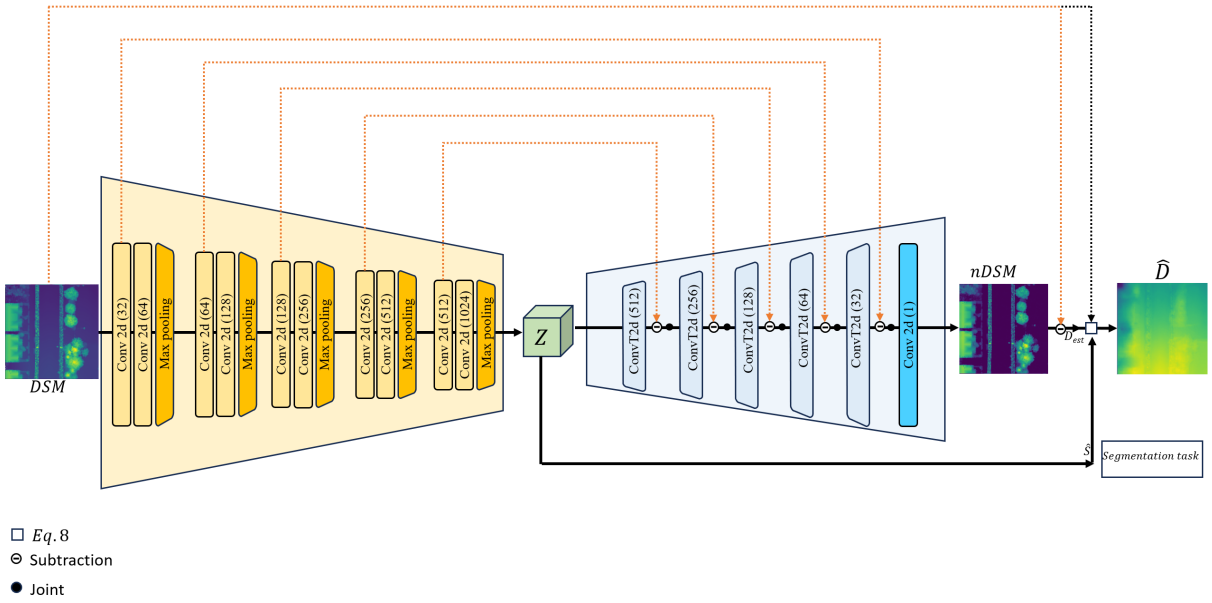


Figure 5.3: Detailed architecture of the regression task.

In the regression branch of our model, each layer employs a subtraction operation defined as:

$$S^l_{\text{sub}} = h_l - \tilde{h}_l, \tag{5.8}$$

where h_l represents the output from the encoder layers, and $\tilde{h}_l$ denotes the output from the decoder at the corresponding layers. This procedure functions as a high-pass filter,

emphasizing the high-frequency components present in the DSM. The resulting high-frequency component, S^l_{sub}, can be interpreted as the difference between DSM and DTM, which is conceptually analogous to the nDSM. This analogy arises because the decoder outputs, $\tilde{h}_l$, influenced by the regression's loss, effectively act as an intermediate representation aiming to capture the essential features of the DTM. Finally, the regressed DTM, D_{est}, is obtained by subtracting nDSM from DSM. However, it's important to clarify that D_{est} represents the model's estimation of the DTM and is distinct from the $\hat{D}$ used in the loss function. The $\hat{D}$ is calculated from equation 5.9, trying to fill gaps caused by non-ground areas.

$$\hat{D} = \text{DSM} \odot (1 - \hat{S}) + D_{\text{est}} \odot \hat{S} \tag{5.9}$$

where $\odot$ is the Hadamard product.

The loss function of this task is designed to minimize the difference between the predicted elevation values and the actual DSM values, facilitating the model's ability to accurately capture the underlying terrain elevation by adjusting the model parameters. The objective of the regression is to estimate D_{est} by optimizing the parameters θ_{enc} and θ_{reg} through the minimization of the following loss function:

$$L_{rec}(\theta_{\text{enc}}, \theta_{\text{reg}}) = \sum_{i=1}^{N} \ell(\hat{D}_i, D_i), \tag{5.10}$$

where ℓ represents a loss metric (here, mean absolute error), $\hat{D}_i$ are the estimated elevation values from the regression model, D_i are the actual target DSM values, and N signifies the number of samples.

5.3.3 Segmentation Task

The segmentation task, as illustrated in Figure 5.4, pertains to an autoencoder that shares its encoder with the regression task. Meanwhile, its decoder is fortified by the features originating from the aforementioned branch. The incorporation of nDSM features into the segmentation branch results in better outcomes in terrain analysis. These features estimate non-ground points and provide crucial insights into elevation variations and the structure

of objects above the ground. This concatenation provides the segmentation branch with essential details that help distinguish between ground and non-ground points, especially in complex environments.

The encoder-derived latent, Z, undergoes a transformation process through a decoding mechanism, employing a series of transposed convolution operations denoted as $T^{(l)}$ for the l^{th} layer. These operations upscale and refine the segmentation feature maps C_l^{con}, which are essential for recovering spatial details pivotal for precise segmentation. The transformation can be represented by Equation 5.11.

$$C_{l+1}^{\text{con}} = T^{(l)}(C_l^{\text{con}} \oplus S_l^{\text{sub}}), \tag{5.11}$$

where C_{l+1}^{con} denotes the feature map output from the subsequent decoder layer, and S_l^{sub} is the feature map from a parallel branch that integrates complementary spatial information through concatenation. The final stage of the process involves the production of the ultimate

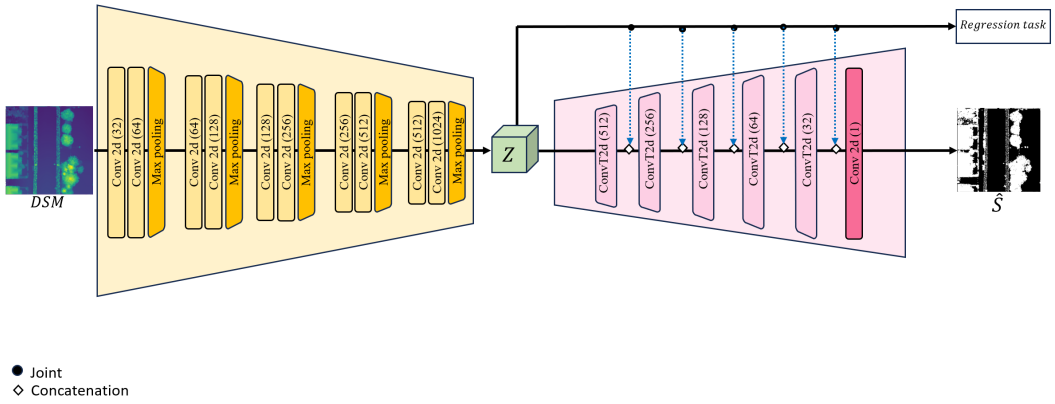


Figure 5.4: Detailed architecture of the segmentation task

segmentation output denoted as $\hat{S}$, which is obtained by applying a sigmoid activation function to the last feature map of the decoder. This is mathematically expressed as:

$$\hat{S} = \sigma(\text{Conv}(C_1^{\text{con}} \oplus S_0^{\text{sub}})), \tag{5.12}$$

where σ denotes the sigmoid function. This function ensures that the segmentation output is a binary map, representing the probability of each pixel belonging to the target feature class.

The optimization of this segmentation process is governed by a specifically designed loss function $L_{seg}(\theta_{enc}, \theta_{seg})$ (equation 5.13) that aims to minimize the discrepancy between the predicted segmentation map $\hat{S}$ and the ground truth.

$$L_{seg}(\theta_{enc}, \theta_{seg}) = -\sum_{i=1}^{N} \left[S_i \log(\hat{S}_i) + (1 - S_i) \log(1 - \hat{S}_i) \right], \tag{5.13}$$

Here, N denotes the total number of samples in the training set. This formulation not only facilitates the model's learning by encouraging adjustments in the model parameters θ_{enc} and θ_{seg} to more accurately reflect the presence or absence of specific features within the DSM but also ensures a direct alignment with the segmentation objective. :

$$h_i = f_{\text{enc}}^{i}(h_{i-1}), \tag{5.14}$$

5.3.4 Multi Task Learning

The development of the DB-SUBNET methodology is predicated on a multi-task learning paradigm, operationalized through the formulation of a composite loss function, denoted as L_{total}. This function is adeptly constructed to amalgamate the segmentation loss L_{seg}, regression loss L_{reg}, and a cross-regularization term R_1, thereby fostering a holistic and balanced optimization landscape.

The aggregate loss function, L_{total}, is articulated as a weighted amalgamation, given by:

$$L_{\text{total}} = \lambda_1 L_{\text{seg}} + \lambda_2 L_{\text{reg}} + \lambda_3 R_1, \tag{5.15}$$

where λ_1, λ_2, and λ_3 represent the weights allocated to each loss component, a configuration crucial for achieving equilibrium within the learning schema, ensuring the proportional influence of each constituent loss. The cross-regularization term R_1, aimed at mitigating overestimation in non-ground regions, thereby refining the accuracy of DTM generation, is defined as:

$$R_1 = \frac{1}{N} \sum_{i=1}^{N} \max(0, \hat{D}_i - D_i) \cdot \hat{S}_i, \tag{5.16}$$

where this term plays a pivotal role in controlling the elevation feature overestimation in non-ground regions.

In the optimization of the DB-SUBNET parameters, namely θ_{enc}, θ_{seg}, and θ_{nDSM}, a comprehensive approach is adopted that involves the synergistic consideration of all components within the composite loss function L_{total}. This strategy ensures a balanced refinement of the model parameters, aiming to minimize the overall loss by simultaneously addressing the segmentation, regression, and regularization objectives. This approach underscores the importance of a unified optimization framework where the network's learning process is intricately aligned with the multi-faceted nature of the loss components, ensuring that the parameter updates are informed by a holistic perspective of the learning objectives.

5.4 Experimental Result

5.4.1 Quality Assessment Criteria

In our study, we have adopted a comprehensive framework for quality assessment, which is divided into three principal categories: 1) Predictive error metrics (discussed in section 2.6.1), 2) Classification accuracy metrics(discussed in section 2.6.2), and 3) Spectral analysis metrics (discussed in section 2.6.3).

5.4.2 Experimental Setup

In the preparation of our data and the configuration of our model, we implemented several key steps to ensure optimal training and results. Initially, DSMs and their corresponding DTMs were divided into images with a 50% overlap, each having a size of 256×256 pixels. These images were then localized and subsequently normalized to mitigate the effects of domain shift [169]. We then incorporated horizontal and vertical flips into our data augmentation strategy to enhance the diversity and robustness of the training set. For the training, recognizing the distinct characteristics of various datasets as discussed in [169], and meanwhile our thorough

data analysis in section 4.4.1, we decided to utilize two separate datasets for training: one for urban and hilly areas, named 'SUI', and another for mountainous and predominantly vegetation-covered areas, named 'KA'. The model training parameters were meticulously set, starting with a learning rate of 10E-2, gradually reducing to 10E-5 every 70 iterations throughout 250 epochs. Finally, during inference, the weights derived from the SUI dataset were applied to SUII, RT, DALES, NB, and S4; and the weights from the KAI dataset were specifically applied to KAII, KW, S8, and S9. It is important to note that for both SU and KA, we maintain two sets: one is used during training, and the other, labeled with 'II', is employed during the test. This setup enables us to conduct both regular and blind testing in our analysis.

5.4.3 Competing Methods

Competing methods have been selected from two primary categories in this domain, regression-based and classification-based. As far as regression-based methods are concerned, three state-of-the-art methods have been considered, which are HDCNN, pix-to-pix [208], used by [175, 174] for DTM generation, and SB-SUBNET, which is a simple version of the proposed method. The HDCNN framework merges the U-Net architecture with residual networks. This integration enhances feature representation, leading to more accurate predictive outcomes. Secondly, the pix-to-pix model showcases its effectiveness in converting DSMs into DTMs. It utilizes conditional adversarial networks, ensuring high fidelity in the transformation process. Thirdly, SB-SUBNET, a variant of our proposed method, acts as a one-branch version of DB-SUBNET. It plays a crucial role in demonstrating the efficiency of subtractive skip connections and highlights the importance of segmentation addition in DB-SUBNET.

In contrast, segmentation-based approaches aim to classify DSMs into ground and non-ground areas. Among these methods, RandLA, as detailed in [181], has been successfully employed for this task. It utilizes random point sampling, a straightforward and efficient method, which bypasses the complexities associated with advanced point selection techniques [209]. EyeNet is also selected because it outperforms RandLA [173]. This method is a novel

semantic segmentation network for point clouds inspired by human peripheral vision. It addresses the issue of functional coverage area size of inputs by introducing a multi-scale input and a parallel processing network with connection blocks. EyeNet overcomes the limitations of conventional networks and achieves state-of-the-art performance on the SensatUrban and Toronto3D datasets.

5.4.4 Results and Discussion

This section presents a comprehensive analysis of our proposed model, incorporating ablation studies aimed at investigating the impact of each component of the proposed method, Intra-domain generalization to evaluate the model's performance on new instances within the same domain, and lastly, inter-domain validation, to assess the model's adaptability across various domains. These evaluations, summarized in Table 5.1, highlight the approach's robustness and versatility across a range of experimental settings.

Table 5.1: Summary of intra-domain experiment, where the ratio of training to testing is 80 to 20.

Analysis Type	**Training**	**Testing**	**Objectives**
Ablation Studies	SUI	SUI	Ablation studies with DB-SUBNET's variation
	KW	KW	
	LMD (Sec: 4.4.2)	LMD	
Intra-domain Validation	SUI	SUII	Evaluation over intra-domain test datasets
	KA	KAII	
Inter-domain Validation	SUI	SUIII	Evaluation over inter-domain test datasets
	SUI	NB	
	SUI	DALES	
	SUI	RT	
	SUI	S4	
	KAI	KW	
	KAI	S8	
	KAI	S9	

Ablation Studies

To evaluate the effectiveness of DB-SUBNET, two other versions are taken into account. SB-SUBNET, which was discussed in the previous chapter, is being considered to demonstrate

the significance of multi-task learning and segmentation. On the other hand, DB-SUBNET Unconstrained is being assessed to exhibit the crucial role of the regularization term between the two regression and segmentation branches. The main goal is determining which method can effectively minimize errors in identifying ground as non-ground. This is a vital aspect of our application because we want to keep accurate ground points that laser scanners have precisely gathered.

The analysis of the mountainous KW dataset, as tabulated in Table 5.2, reveals that DB-SUBNET outperforms the other models in terms of accuracy metrics. Specifically, it exhibits the lowest RMSE and MAE values, indicating its proficiency in handling complex terrains covered by dense trees. Additionally, DB-SUBNET demonstrates the highest IOU ground, highlighting its effectiveness in ground classification amidst the challenging topography of the KW dataset. Conversely, DB-SUBNET Unconstrained shows a slightly better Type I error, but it compromises other accuracy metrics, which underscores the importance of regularization in enhancing model performance. Although SB-SUBNET shows reasonable performance in some areas, it falls short in critical metrics like Type I error and IOU ground, indicating a less reliable ground classification capability in such rugged terrains, which means it changes ground points.

Transitioning to the urban landscapes of the SUII dataset, the report remains consistent with DB-SUBNET outperforming in key metrics, reinforcing its robustness across diverse environments. It achieves the lowest RMSE and MAE, along with the highest IOU ground, which is crucial for urban applications where precise ground delineation can significantly impact subsequent analyses or decision-making processes. The performance of DB-SUBNET Unconstrained and SB-SUBNET, while commendable in specific aspects, does not match the balanced accuracy and reliability exhibited by DB-SUBNET.

In the LMD data scenario, which blends different terrain characteristics together, DB-SUBNET once again emerges as the top performer, showcasing its adaptability and consistent precision across various environments. It maintains the lowest RMSE, MAE, and the highest IOU for ground classification, establishing its overall superiority in distinguishing ground

from non-ground features. Although other methods show promise in specific metrics, they do not offer the comprehensive performance DB-SUBNET provides.

Table 5.2: Comparative analysis of different elements of the proposed method.

Metric	**Dataset**	**DB-SUBNET**	**DB-SUBNET Unconstrained**	**SB-SUBNET**
RMSE [m]	KW	**0.835**	1.354	1.110
	SUII	**0.117**	0.336	0.198
	LMD	**0.388**	0.680	0.477
MAE [m]	KW	**0.511**	0.949	0.720
	SUII	**0.060**	0.174	0.105
	LMD	**0.224**	0.428	0.293
IOU ground	KW	**0.7253**	0.7278	0.4927
	SUII	**0.9377**	0.9281	0.8312
	LMD	**0.8479**	0.8382	0.6618
IOU nonground	KW	0.6600	**0.6564**	0.6191
	SUII	**0.8103**	0.7843	0.6542
	LMD	**0.7809**	0.7633	0.6758
Type I Error	KW	0.0219	**0.0200**	0.1151
	SUII	**0.0159**	0.0172	0.1100
	LMD	0.0228	**0.0211**	0.1191
Type II Error	KW	0.0304	0.0320	**0.0068**
	SUII	0.0306	0.0368	**0.0123**
	LMD	0.0346	0.0428	**0.0160**

Intra-Domain Validation

In this section, we evaluate the efficacy of the proposed methods in comparison to competitors, focusing on scenarios where the domain of the test and that of the terrain are identical. The average results of the proposed methods, along with those of their competitors, are presented in Table 5.3, which pertains to the mountainous area predominantly covered with various

types of flora, including low and high vegetation, and Table 5.4, which is associated with the urban area. These tables demonstrate that the two proposed methods surpass current state-of-the-art techniques. To elaborate, the mean results and standard deviations of the proposed methods, as well as those of competitors pix-to-pix and HDCNN (which have similar inputs to our methods), are illustrated in Figure 5.5. This figure aims to demonstrate the stability of these methods. To this end, a hypothetical normal distribution is designed, centered around the mean of the results and extending to three times their standard deviation. The figure clearly demonstrates that the proposed methods not only yield superior results but also maintain consistent performance across different depth images during testing. This consistency is evidenced by their minimal standard deviations, as reflected in the narrow probability distributions of the methods.

Table 5.3: Evaluation of the proposed methods and their rivals on KAII.

Method/QAM	**RSME [m]**	**MAE [m]**	**Accuracy**	**Type1 Error**	**Type2 Error**	**IOU**
SB-SUBNET	0.976	0.675	0.354	0.989	0	0.298
DB-SUBNET	0.861	0.586	0.831	0.105	0.417	0.428
pix-to-pix	5.09	4.13	-	-	-	-
HDCNN	1.26	0.974	0.355	0.986	0	0.299
EyeNet	3.34	0.612	0.812	0.254	0.0769	0.647
RandLA	3.25	0.577	0.797	0.269	0.091	0.626

Table 5.4: Evaluation of the proposed methods and their rivals on SUII.

Method/QAM	**RSME [m]**	**MAE [m]**	**Accuracy**	**Type1 Error**	**Type2 Error**	**IOU**
SB-SUBNET	0.12	0.0495	0.853	0.57	0.00233	NA
DB-SUBNET	0.12	0.058	0.960	0.0819	0.026	0.889
pix-to-pix	0.523	0.289	-	-	-	-
HDCNN	0.22	0.162	0.853	0.575	0.00227	NA
EyeNet	3.64	1.29	0.934	0.065	0.0734	0.703
RandLA	3.34	1.05	0.944	0.052	0.077	0.736

In comparing the two proposed methods, DB-SUBNET performs better on the KAII dataset, as confirmed by a t-test at the 95 percent confidence level. Regarding SUII, both methods exhibit similar performance.

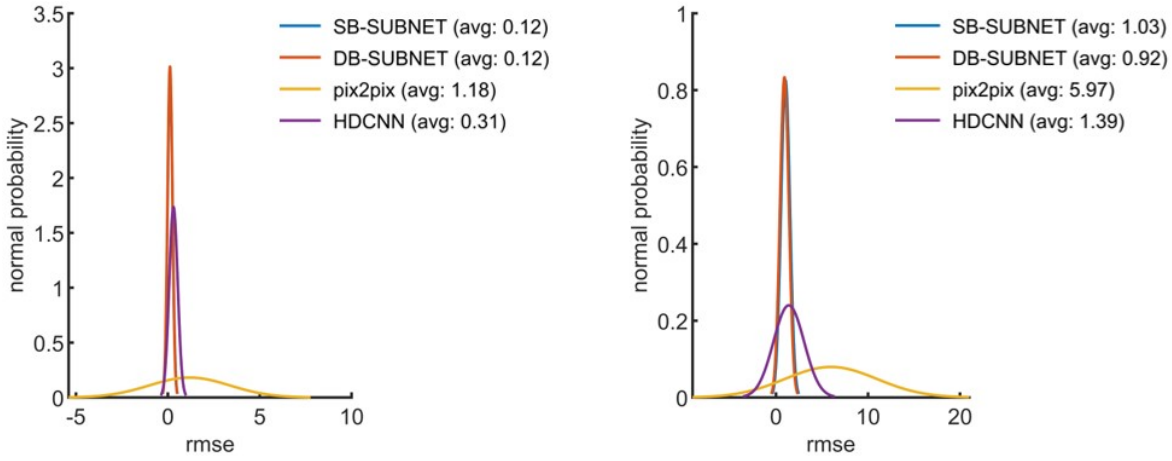


Figure 5.5: Methods' stability: left - SUII, right - KAII.

The efficacy of the proposed methods is apparent in the visual representations of two samples from the KAII and SUII datasets, as depicted in Figures 5.6 and 5.8 respectively. The corresponding cross-sections for each sample can be observed in Figures 5.7 and 5.9. These graphical representations demonstrate the effectiveness of the methods employed in the analysis. These figures demonstrate that the correlation between the generated depth and the ground truth is absolutely high. Additionally, the EMD for these methods is smaller than that of pix-to-pix and HDCNN. Evaluation of cross sections in these figures demonstrates that SB-SUBNET and DB-SUBNET are more closely aligned with the ground truth, in contrast to their competitors, particularly pix-to-pix, which shows a greater deviation.

In the comparison of competitors, as tabulated in Tables 5.3 and 5.4, HDCNN outperforms the others. This conclusion is supported by correlations and histograms for both areas in question. The depths generated by HDCNN are highly correlated with the ground truth, a finding confirmed by the EMD criterion.

Regarding pix-to-pix, the data presented in the tables indicate its weaker performance, while its cross-section analysis seems to suggest some efficacy. To understand this discrepancy between the tabulated results and cross-section observations, Figure 5.8 is insightful. It reveals noisy areas in the lower right part of the generated DTM by pix-to-pix. This specific section leads to pix-to-pix's overall weak performance; however, its performance appears satisfactory

in other sections, like the one examined in the cross-section. Our experimental results reveal that small heterogeneous areas on the corners are frequently observed in several depth maps generated by pix-to-pix. This localized inconsistency significantly affects the overall evaluation of its performance, highlighting its poor performance in terms of quantitative criteria.

As far as the regression-based methods are concerned, this point should be first considered that SB-SUBNET, HDCNN, and pix-to-pix do not inherently produce any classified depth maps. However, for HDCNN and SB-SUBNET, we have generated a classification product as detailed in [169]. This process was not applied to pix-to-pix because no classification map has been produced in the referenced studies [174] and [175], where it is utilized for DTM generation.

From the classification perspective, as expected, RandLA and EyeNet significantly outperform SB-SUBNET, pix-to-pix, and HDCNN. However, DB-SUBNET surpasses these two methods in terms of all classification criteria. Regarding the criteria associated with depth/ DTMs, it is important to note that the performance quality of these criteria is not directly attributable to the RandLA and EyeNet but rather to the interpolation used subsequently. In this context, these methods exhibit poorer performance compared to the proposed methods and HDCNN.

The rationale for discussing state-of-the-art classification methods in this paper is twofold. Firstly, it aims to emphasize the superior performance of DB-SUBNET compared to segmentation-based methods. Secondly, it highlights that the effectiveness of segmentation-based methods, such as the traditional CSF used in software like LASTools and CloudCompare and the more advanced RandLA, is significantly influenced by the quality of interpolation. This dependence is particularly noticeable in areas with large voids or insufficient surrounding data. The results presented in Tables 5.3 and 5.4 support this notion. Although RandLA and EyeNet outperform HDCNN in terms of classification, they are inferior to HDCNN in the context of DTM extraction.

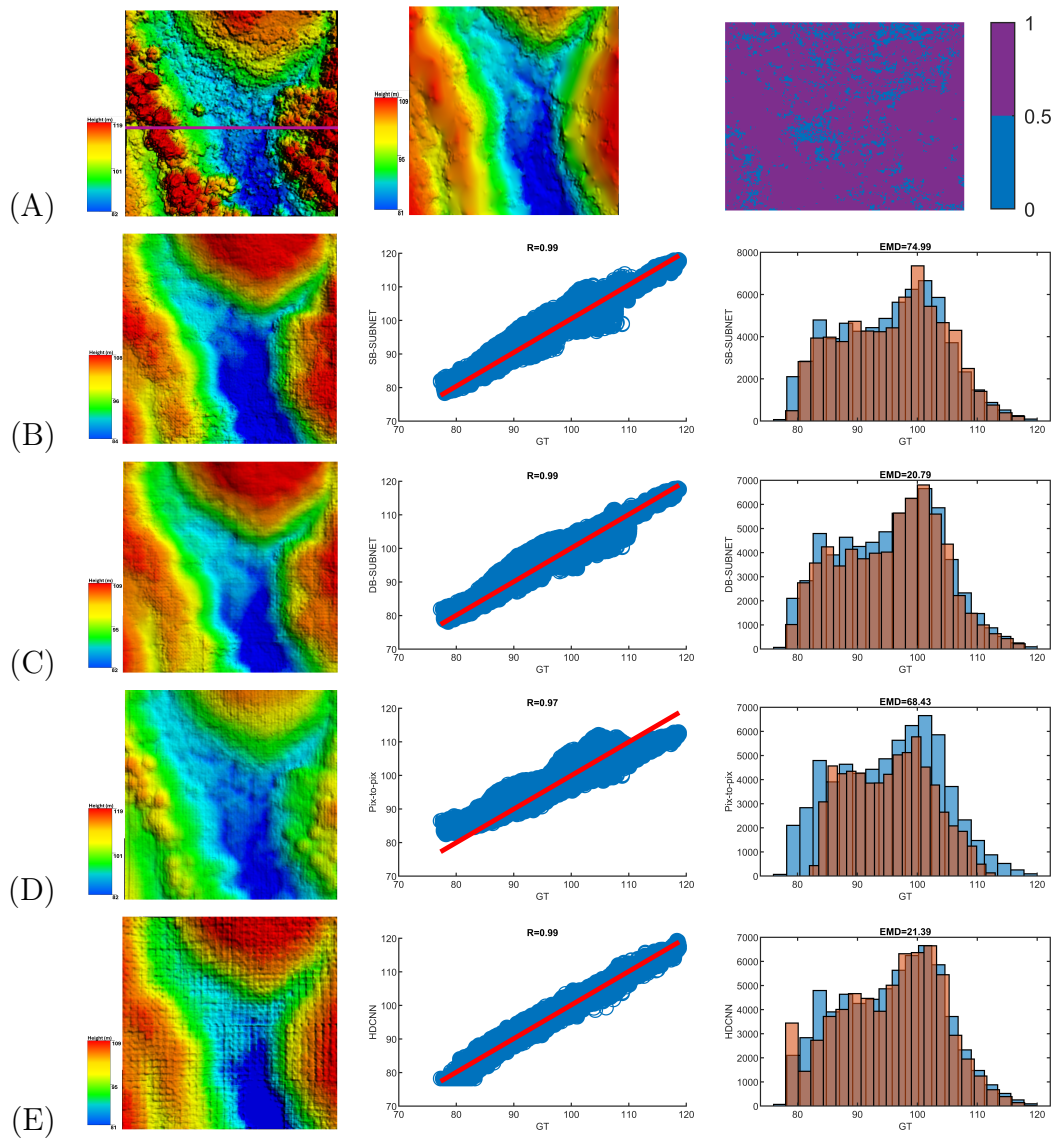


Figure 5.6: DTM generation results for KAII: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

Inter-Domain Validation

In this part, the efficacy of the proposed methods is assessed in unseen areas, DALES, S4, NB, RT, S8, S9, and KW, where the domain of the test differs from the training one.

In this comparative analysis, regression-based methods are first compared with DB-SUBNET. As illustrated by the results presented in Figures 5.10 through 5.23 and tabulated

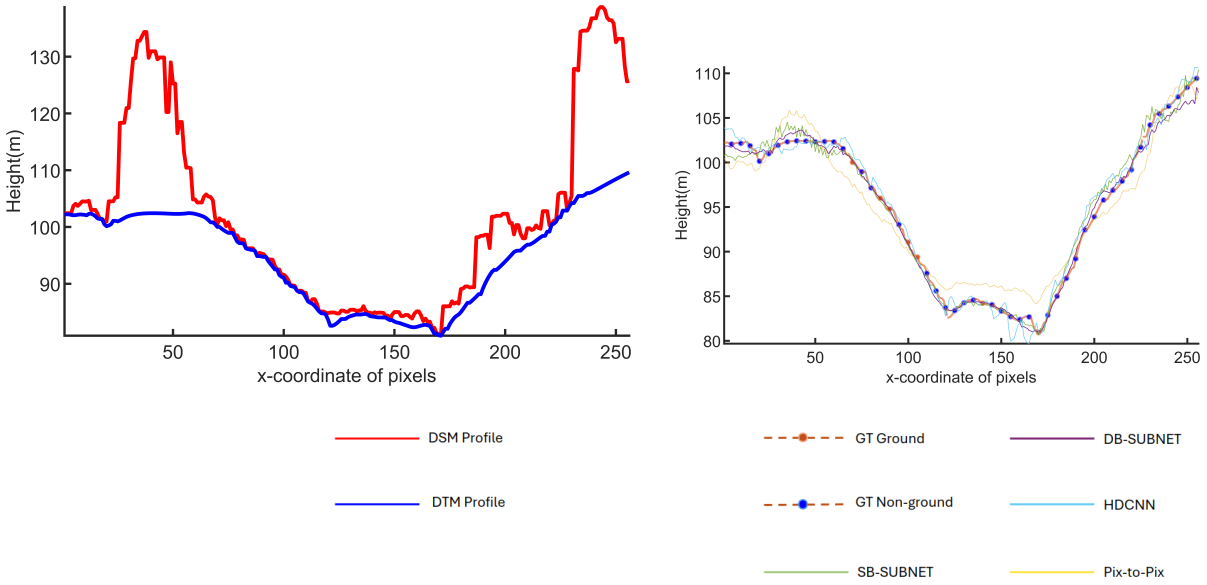


Figure 5.7: Cross-sections of each method for a KAII sample.

in Tables 5.5 through 5.10, the proposed method clearly emerges as the leading candidate, displaying remarkable alignment with ground truth representations of diverse terrains. SB-SUBNET, while showing promise, assumes the position of the second-best performer, which is discussed the reason of which in the following. While SB-SUBNET generally follows the ground truth trend, as illustrated in Figures 5.10, 5.11 and 5.12, 5.14, it does not perform as effectively as DB-SUBNET in non-terrain areas. This performance gap can be attributed to SB-SUBNET's deficiency in segment-specific knowledge. However, An in-depth analysis of Figures (5.13,5.15), (5.16,5.19), and (5.18, 5.21) demonstrates that SB-SUBNET is competitive with DB-SUBNET in mountainous areas, especially in landscapes devoid of buildings. Shifting the focus to the pix-to-pix method, it becomes evident that this approach tends to overstate terrain features, as observed in cross-sectional views. This trait is substantiated by the lower correlation values and higher EMD scores associated with pix-to-pix, indicating an inclination towards topographical exaggeration. In contrast, HDCNN exhibits a more consistent performance across various datasets. Meanwhile, HDCNN maintains a relatively balanced performance across the datasets, although slight deviations suggest room for refinement.

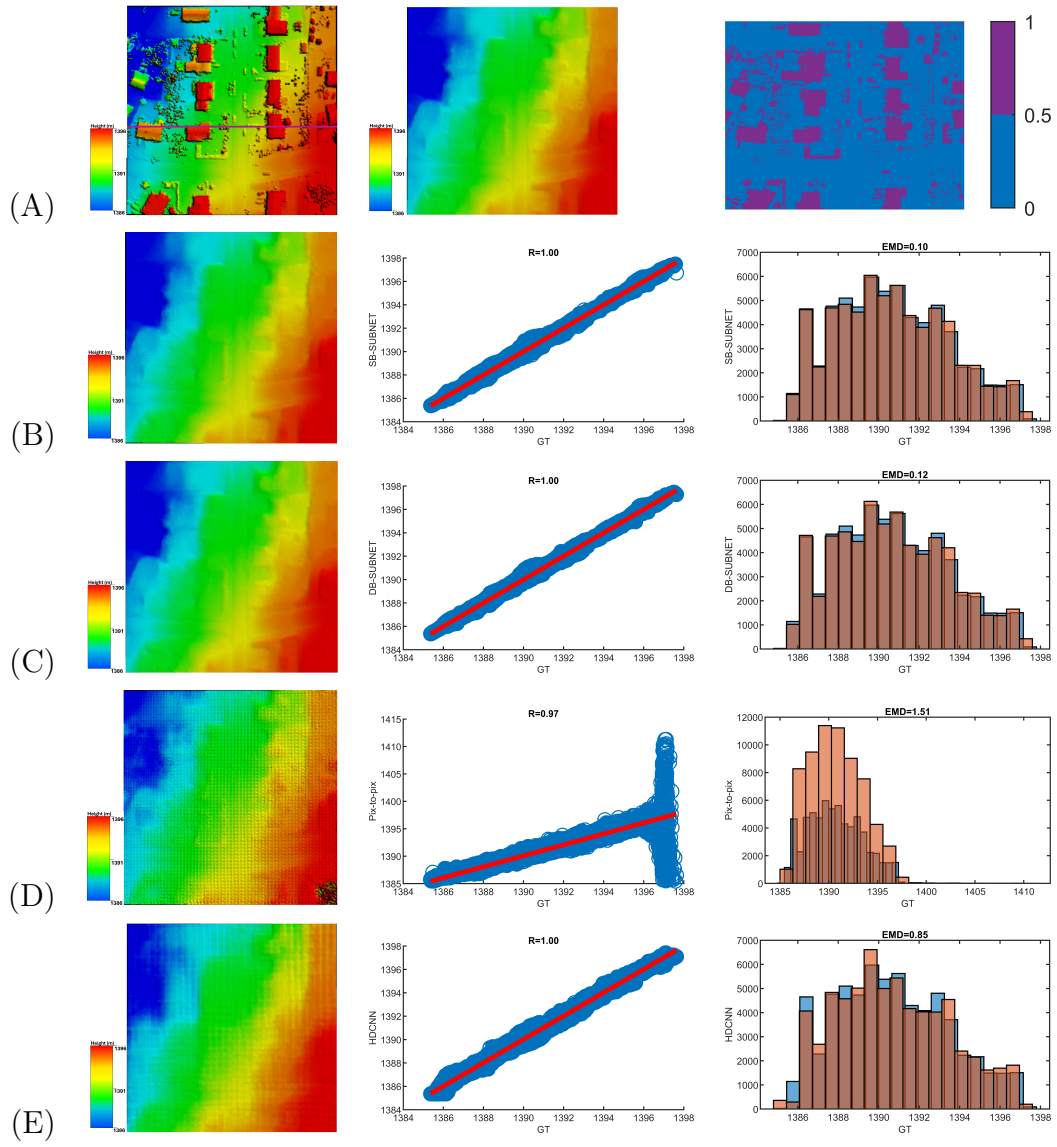


Figure 5.8: DTM generation results for SUII: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

The accurate rendering of flat terrains is of paramount importance, particularly in urban datasets such as RT and NB, where precise mapping is crucial. DB-SUBNET's consistency is underscored by its uniformly low EMD values, reinforcing the cross-sectional observations of its ability to capture terrain flatness accurately (seen in Figures (5.12,5.14), (5.17, 5.20) and (5.22,5.23) clearly). SB-SUBNET's minor systematic elevation offset points to a less precise

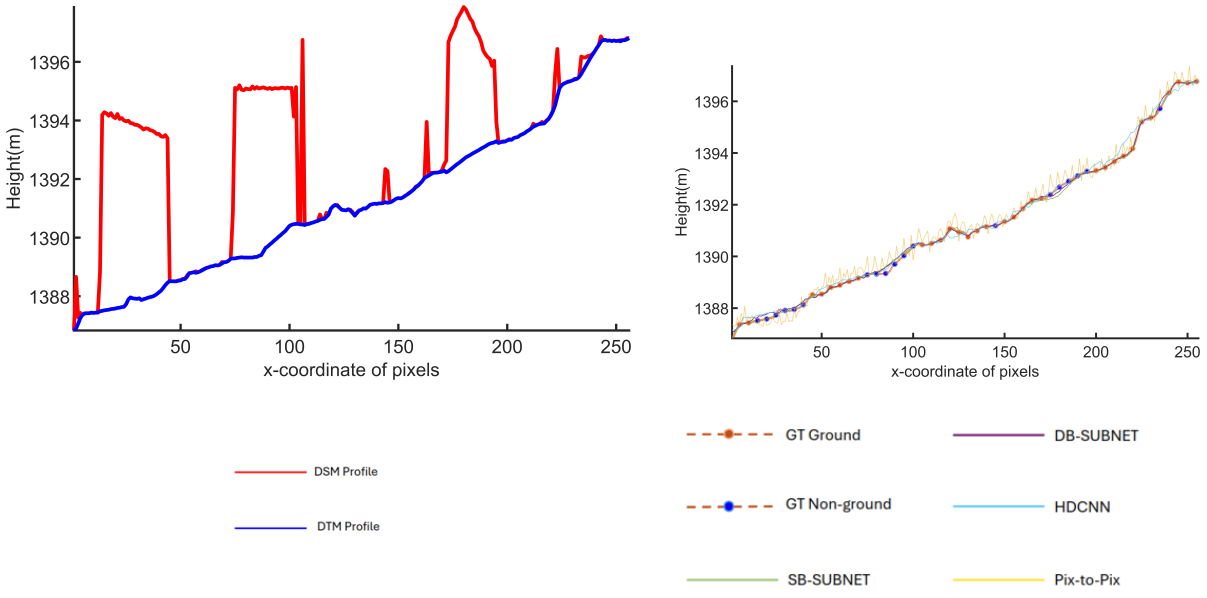


Figure 5.9: Cross-sections oF each method for an SUII sample.

representation of flat terrain, as corroborated by both cross-sectional and global evaluations. Despite the better performance of HDCNN over pix-to-pix, both exhibit variability in depicting flat terrains.

In the context of vegetation-covered areas, as is evident from Figures 5.10 through 5.23, and particularly highlighted in Figures (5.12, 5.14), DB-SUBNET's performance stands out significantly. This outstanding performance is characterized by the presence of well-aligned spikes in cross-sections. These distinctive characteristics serve to emphasize DB-SUBNET's robust capability for the accurate recognition and efficient removal of vegetation. This is also evident in densely vegetated datasets such as S8 and S9 (Figures (5.13,5.15) and (5.16,5.19), respectively). In SB-SUBNET, the combination of lower cross-correlation and higher EMD scores, as shown in the DALES dataset (Figures (5.10,5.11)), suggests an underestimation of vegetation coverage. In contrast, pix-to-pix exhibits pronounced spikes in cross-sections alongside higher EMD scores. This pattern, detailed respectively in Figures (5.17,5.20) and (5.18,5.21), implies a fluctuating overestimation or underestimation of vegetation density.

In the critical task of representing building areas, DB-SUBNET's accuracy is paramount,

delivering distinct flat-topped profiles that closely match the ground truth, with this consistency verified by both detailed cross-sectional and broader scene evaluations. The method's low EMD scores emphasize its precision, which is essential for urban datasets with pronounced man-made structures. As observed in cross-sections, SB-SUBNET's less defined building profiles result in higher EMD values, underscoring areas for improvement in urban structure representation. The pix-to-pix's tendency to exaggerate building features is reflected in both cross-sectional analysis and EMD evaluations, raising concerns about misclassification.

DB-SUBNET demonstrates remarkable proficiency in differentiating between buildings and vegetation, adeptly capturing the intricate relationship between natural and built environments. This capability is confirmed by its low EMD scores and high correlation in mixed urban-vegetation landscapes. Such performance is notably evident in the S9 and NB datasets, as illustrated in Figures (5.16,5.19) and (5.17, 5.20). In contrast, SB-SUBNET exhibits less defined transitions from ground truth, which might pose challenges in accurately analyzing vegetated urban terrains. This aspect is particularly crucial in terrains like those represented in the S9 dataset (Figures (5.16,5.19)). Meanwhile, pix-to-pix and HDCNN show a tendency for notable classification errors in scenarios featuring complex terrain features. This is especially relevant in the KW dataset, where the accurate representation of dense vegetation is critical (Figures (5.18,5.21)). These observations highlight the nuanced performance of each method across varied landscapes, underscoring the importance of selecting the appropriate model based on specific terrain characteristics and analysis requirements.

A comprehensive evaluation of DTM estimation methods, integrating both cross-sectional and scene-wide data, places DB-SUBNET at the forefront for its consistent and accurate landscape modeling across both urban and mountainous terrains. Its robustness in mapping terrain features, vegetation density, and built structures aligns closely with the ground truth, ensuring fidelity in the final terrain representation. SB-SUBNET may be favored for broader trend analyses where fine detail is not critical. pix-to-pix requires careful calibration for precise mapping applications, while HDCNN, with its balance of strengths and weaknesses, indicates the potential for targeted improvements to enhance accuracy.

Table 5.5: Evaluation of SB-SUBNET over seven unseen data.

Data/QAM	RSME [m]	MAE [m]	Accuracy	Type1 Error	Type2 Error	IOU
DALES	0.355	0.179	0.822	0.412	0.00432	0.737
KW	0.637	0.344	0.742	1	0	0.741
RT	0.128	0.0744	0.755	0.526	0.000251	-
S4	0.147	0.0885	0.868	0.274	0.00259	0.788
S8	2.36	1.2	0.53	0.995	0	0.525
S9	3.94	3.43	0.012	0.997	0	0.008
NB	2.53	1.59	0.37	0.846	4.73E-05	0.143

Table 5.6: Evaluation of DB-SUBNET over seven unseen data.

Data/QAM	RSME [m]	MAE [m]	Accuracy	Type1 Error	Type2 Error	IOU
DALES	0.286	0.141	0.971	0.0428	0.0187	0.945
KW	0.523	0.256	0.872	0.302	0.0688	0.822
RT	0.14	0.0771	0.942	0.0664	0.0412	-
S4	0.138	0.0837	0.984	0.00977	0.0222	0.969
S8	2.21	1.08	0.893	0.109	0.0721	0.81
S9	3.83	3.24	0.941	0.0562	0.451	0.0604
NB	2.34	1.59	0.832	0.208	0.0472	0.35

Table 5.7: Evaluation of pix-to-pix over seven unseen data.

Data/QAM	RSME [m]	MAE [m]	Accuracy	Type1 Error	Type2 Error	IOU
DALES	1.08	0.679	-	-	-	-
KW	7.79	6.21	-	-	-	-
RT	0.35	0.227	-	-	-	-
S4	0.535	0.369	-	-	-	-
S8	6.94	5.52	-	-	-	-
S9	6.51	4.99	-	-	-	-
NB	3.12	2.26	-	-	-	-

Table 5.8: Evaluation of HDCNN over seven unseen data.

Data/QAM	RSME [m]	MAE [m]	Accuracy	Type1 Error	Type2 Error	IOU
DALES	0.363	0.255	0.824	0.409	0.00412	0.739
KW	1.42	1.08	0.745	1	0	0.742
RT	0.176	0.125	0.747	0.532	0.000213	-
S4	0.228	0.172	0.867	0.276	0.00255	0.789
S8	2.37	1.56	0.531	0.993	0	0.525
S9	3.62	2.67	0.0113	0.997	0	0.00804
NB	2.94	2.01	0.339	0.851	4.12E-05	0.14

Table 5.9: Evaluation of EyeNET over seven unseen data.

Data/QAM	RSME [m]	MAE [m]	Accuracy	Type1 Error	Type2 Error	IOU
DALES	1.140	0.359	37.72	0.658	0.006	0.179
S4	0.890	0.196	0.962	0.043	0.033	0.926
S8	4.783	1.704	0.947	0.009	0.918	0.069
S9	6.35	3.036	0.999	0.000	0.978	0.022
NB	1.924	0.667	0.884	0.103	0.184	0.6335

Table 5.10: Evaluation of RandLA over seven unseen data.

Data/QAM	RSME [m]	MAE [m]	Accuracy	Type1 Error	Type2 Error	IOU
DALES	5.435	3.6237	0.827	0.27	0.017	0.701
S4	0.652	0.329	0.728	0.019	0.538	0.452
S8	4.285	1.220	0.946	0.013	0.890	0.088
S9	4.539	1.681	0.999	0.000	0.967	0.028
NB	2.473	0.887	0.854	0.074	0.363	0.525

As shown in Figure 5.24, segmentation-based methods are moderately successful in identifying non-ground objects, but their effectiveness in terrain estimation is less effective. This limitation is due to the use of interpolation methods, which are widely used in this field as referenced in [169, 181]. The success of these methods depends on several factors, including the size and location of void areas. A significant problem with these methods is their failure to accurately fill voids that are not surrounded by ground points, as this requires extrapolation. This issue is clearly seen in RandLA's performance in the first cross-section

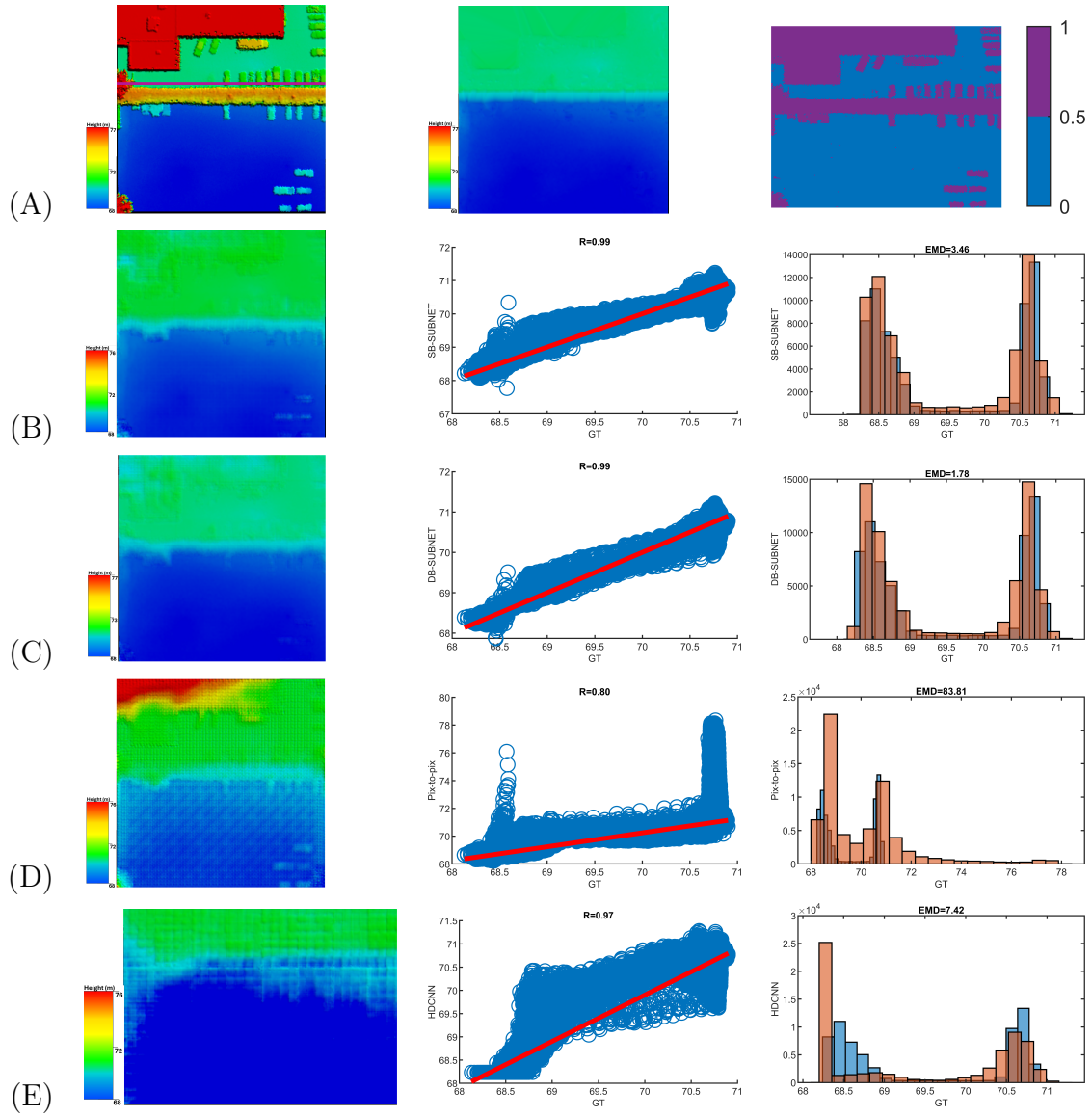


Figure 5.10: DTM generation results for DALES: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

from the top of the image in Figure 5.24, where regression-based methods do not properly address gaps in the data. The striped areas in the cross sections demonstrate this inadequacy, particularly in regions where the voids are too large to be effectively filled by interpolation. This problem is not unique to RandLA but is a common issue in commercial software like LAStools, which typically identify voids or non-ground points and then try to fill them using

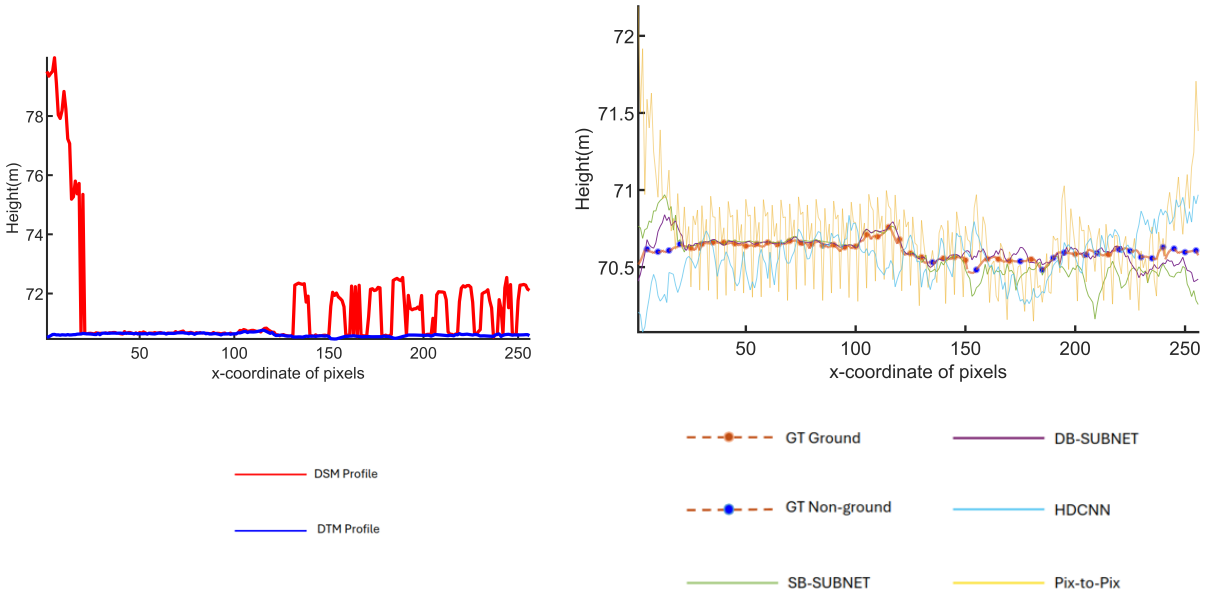


Figure 5.11: Cross-sections of each method for a DALES sample.

interpolation methods. In contrast, as can be seen from the results, the proposed method is effective in filling gaps without changing the terrain area, indicating its potential for more accurate terrain estimation.

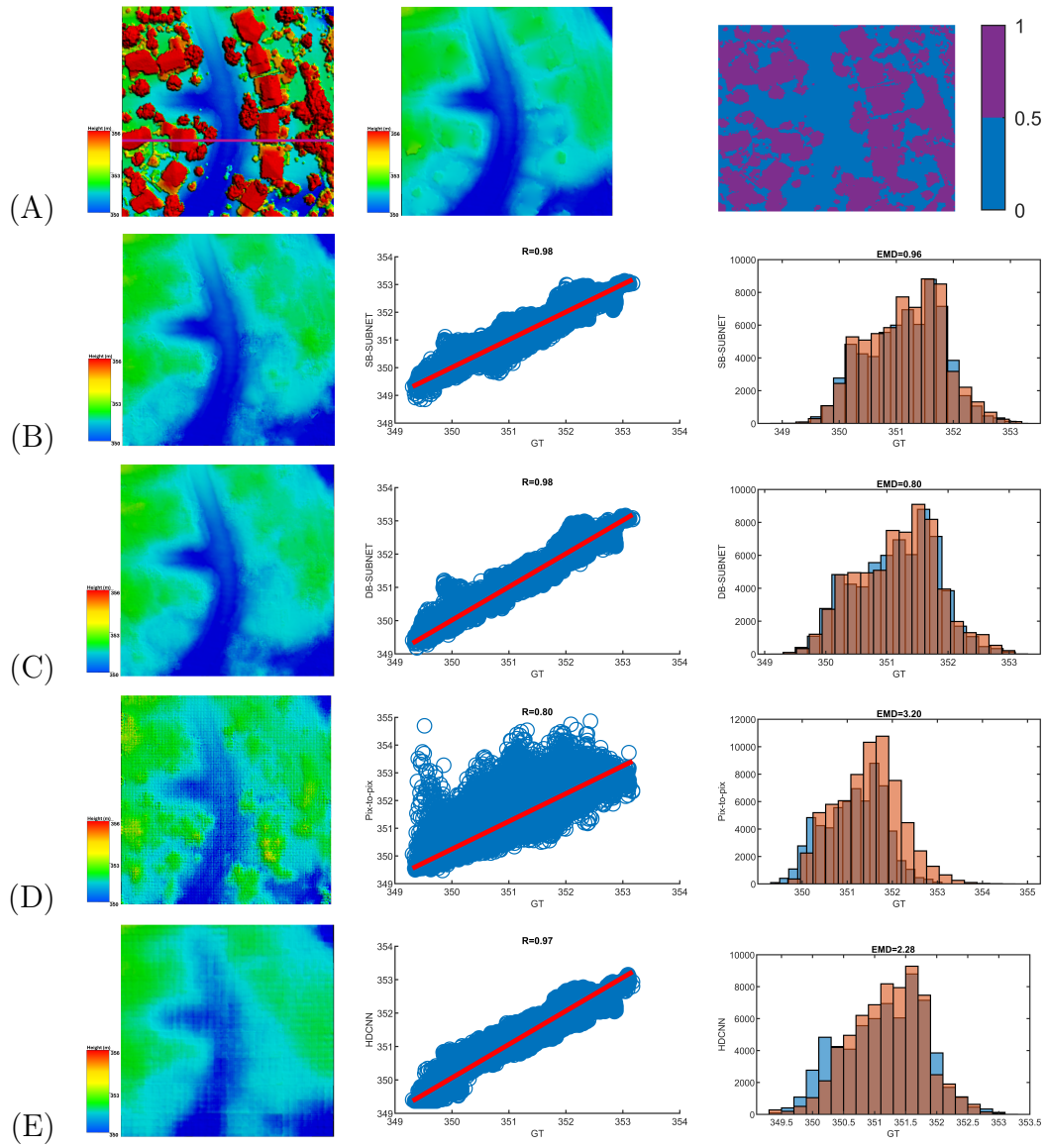


Figure 5.12: DTM generation results for S4: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

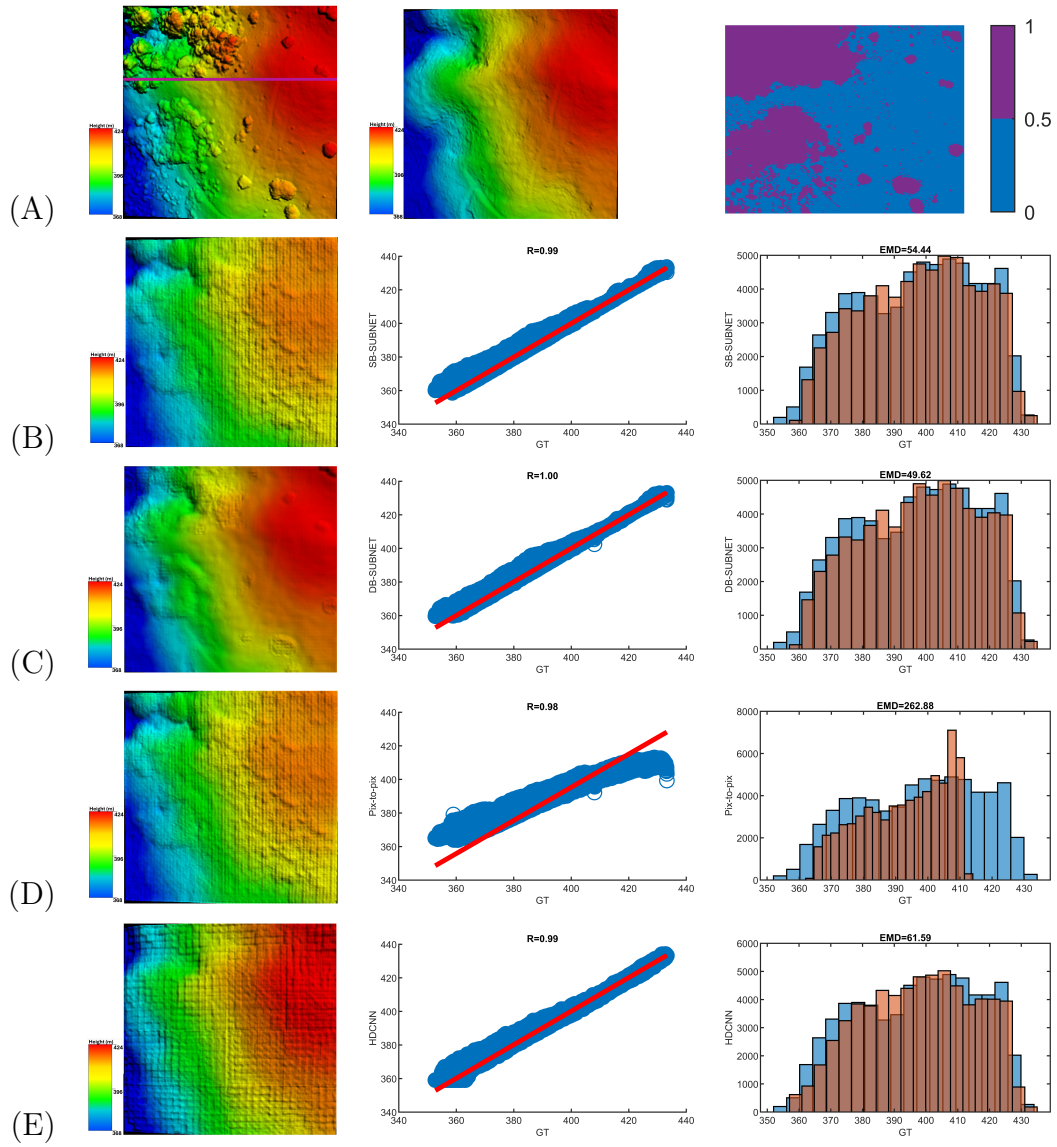


Figure 5.13: DTM generation results for S8: (A) Input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

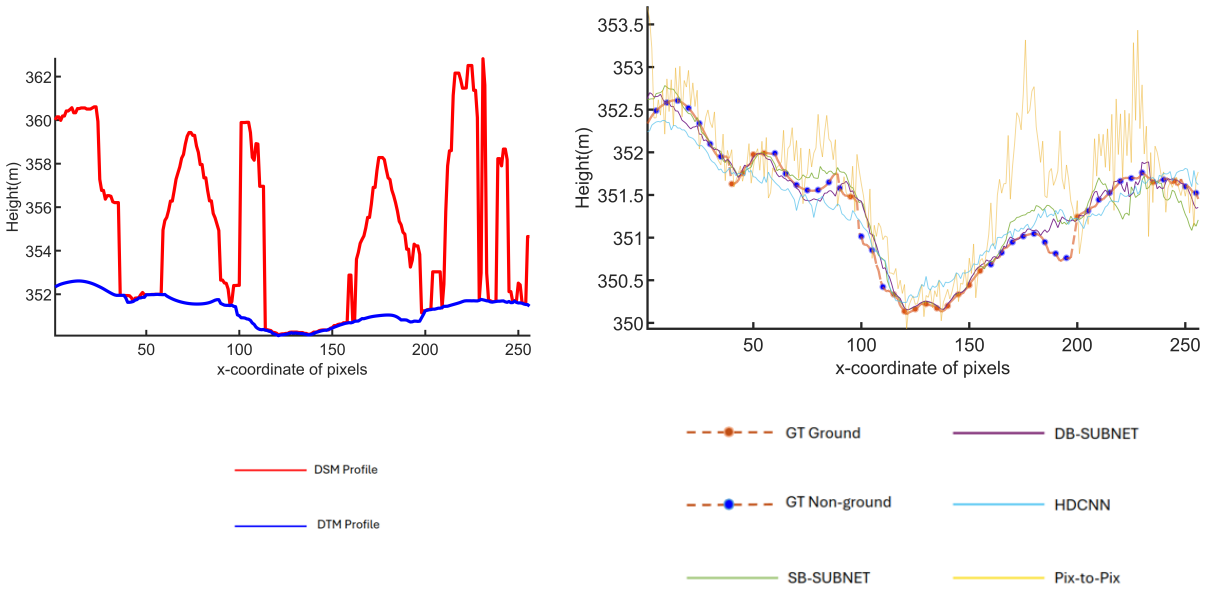


Figure 5.14: Cross-sections of each method for a S4 sample.

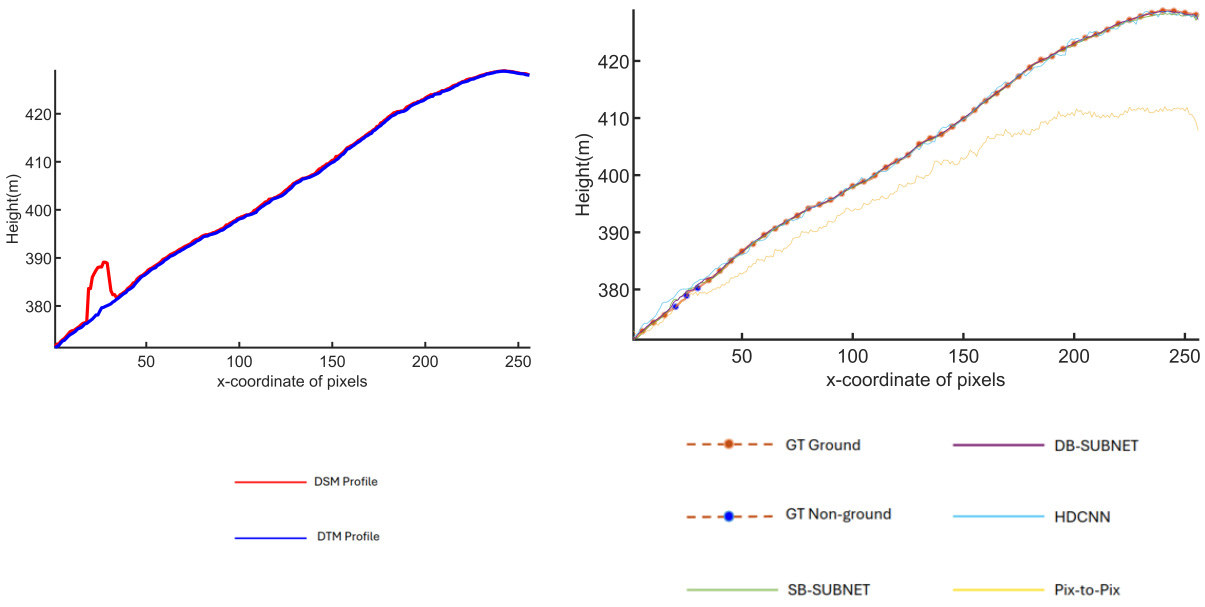


Figure 5.15: Cross-sections of each method for a S8 sample.

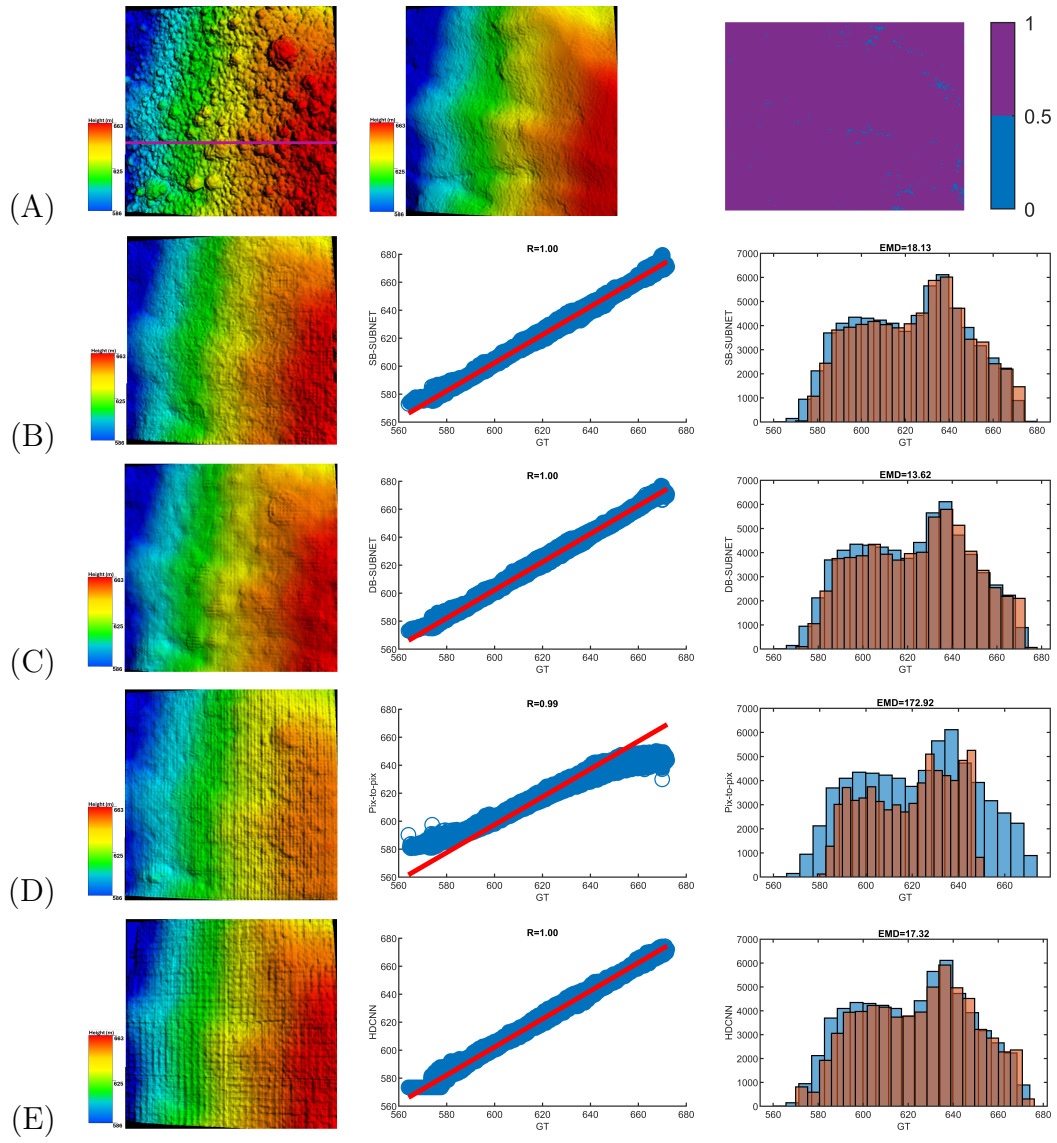


Figure 5.16: DTM generation results for S9: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

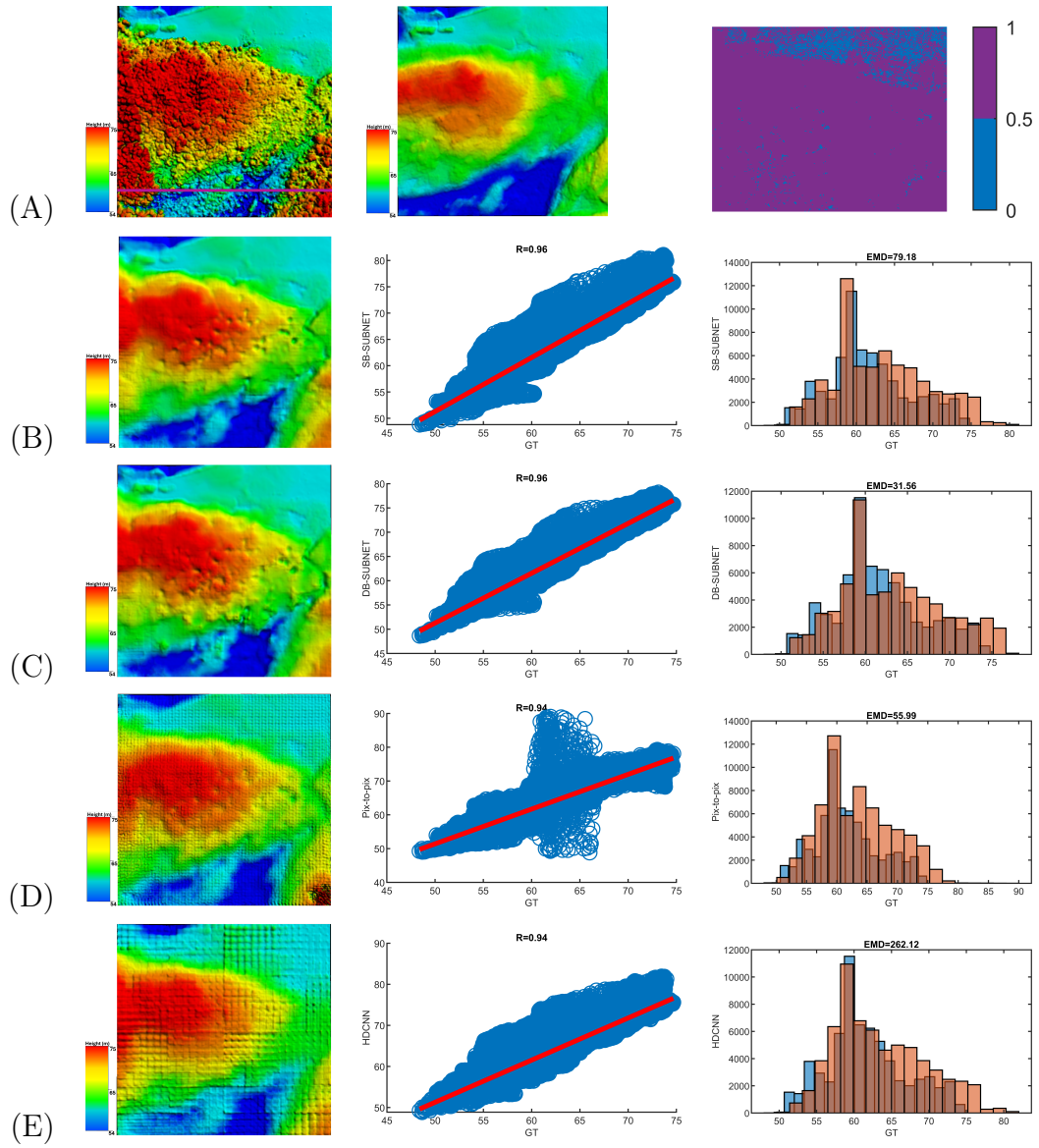


Figure 5.17: DTM generation results for NB: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

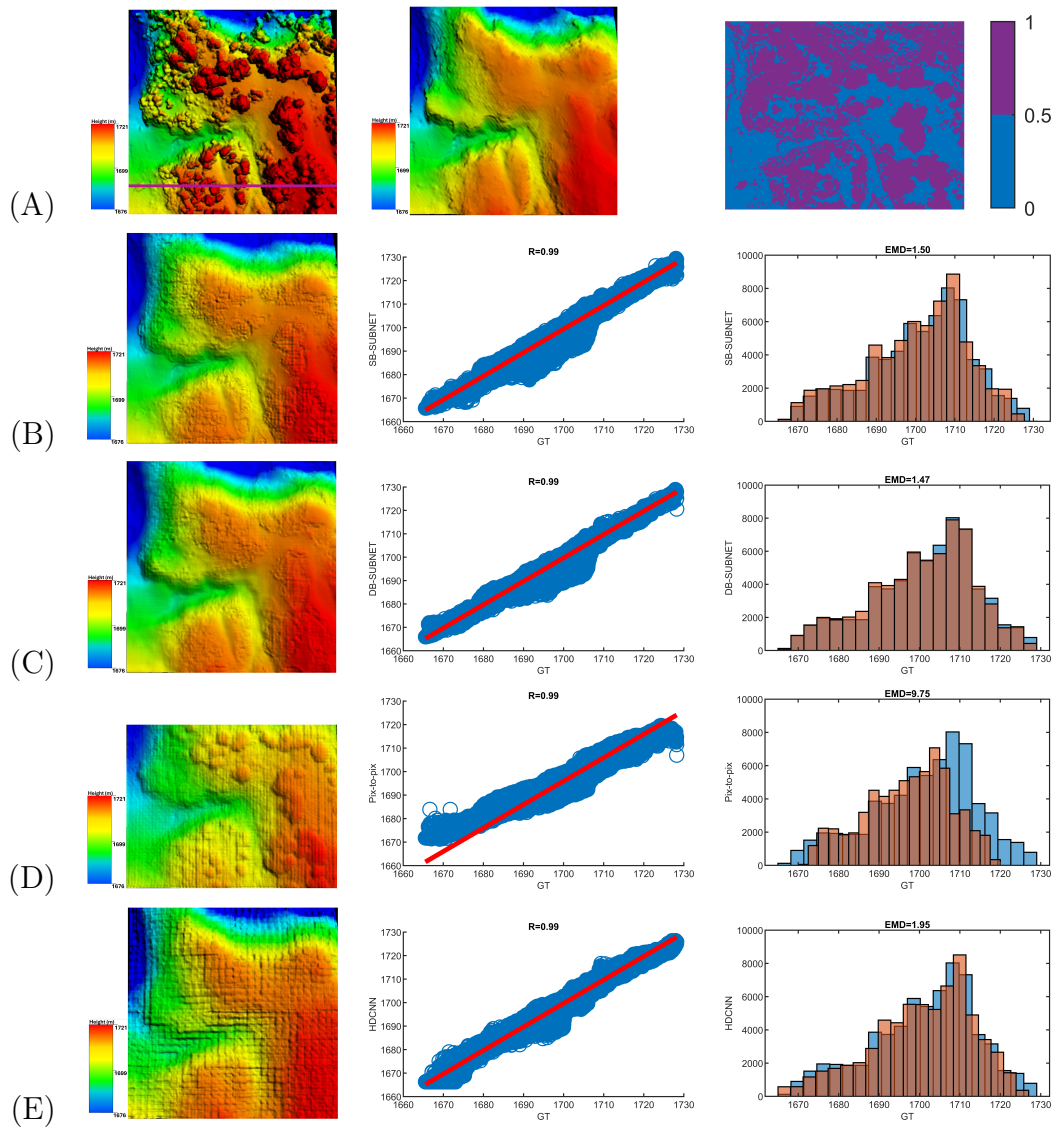


Figure 5.18: DTM generation results for KW: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

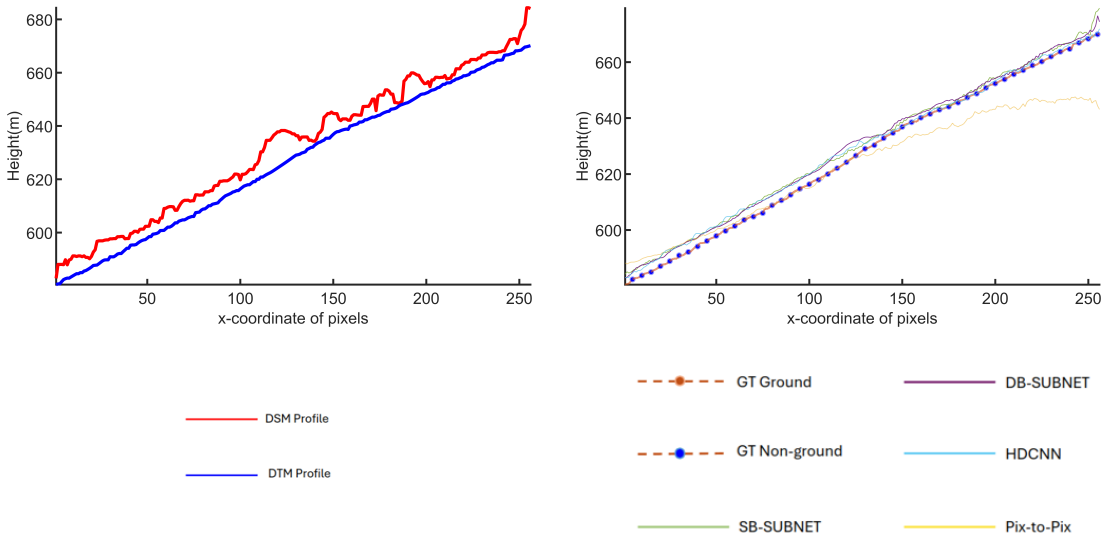


Figure 5.19: Cross-sections of each method for an S9 sample.

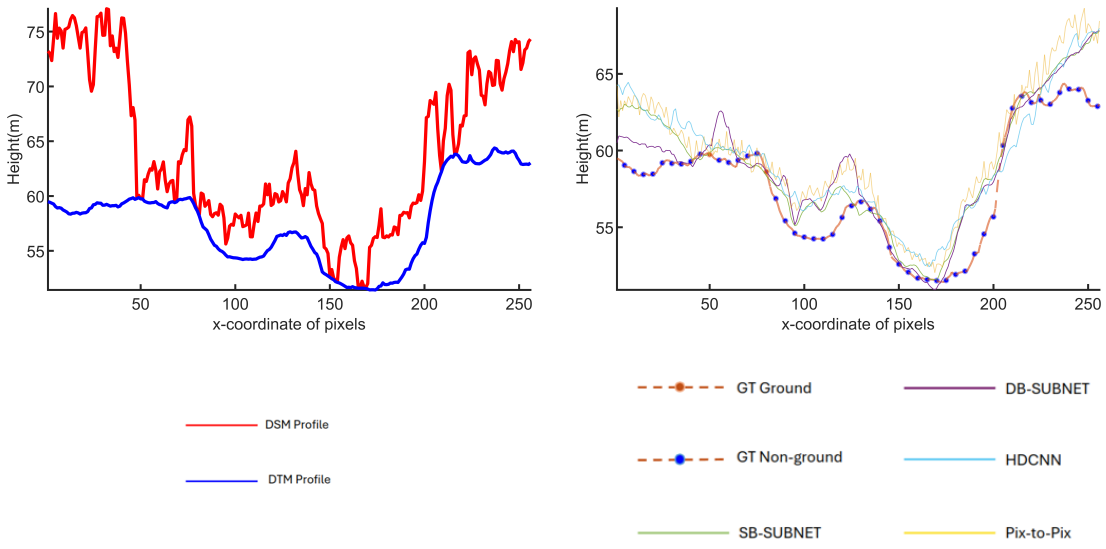


Figure 5.20: Cross-sections of each method for a NB sample.

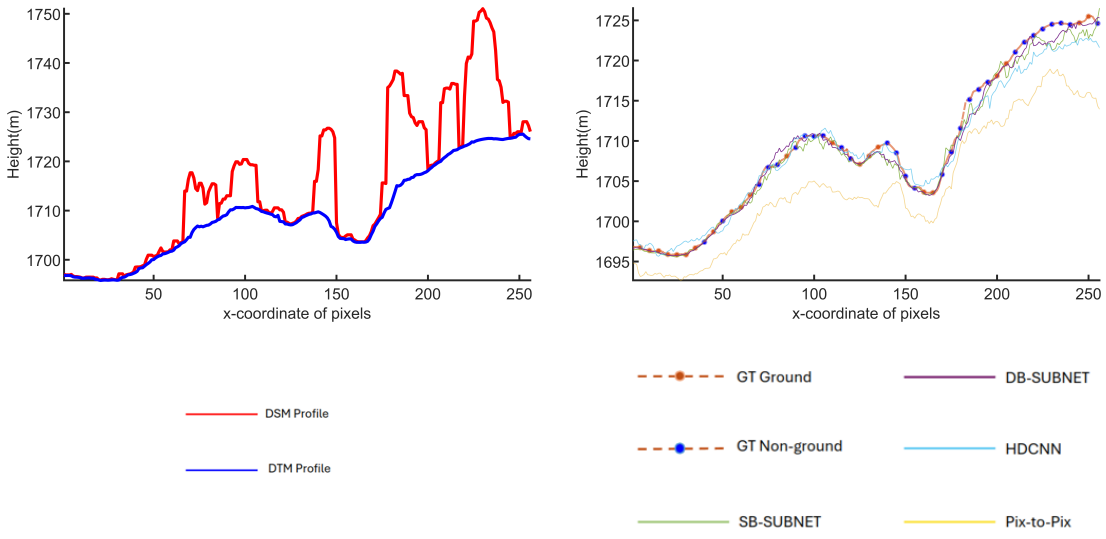


Figure 5.21: Cross-sections of each method for a KW sample.

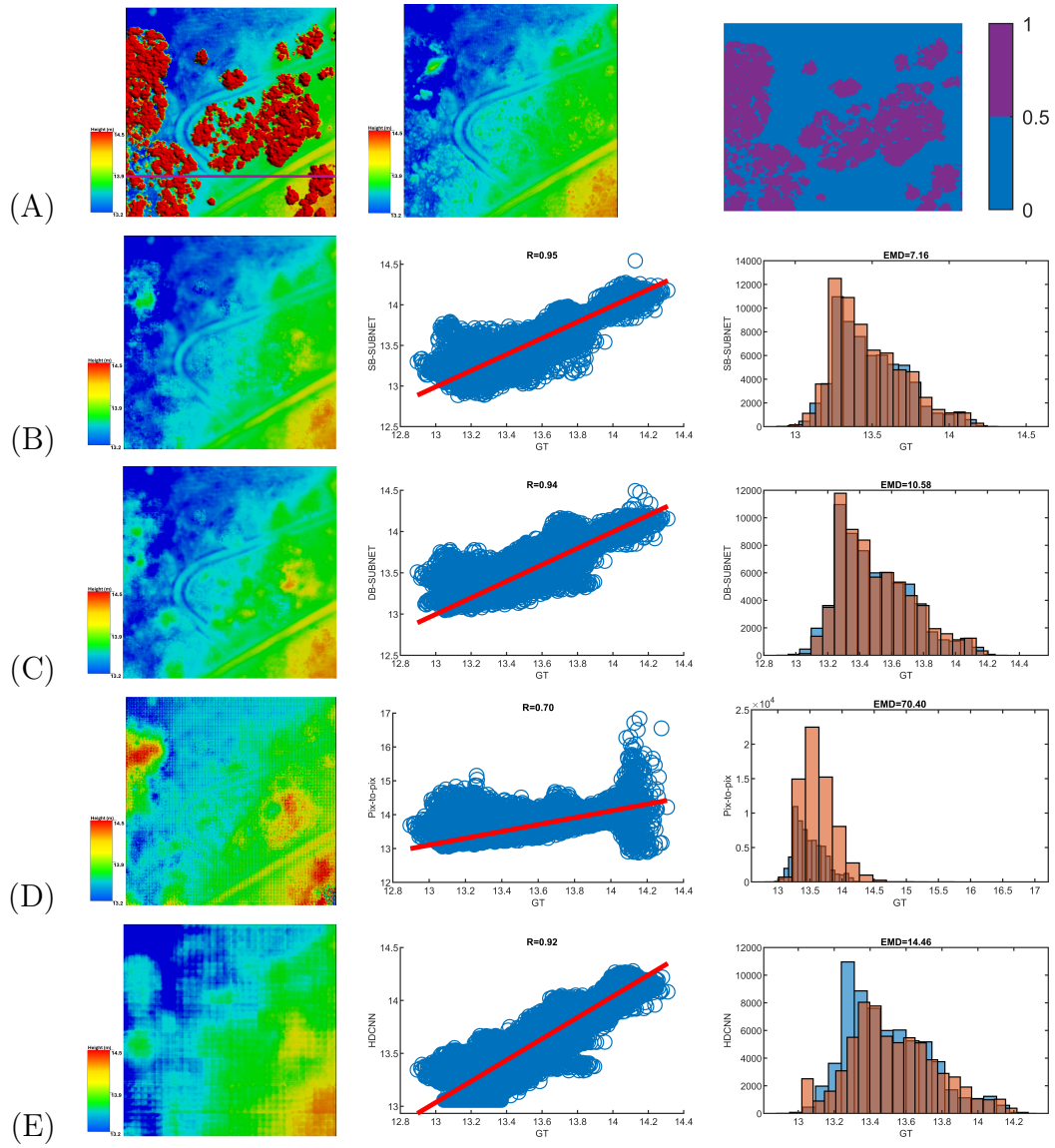


Figure 5.22: DTM generation results for RT: (A) input's network and ground truth, (B) SB-SUBNET, (C) DB-SUBNET, (D) pix-to-pix, (E) HDCNN.

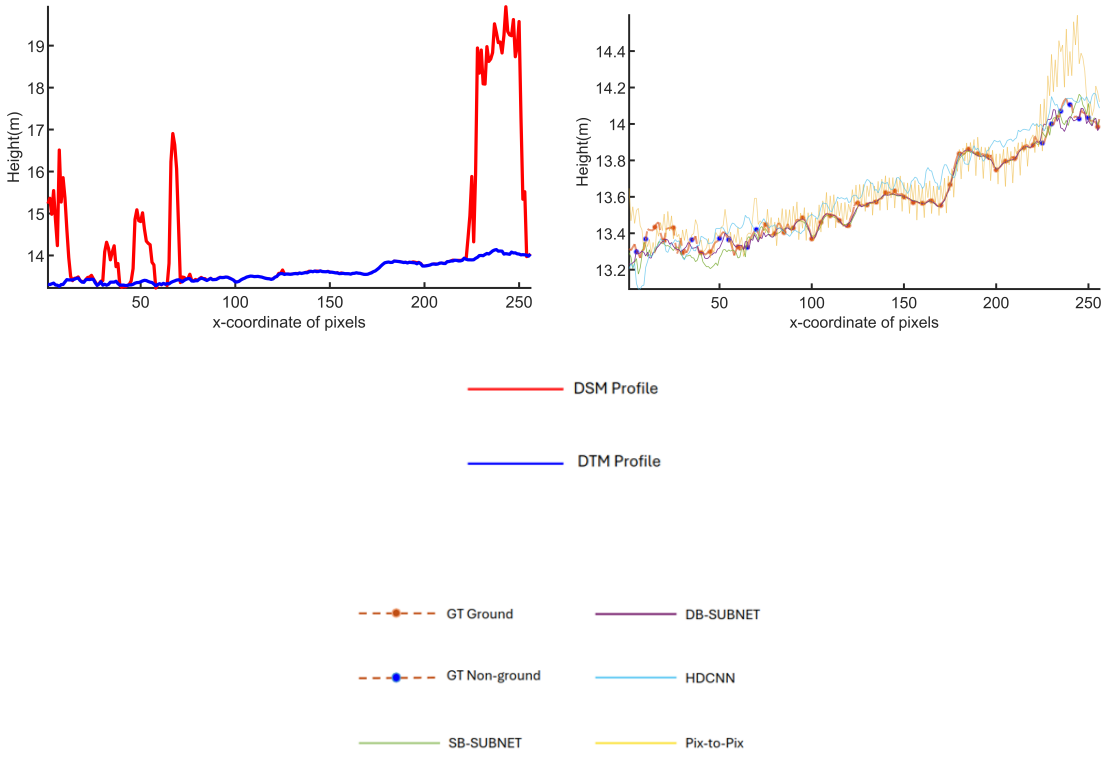


Figure 5.23: Cross-sections of each method for an RT sample.

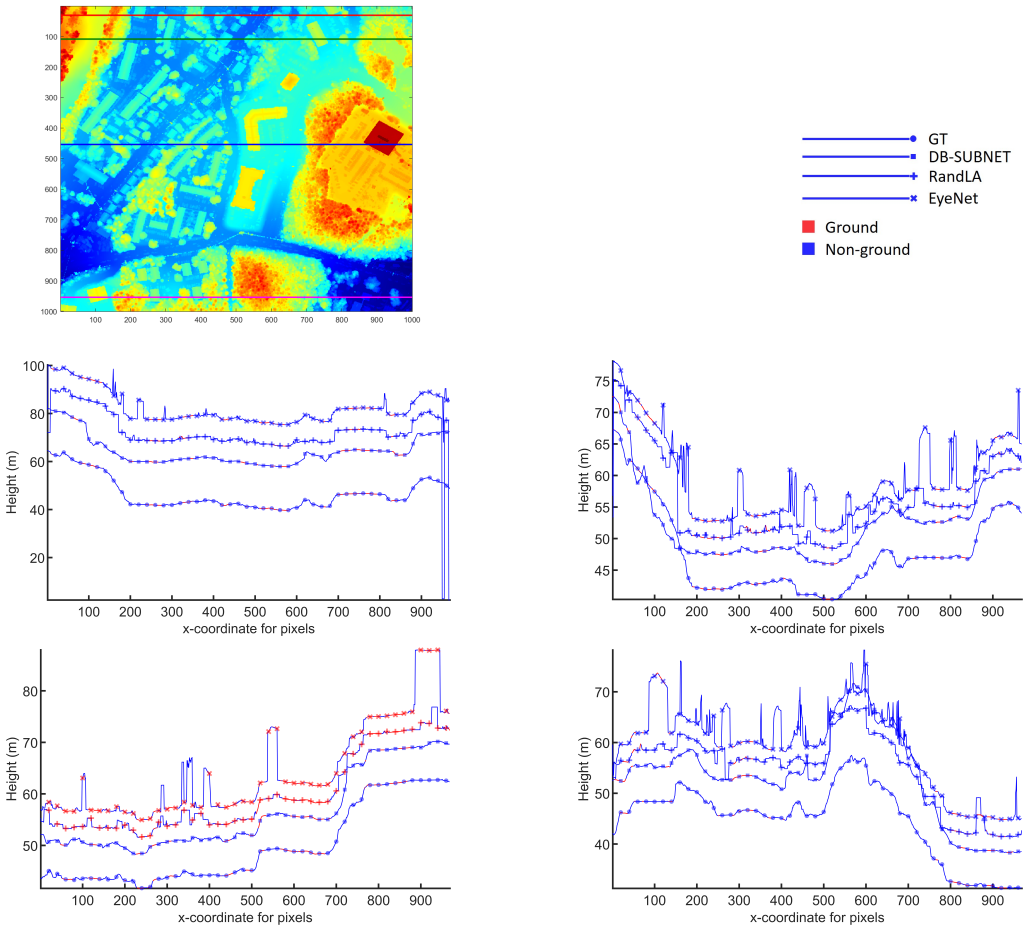


Figure 5.24: Cross-sectional analysis for classification-based methods versus DB-SUBNET.

5.5 Summary

In this chapter, we have extensively examined the domain of depth completion, focusing on generating DTMs from DSMs. Our exploration spanned a comprehensive overview of existing methodologies, delving into the distinctions between blind and non-blind depth completion approaches and underscoring the inherent complexities of terrain reconstruction, especially in environments covered with non-terrain elements like vegetation and buildings.

This chapter significantly advances by introducing a novel multi-task autoencoder architecture featuring a joint encoder and dual decoders. This architecture, designed to tackle terrain segmentation and gap-filling concurrently, demonstrates a robust approach to DTM generation. By incorporating subtractive skip connections, our method infuses domain-specific knowledge into the network, thereby enhancing the fidelity of the terrain reconstruction process.

Our extensive evaluation, encompassing a series of ablation studies and intra-domain generalization tests, underscores the superiority of our proposed DB-SUBNET method. Through rigorous comparisons across diverse datasets encompassing urban, mountainous, and mixed terrains, DB-SUBNET consistently outperformed existing state-of-the-art methods in terms of accuracy, particularly in challenging environments characterized by dense vegetation or complex urban structures.

The cross-domain validation further attested to the adaptability and robustness of our method. Despite the variance in terrain characteristics across different datasets, DB-SUBNET maintained high levels of accuracy, demonstrating its efficacy in generalizing across unseen terrains. This is particularly noteworthy in the context of DTM generation, where the precision of terrain representation is paramount for various applications ranging from urban planning to environmental monitoring.

A crucial insight from our comparative analysis is the limitation of segmentation-based methods in terrain estimation. While moderately successful in identifying non-ground objects, their dependency on interpolation for terrain reconstruction often leads to suboptimal results,

particularly in areas with large voids or needing more ground points for extrapolation. Our method's ability to effectively fill gaps without altering the terrain area contrasts these methods, showcasing its potential for more accurate and reliable terrain estimation.

In the last chapter, we have delved into the sophisticated domain of depth completion, particularly emphasizing the multi-task generation of DTMs from DSMs. Our exploration spanned various methodologies, focusing on the transition from traditional methods to cutting-edge supervised approaches, highlighting the role of deep learning in transforming the landscape of depth completion.

The core of our analysis was the introduction and evaluation of a novel multi-task autoencoder architecture, DB-SUBNET, designed to concurrently tackle terrain segmentation and gap-filling. This approach represents a significant leap forward, combining segmentation and regression to enhance DTM generation accuracy. Our experimental results, validated against state-of-the-art benchmarks across diverse and challenging datasets, showcased the superior performance of DB-SUBNET, particularly in its ability to maintain the integrity of ground points and effectively differentiate between terrain features. A comparative analysis with existing methods revealed that while traditional and some deep learning-based approaches have their merits, they often fall short in complex environments or when precise differentiation between terrain and non-terrain features is required.

Our proposed DB-SUBNET architecture has proven to be a powerful tool for various terrains, surpassing other methods in key metrics such as RMSE, MAE, and IOU. Particularly, it has performed exceptionally well in environments with intricate vegetation and urban structures. Moreover, our method's versatility in operating efficiently in supervised and unsupervised settings (unseen areas) showcases its adaptability and flexibility to different data conditions and requirements. It's worth noting that incorporating domain-specific knowledge through subtractive skip connections provides a more nuanced understanding of the terrain.

Chapter 6

Conclusion and future direction

6.1 Conclusion

This chapter of my dissertation covers the intricate challenges of estimating depth in dynamic environments. Moreover, it addresses the issues that arise due to the limitations of computational resources, especially with deep learning frameworks. The introduction of DARS marks a significant advancement in the field, offering a fine, in-domain adaptive strategy that enhances the depth estimation process during inference, thereby circumventing the need for extensive retraining on new datasets.

The integration of DARS with a pre-trained monodepth2 network, enabling adaptive refinements through channel-wise scaling and biasing, exemplifies the potential of functionally adaptive neural networks in real-world applications. This dual-module approach, combining a novel distance-based correction module with an optimization module utilizing particle swarm optimization, not only optimizes the depth estimation process but also sets new standards in accuracy and efficiency. The effectiveness of DARS has been validated through empirical evaluations on the KITTI and NYUv2 datasets. The correction module has established a strong baseline, and the optimization module has further refined the depth estimations, resulting in superior accuracy. The adaptability and robustness of this methodology have been extended to panoramic data through PanoDARS, indicating its potential for various

applications beyond traditional settings. The study's results highlight the significance of PanoDARS' adaptive and computationally efficient features in enhancing the visual SLAM capabilities of HDPV-SLAM.

The next chapter clarifies the significance and complexities involved in DTM reconstruction. Traditional methods of DTM creation have been greatly enhanced by modern technologies like LiDAR and remote sensing, which have revolutionized the precision and efficiency of these models. Despite advancements, extracting DTMs from DSMs remains challenging, particularly when addressing complex terrains. Various methods for extracting DTMs have been explored, including slope-based, morphology-based, interpolation-based, and segmentation-based techniques. However, these methods face significant challenges when applied to complex terrains, which has led to a shift towards deep learning strategies, particularly CNNs. These networks offer promising solutions to the challenges faced by traditional methods. The incorporation of a geospatially informed autoencoder in this study represents a significant advancement in DTM reconstruction. Through the utilization of a deep learning framework, this innovative approach circumvents the requirement for extensive post-processing, a common constraint in current methodologies. The proposed methodology incorporates two geospatial inductive biases into the neural networks. The efficacy of these biases is assessed through comparative analysis and ablation studies conducted on three distinct benchmarks: ALS2DTM, OpenGF, and USGS, as detailed in this chapter.

The primary geospatial bias pertains to the indirect reconstruction of DTM, achieved by subtracting the nDSM from the network's input, DSM. In this context, the direct versions, SUBNETD and CNETD, are compared with the indirect versions, SUBNETI and CNETI, respectively. The superior performance of the indirect versions underscores the efficiency of the first bias. The second geospatial bias involves using subtractive skip connections, which were evaluated through autoencoders equipped with such connections. In this case, SUBNETD performed better than CNETD. The only difference between the two was the type of skip connections used - SUBNETD had subtractive skip connections, while CNETD had concatenative skip connections. The overall evaluation, finally, indicates that The SUBNETI

method proposed in this study outperforms its counterparts in terms of accuracy, efficiency, and stability across diverse terrain conditions. The comparative analysis between SUBNETI and CNETI highlights how using a piece of prior knowledge in deep learning models can significantly enhance model stability and performance, especially when subtractive skip connections are employed. Additionally, the robustness and adaptability of SUBNETI, as demonstrated through its dataset-specific performance evaluations, are key qualities for a model intended for widespread geospatial applications. SUBNETI has a notable limitation: the regression process impacts all pixels, including those that accurately represent ground points from LiDAR. This underscores the necessity for further improvements.

To address the aforementioned issue, Chapter Five introduces an enhancement to SUBNETI. This enhancement involves integrating a segmentation branch in addition to the existing regression branch. For clarity, in this chapter, SUBNETI is referred to as SB-SUBNET, contrasting it with DB-SUBNET, which features two branches. The primary distinction between the two lies in their learning approaches: SB-SUBNET employs single-task learning, while DB-SUBNET utilizes multi-task learning. DB-SUBNET adopts a multi-task learning approach to optimize both the regression and segmentation branches. This is achieved by incorporating a regularizer between the two branches, ensuring that the regression branch exclusively regresses values for non-ground areas. Experimental findings on the ALS2DTM, OpenGF, and USGS datasets demonstrate that DB-SUBNET effectively preserves the LiDAR-derived ground points, resulting in superior overall performance compared to SB-SUBNET.

Despite the promising outcomes of the proposed DTM reconstruction methods in this thesis, particularly in comparison to various state-of-the-art methods across multiple datasets, its precision remains limited to within 4 meters for specific datasets such as S9 and KW, which are densely forested areas. This level of precision underscores the necessity of acknowledging the constraints of the proposed methods and identifying specific applications where such precision is both relevant and acceptable. These applications are as follows:

In agricultural land management, DTMs with an accuracy of 2 to 4 meters are used to plan large-scale farming operations, manage soil erosion, and optimize water flow across vast

areas. This level of accuracy is sufficient for identifying broad land features and planning crop layouts, irrigation systems, and erosion control measures [210]. Similarly, in forestry management, DTMs help estimate forest biomass, monitor deforestation, and plan logging operations. The acceptable accuracy allows for effective forest resource management, enabling the identification of significant terrain features and changes over large forested areas [211].

In urban planning and infrastructure development, DTMs with a 2 to 4-meter accuracy are used for broad-scale urban development projects, initial feasibility studies, and infrastructure maintenance. This accuracy level provides a reliable overview of the urban landscape, aiding in the planning of roads, buildings, and utilities [212]. Environmental monitoring also benefits from DTMs at this accuracy, particularly for assessing natural disaster impacts, monitoring land cover changes, and planning conservation efforts. The resolution is sufficient for detecting significant environmental changes and supporting large-scale environmental assessments [213].

6.2 Future studies

To advance the field of depth estimation and DTM reconstruction, it is essential to identify and pursue several key areas of future research. These endeavors aim to refine the methodologies and expand the capabilities of existing systems, with a specific focus on optimizing, interpolating, and generalizing DARS and enhancing the input variety and network architecture for DTM reconstruction. The following points outline the potential research directions, emphasizing a methodical and scholarly approach to advancing the current state of the art in these domains.

- **Comparative Analysis of Global Optimizations:** A systematic comparative analysis should be undertaken to identify the optimal global optimization technique for enhancing the DARS, with particular emphasis on evaluating various methodologies beyond the currently utilized PSO.
- **Evaluation of Interpolation Methods:** It is imperative to explore and assess the efficacy of incorporating finite element methods within the interpolation module of

DARS, aiming to benchmark their performance against existing methodologies to ascertain potential improvements in interpolation accuracy.

- **Generalization of Imaging Geometry for DARS:** The adaptability of DARS should be expanded to encompass a wider array of imaging geometries, extending its applicability from the conventional perspective to panoramic imaging, thereby enhancing its utility in diverse application scenarios.

- **Incorporation of Diverse Input Features in DTM Reconstruction:** Future research should focus on the enhancement of DTM reconstruction methodologies by integrating additional input features, such as slope and aspect, alongside the conventional DSM, to enrich the model's input data spectrum.

- **Integration of Lidar-Derived Feature Layers:** There is a necessity to harness a layer stack of lidar-derived features, including voxel-top, voxel-mean, and voxel-bottom, as inputs for DTM reconstruction, facilitating a more nuanced and detailed data foundation for the reconstruction process.

- **Utilization of Point Clouds as Input:** The development of methodologies that enable the processing of point cloud data as input, extracting relevant features in a voxel format for DTM output, should be prioritized to bridge the gap between raw point cloud data and its application in DTM reconstruction.

- **Adoption of Advanced Neural Network Architectures:** Investigating the application of cutting-edge neural network architectures, such as SegFormer, in DTM reconstruction is recommended, with a focus on leveraging features like subtractive skip connections and the indirect generation of DTM from normalized DSM (nDSM), to harness recent advancements in neural network research and improve reconstruction outcomes.

Bibilography

References

[1] Julián Cristiano, M. Salman Asif, and Domenec Puig. "Monocular Depth Estimation Using Deep Learning: A Review". In: *Sensors* 22.14 (2022), p. 5353. URL: https://doi.org/10.3390/s22145353.

[2] Peter Biber and Wolfgang Straßer. "The normal distributions transform: A new approach to laser scan matching". In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. Vol. 3. IEEE. 2003, pp. 2743–2748.

[3] Jürgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 573–580.

[4] Mostafa Elhashash, Hessah Albanwan, and Rongjun Qin. "A Review of Mobile Mapping Systems: From Sensors to Applications". In: *Sensors* 22.11 (2022), p. 4262. URL: https://mdpi-res.com/d_attachment/sensors/sensors-22-04262/article_deploy/sensors-22-04262-with-cover.pdf?version=1667351096.

[5] Elliott D Kaplan and Christopher Hegarty. *Understanding GPS: Principles and Applications*. Artech House, 2005.

[6] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous Localization and Mapping: Part I". In: *IEEE Robotics & Automation Magazine* 13.2 (2006), pp. 99–110.

[7] Paul D Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. Artech House, 2013.

[8] Paul J Besl and Neil D McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256.

[9] Yang Chen and Gerard Medioni. "Object modeling by registration of multiple range images". In: *Proceedings. IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE. 1991, pp. 2724–2729.

[10] Michael Kaess, Arvind Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378.

[11] Ian Dowman and Andrew B. Smith. "Digital Terrain Models". In: *Manual of Photogrammetry*. American Society of Photogrammetry, 1980, pp. 441–500.

[12] Zhengyou Zhang. "Flexible camera calibration by viewing a plane from unknown orientations". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. IEEE. 1999, pp. 666–673.

[13] Thomas H Kolbe, Gerhard Gröger, and Lutz Plümer. "CityGML: Interoperable access to 3D city models". In: *Geo-information for Disaster Management*. Springer, 2005, pp. 883–899.

[14] Paul V Bolstad. *GIS Fundamentals: A First Text on Geographic Information Systems*. 5th ed. Eider Press, 2016.

[15] Michael J Smith and João W Painho. "Assessing the accuracy of digital terrain models created from airborne laser scanning data". In: *International Journal of Geographical Information Science* 18.8 (2004), pp. 771–789.

[16] David F Maune. *Digital Elevation Model Technologies and Applications: The DEM Users Manual*. 2nd ed. American Society for Photogrammetry and Remote Sensing, 2007.

[17] Stan Z Li and Anil K Jain. *Encyclopedia of Biometrics*. Springer, 2015.

[18] Dirk J. C. B. van de Walle. “Hydrological applications of digital terrain models”. In: *Proceedings of the Third International Conference on Geographic Information Systems*. 1992, pp. 223–231.

[19] Robert J Nicholls. “Coastal flooding and wetland loss in the 21st century: Changes under the SRES climate and socio-economic scenarios”. In: *Global Environmental Change* 14.1 (2004), pp. 69–86.

[20] Tatiana Bandrova, Milan Konecny, and Sisi Zlatanova. *Thematic Cartography for the Society*. Springer, 2014.

[21] Michael K Lindell, Carla Prater, and Ronald W Perry. *Introduction to Emergency Management*. Wiley, 2006.

[22] Jie Shan and Charles K Toth. *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press, 2008.

[23] George Vosselman and Hans-Gerd Maas. *Airborne and Terrestrial Laser Scanning*. Whittles Publishing, 2010.

[24] David F Maune. “Digital Elevation Models: Theory and Applications”. In: *Digital Elevation Model Technologies and Applications*. American Society for Photogrammetry and Remote Sensing, 2007.

[25] Thomas H Kolbe. “CityGML—3D City Models and Their Applications”. In: *Photogrammetrie, Fernerkundung, Geoinformation* 6 (2009), pp. 485–496.

[26] Minsoo Song and Wonjun Kim. “Depth estimation from a single image using guided deep network”. In: *IEEE Access* 7 (2019), pp. 142595–142606.

[27] Alican Mertan, Damien Jade Duff, and Gozde Unal. “Single image depth estimation: An overview”. In: *Digital Signal Processing* 123 (2022), p. 103441.

[28] Yue Ming et al. “Deep learning for monocular depth estimation: A review”. In: *Neurocomputing* 438 (2021), pp. 14–33.

[29] Haisong Liu et al. “Camliflow: bidirectional camera-lidar fusion for joint optical flow and scene flow estimation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5791–5801.

[30] Omar Elharrouss et al. “Image inpainting: A review”. In: *Neural Processing Letters* 51 (2020), pp. 2007–2028.

[31] Amin Alizadeh Naeini, Mohammad Moein Sheikholeslami, and Gunho Sohn. “An Adaptive Refinement Scheme for Depth Estimation Networks”. In: *Sensors* 22.24 (2022), p. 9755.

[32] Mostafa Ahmadi et al. “HDPV-SLAM: Hybrid depth-augmented panoramic visual SLAM for mobile mapping system with tilted LiDAR and panoramic visual camera”. In: *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2023, pp. 1–8.

[33] Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

[34] Jorge Nocedal and Stephen J Wright. “Large-scale unconstrained optimization”. In: *Numerical optimization* (2006), pp. 164–192.

[35] Anton J Kleywegt and Alexander Shapiro. “Stochastic optimization”. In: *Handbook of industrial engineering* (2001), pp. 2625–2649.

[36] Dimitri Bertsekas. *Network optimization: continuous and discrete models*. Vol. 8. Athena Scientific, 1998.

[37] Ramachandran Jaikumar and Roger E Bohn. “A dynamic approach to operations management: An alternative to static optimization”. In: *International Journal of Production Economics* 27.3 (1992), pp. 265–282.

[38] Dragan Savic. “Single-objective vs. multiobjective optimisation for integrated decision support”. In: (2002).

[39] Sanjoy Das et al. "Adding local search to particle swarm optimization". In: *2006 IEEE International Conference on Evolutionary Computation*. IEEE. 2006, pp. 428–433.

[40] Anuj Kumar et al. *Meta-heuristic optimization techniques: applications in engineering*. Vol. 10. Walter de Gruyter GmbH & Co KG, 2022.

[41] Saad Hikmat Haji and Adnan Mohsin Abdulazeez. "Comparison of optimization techniques based on gradient descent algorithm: A review". In: *PalArch's Journal of Archaeology of Egypt/Egyptology* 18.4 (2021), pp. 2715–2743.

[42] Ruo-Yu Sun. "Optimization for deep learning: An overview". In: *Journal of the Operations Research Society of China* 8.2 (2020), pp. 249–294.

[43] Shun-ichi Amari. "Backpropagation and stochastic gradient descent method". In: *Neurocomputing* 5.4-5 (1993), pp. 185–196.

[44] Benoit Liquet, Sarat Moka, and Yoni Nazarathy. *The mathematical engineering of deep learning*. 2024.

[45] Agnes Lydia and Sagayaraj Francis. "Adagrad—an optimizer for stochastic gradient descent". In: *Int. J. Inf. Comput. Sci* 6.5 (2019), pp. 566–568.

[46] S Vani and TV Madhusudhana Rao. "An experimental approach towards the performance assessment of various optimizers on convolutional neural network". In: *2019 3rd international conference on trends in electronics and informatics (ICOEI)*. IEEE. 2019, pp. 331–336.

[47] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[48] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[49] Roberto Battiti. "First-and second-order methods for learning: between steepest descent and Newton's method". In: *Neural computation* 4.2 (1992), pp. 141–166.

[50] John D Head and Michael C Zerner. "A Broyden—Fletcher—Goldfarb—Shanno optimization procedure for molecular geometries". In: *Chemical physics letters* 122.3 (1985), pp. 264–270.

[51] Dewi Retno Sari Saputro and Purnami Widyaningsih. "Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR)". In: *AIP conference proceedings*. Vol. 1868. 1. AIP Publishing. 2017.

[52] Zahra Beheshti and Siti Mariyam Hj Shamsuddin. "A review of population-based meta-heuristic algorithms". In: *Int. j. adv. soft comput. appl* 5.1 (2013), pp. 1–35.

[53] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95-international conference on neural networks*. Vol. 4. IEEE. 1995, pp. 1942–1948.

[54] Ian Goodfellow et al. "Deep Learning-Ian Goodfellow, Yoshua Bengio, Aaron Courville". In: *Adapt. Comput. Mach. Learn.* (2016).

[55] Jingjing Xu et al. "Understanding and improving layer normalization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[56] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[57] Yuxin Wu and Kaiming He. "Group normalization". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[58] Andrea Galassi, Marco Lippi, and Paolo Torroni. "Attention in natural language processing". In: *IEEE transactions on neural networks and learning systems* 32.10 (2020), pp. 4291–4308.

[59] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[60] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

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

[62] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer. 2015, pp. 234–241.

[63] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark". In: *Neurocomputing* (2022).

[64] Larry R Medsker and LC Jain. "Recurrent neural networks". In: *Design and Applications* 5.64-67 (2001), p. 2.

[65] Alex Graves and Alex Graves. "Long short-term memory". In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[66] Rahul Dey and Fathi M Salem. "Gate-variants of gated recurrent unit (GRU) neural networks". In: *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*. IEEE. 2017, pp. 1597–1600.

[67] Ceren Güzel Turhan and Hasan Sakir Bilge. "Recent trends in deep generative models: a review". In: *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. IEEE. 2018, pp. 574–579.

[68] Antonia Creswell et al. "Generative adversarial networks: An overview". In: *IEEE signal processing magazine* 35.1 (2018), pp. 53–65.

[69] Juan Terven et al. "Loss functions and metrics in deep learning. A review". In: *arXiv preprint arXiv:2307.02694* (2023).

[70] François Chollet. *Deep Learning with Python*. Manning Publications, 2021.

[71] Aston Zhang et al. *Dive into Deep Learning*. Amazon Web Services, 2021.

[72] Mohammad Koushafar. "Automated Plume Rise Measurement based on Deep Neural Networks using Video Images". In: (2024).

[73] Wayne Xin Zhao et al. "A survey of large language models". In: *arXiv preprint arXiv:2303.18223* (2023).

[74] Andrew Cotter et al. "Better mini-batch algorithms via accelerated gradient methods". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 1647–1655.

[75] Sandro Skansi. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer, 2018.

[76] T. Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. New York: Springer, 2009, pp. 1–758.

[77] P. Cunningham, M. Cord, and S. J. Delany. "Supervised learning". In: *Machine learning techniques for multimedia: case studies on organization and retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49.

[78] Zoubin Ghahramani. "Unsupervised learning". In: *Summer school on machine learning*. Springer, 2003, pp. 72–112.

[79] Xiaojin Zhu and Andrew B Goldberg. *Introduction to semi-supervised learning*. Springer Nature, 2022.

[80] Yao-Hung Hubert Tsai et al. "Self-supervised learning from a multi-view perspective". In: *arXiv preprint arXiv:2006.05576* (2020).

[81] Sebastian Ruder. "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098* (2017).

[82] Neeraj Gupta. "A Pre-Trained Vs Fine-Tuning Methodology in Transfer Learning". In: *Journal of Physics: Conference Series*. Vol. 1947. 1. IOP Publishing. 2021, p. 012028.

[83] Joel Michael. "Where's the evidence that active learning works?" In: *Advances in physiology education* (2006).

[84] Amirreza Shaban et al. "One-shot learning for semantic segmentation". In: *arXiv preprint arXiv:1709.03410* (2017).

[85] Wei Wang et al. "A survey of zero-shot learning: Settings, methods, and applications". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 10.2 (2019), pp. 1–37.

[86] Shruti Jadon and Aryan Jadon. "An overview of deep learning architectures in few-shot learning domain". In: *arXiv preprint arXiv:2008.06365* (2020).

[87] Wenpeng Yin. "Meta-learning for few-shot natural language processing: A survey". In: *arXiv preprint arXiv:2007.09604* (2020).

[88] Mei Wang and Weihong Deng. "Deep visual domain adaptation: A survey". In: *Neurocomputing* 312 (2018), pp. 135–153.

[89] Xin Yi et al. "Two-step image dehazing with intra-domain and inter-domain adaptation". In: *Neurocomputing* 485 (2022), pp. 1–11.

[90] Konstantin Sofiiuk et al. "f-brs: Rethinking backpropagating refinement for interactive segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8623–8632.

[91] Parth Goel and Amit Ganatra. "Unsupervised Domain Adaptation for Image Classification and Object Detection Using Guided Transfer Learning Approach and JS Divergence". In: *Sensors* 23.9 (2023), p. 4436.

[92] Eric Tzeng et al. "Adversarial discriminative domain adaptation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7167–7176.

[93] Gabriela Csurka. "Domain adaptation for visual applications: A comprehensive survey". In: *arXiv preprint arXiv:1702.05374* (2017).

[94] Adrian Lopez-Rodriguez and Krystian Mikolajczyk. "Desc: Domain adaptation for depth estimation via semantic consistency". In: *International Journal of Computer Vision* 131.3 (2023), pp. 752–771.

[95] Jun Li, Reinhard Klein, and Angela Yao. "A two-streamed network for estimating fine-scaled depth maps from single rgb images". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 3372–3380.

[96] Arun CS Kumar, Suchendra M Bhandarkar, and Mukta Prasad. "Depthnet: A recurrent neural network architecture for monocular depth prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 283–291.

[97] Xingjian Shi et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in neural information processing systems* 28 (2015).

[98] Hyungjoo Jung et al. "Depth prediction from a single image with conditional adversarial networks". In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2017, pp. 1717–1721.

[99] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 270–279.

[100] Tinghui Zhou et al. "Unsupervised learning of depth and ego-motion from video". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1851–1858.

[101] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. "Semi-supervised deep learning for monocular depth map prediction". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6647–6655.

[102] Nikolaus Mayer et al. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4040–4048.

[103] Xiaojuan Qi et al. "Geonet: Geometric neural network for joint depth and surface normal estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 283–291.

[104] Zhenyu Zhang et al. "Progressive hard-mining network for monocular depth estimation". In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3691–3702.

[105] Huan Fu et al. "Deep ordinal regression network for monocular depth estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2002–2011.

[106] Won-Dong Jang and Chang-Su Kim. "Interactive image segmentation via backpropagating refinement scheme". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5297–5306.

[107] Axel Friedrich, Konrad Polthier, and Markus Schmies. *Interpolation of triangle hierarchies*. IEEE, 1998.

[108] David Salomon and David Salomon. "Linear Interpolation". In: *The Computer Graphics Manual* (2011), pp. 483–503.

[109] Daniel Patel. *Interactive Data Processing and 3D Visualization of the Solid Earth*. Springer, 2021.

[110] Ricardo A Olea. *Geostatistics for engineers and earth scientists*. 2000.

[111] Rahil Garnavi, Mohammad Aldeen, and ME Celebi. "Weighted performance index for objective evaluation of border detection methods in dermoscopy images". In: *Skin Research and Technology* 17.1 (2011), pp. 35–44.

[112] Mohammad Hossin and Md Nasir Sulaiman. "A review on evaluation metrics for data classification evaluations". In: *International journal of data mining & knowledge management process* 5.2 (2015), p. 1.

[113] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. "The earth mover's distance as a metric for image retrieval". In: *International journal of computer vision* 40 (2000), pp. 99–121.

[114] David Prokhorov et al. "Measuring robustness of Visual SLAM". In: *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE. 2019, pp. 1–6.

[115] K. Yousif, Y. Taguchi, and S. Ramalingam. "MonoRGBD-SLAM: Simultaneous localization and mapping using both monocular and RGBD cameras". In: *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. Piscataway Township, NJ, USA, May 2017, pp. 4495–4502.

[116] Florian Sauerbeck et al. "RGB-L: Enhancing Indirect Visual SLAM Using LiDAR-Based Dense Depth Maps". In: *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*. IEEE. 2023, pp. 95–100.

[117] George Dimas, Panagiota Gatoula, and Dimitris K Iakovidis. "MonoSOD: Monocular Salient Object Detection based on Predicted Depth". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4377–4383.

[118] Mu Hu et al. "Penet: Towards precise and efficient image guided depth completion". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 13656–13662.

[119] Jinsun Park et al. "Non-local spatial propagation network for depth completion". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*. Springer. 2020, pp. 120–136.

[120] Akhil Gurram et al. "Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision". In: *arXiv preprint arXiv:2103.12209* (2021).

[121] Noriaki Hirose et al. “Plg-in: Pluggable geometric consistency loss with wasserstein distance in monocular depth estimation”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 12868–12874.

[122] Jun Liu et al. “MiniNet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 166 (2020), pp. 255–267.

[123] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. “Adabins: Depth estimation using adaptive bins”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4009–4018.

[124] Sihaeng Lee et al. “Patch-Wise Attention Network for Monocular Depth Estimation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 3. 2021, pp. 1873–1881.

[125] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth map prediction from a single image using a multi-scale deep network”. In: *arXiv preprint arXiv:1406.2283* (2014).

[126] Xinjing Cheng et al. “Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 10615–10622.

[127] Jin Han Lee et al. “From big to small: Multi-scale local planar guidance for monocular depth estimation”. In: *arXiv preprint arXiv:1907.10326* (2019).

[128] Yu-Kai Huang et al. “S3: Learnable Sparse Signal Superdensity for Guided Depth Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16706–16716.

[129] Yuxuan Liu, Yuan Yixuan, and Ming Liu. “Ground-aware monocular 3d object detection for autonomous driving”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 919–926.

[130] Mian Mian Lau and King Hann Lim. "Review of adaptive activation function in deep neural network". In: *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*. IEEE. 2018, pp. 686–690.

[131] Xun Huang and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 1501–1510.

[132] Clément Godard et al. "Digging into self-supervised monocular depth estimation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3828–3838.

[133] Xinchen Ye et al. "DRM-SLAM: Towards dense reconstruction of monocular SLAM with scene depth fusion". In: *Neurocomputing* 396 (2020), pp. 76–91.

[134] Jiawang Bian et al. "Unsupervised scale-consistent depth and ego-motion learning from monocular video". In: *Advances in neural information processing systems* 32 (2019), pp. 35–45.

[135] Zhichao Yin and Jianping Shi. "Geonet: Unsupervised learning of dense depth, optical flow and camera pose". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1983–1992.

[136] Hong Cai et al. "X-Distill: Improving Self-Supervised Monocular Depth via Cross-Task Distillation". In: *arXiv preprint arXiv:2110.12516* (2021).

[137] Tuo Feng and Dongbing Gu. "Sganvo: Unsupervised deep visual odometry and depth estimation with stacked generative adversarial networks". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4431–4437.

[138] Pan Ji et al. "Monoindoor: Towards good practice of self-supervised monocular depth estimation for indoor environments". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 12787–12796.

[139] Anurag Ranjan et al. “Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 12240–12249.

[140] Xiaohan Fei, Alex Wong, and Stefano Soatto. “Geo-supervised visual depth prediction”. In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 1661–1668.

[141] Nícolas dos Santos Rosa, Vitor Guizilini, and Valdir Grassi. “Sparse-to-continuous: Enhancing monocular depth estimation using occupancy maps”. In: *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE. 2019, pp. 793–800.

[142] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. “Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3288–3295.

[143] Rajendra Madhav Palnitkar and James Cannady. “A review of adaptive neural networks”. In: *IEEE SoutheastCon, 2004. Proceedings.* IEEE. 2004, pp. 38–47.

[144] Theodora Kontogianni et al. “Continuous adaptation for interactive object segmentation by learning from corrections”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 579–596.

[145] J. Smith and N. Topin. “Enhancing Neural Network Performance through Advanced Weight Modification Techniques”. In: *Journal of Artificial Intelligence Research* 58.4 (2022), pp. 771–789. DOI: 10.5555/1234567.7654321.

[146] M. Johnson, A. Roberts, and X. Liu. “Adaptive Learning Rate Algorithms for Improved Neural Network Training”. In: *Proceedings of the International Conference on Machine Learning*. Vol. 35. 2023, pp. 1024–1032.

[147] Yongyi Chen, Dan Zhang, and Ruqiang Yan. “Domain adaptation networks with parameter-free adaptively rectified linear units for fault diagnosis under variable oper-

ating conditions". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[148] Vladimír Kunc. "Exploring the Relationship: Transformative Adaptive Activation Functions in Comparison to Other Activation Functions". In: *arXiv:2402.09249* (2024).

[149] Isaac Amidror. "Scattered data interpolation methods for electronic imaging systems: a survey". In: *Journal of electronic imaging* 11.2 (2002), pp. 157–176.

[150] Mostafa Ahmadi. "Visual-LiDAR Simultaneous Localization and Mapping". Master's thesis. York University, 2023. URL: `https://yorkspace.library.yorku.ca/items/9edd2924-c514-4fd1-a770-74071c8bc646`.

[151] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)*. 2013.

[152] Nathan Silberman, David Sontag, and Rob Fergus. "Instance segmentation of indoor scenes using a coverage loss". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 616–631.

[153] Chenxu Luo et al. "Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding". In: *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), pp. 2624–2641.

[154] Jianing Zhang et al. "LEAD: LiDAR Extender for Autonomous Driving". In: *arXiv preprint arXiv:2102.07989* (2021).

[155] Minhyeok Lee et al. "EdgeConv with Attention Module for Monocular Depth Estimation". In: *arXiv preprint arXiv:2106.08615* (2021).

[156] Jungwon Kang et al. "RPV-SLAM: Range-augmented panoramic visual SLAM for mobile mapping system with panoramic camera and tilted LiDAR". In: *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE. 2021, pp. 1066–1072.

[157] Nikolaos Xafoulis et al. "Evaluation of various resolution DEMs in flood risk assessment and practical rules for flood mapping in data-scarce geospatial areas: A case study in Thessaly, Greece". In: *Hydrology* 10.4 (2023), p. 91.

[158] Tatiane Ferreira Olivatto, Felipe Facci Inguaggiato, and Fábio Noel Stanganini. "Urban mapping and impacts assessment in a Brazilian irregular settlement using UAV-based imaging". In: *Remote Sensing Applications: Society and Environment* 29 (2023), p. 100911.

[159] Yi-Ru Wu et al. "Applications of Multi-temporal DTMs in Mining Management and Environmental Analysis". In: *EGU General Assembly Conference Abstracts*. 2023, EGU–15511.

[160] Zhilin Li, Christopher Zhu, and Chris Gold. *Digital terrain modeling: principles and methodology*. CRC press, 2004.

[161] Vladimir A Knyaz et al. "3D reconstruction of a complex grid structure combining UAS images and deep learning". In: *Remote Sensing* 12.19 (2020), p. 3128.

[162] Pinliang Dong and Qi Chen. *LiDAR remote sensing and applications*. CRC Press, 2017.

[163] Chuanfa Chen et al. "Segmentation-based hierarchical interpolation filter using both geometric and radiometric features for LiDAR point clouds over complex scenarios". In: *Measurement* 211 (2023), p. 112668.

[164] Aldino Rizaldy. "Deep learning-based DTM extraction from LIDAR point cloud". MA thesis. University of Twente, 2018.

[165] T. J. Pingel, K. C. Clarke, and W. A. McBride. "An improved simple morphological filter for the terrain classification of airborne lidar data". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 77 (2013), pp. 21–30.

[166] P. Axelsson. "DEM generation from laser scanner data using adaptive TIN models". In: *International Archives of Photogrammetry and Remote Sensing* 33 (2000), pp. 110–117.

[167] D. Továri and N. Pfeifer. "Segmentation based robust interpolation-a new approach to laser data filtering". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36 (2005), pp. 79–84.

[168] W. Zhang et al. "An easy-to-use airborne lidar data filtering method based on cloth simulation". In: *Remote Sensing* 8.6 (2016), p. 501. DOI: 10.3390/rs8060501.

[169] Hamed Amini Amirkolaee et al. "DTM extraction from DSM using a multi-scale DTM fusion strategy based on deep learning". In: *Remote Sensing of Environment* 274 (2022), p. 113014.

[170] Nadeem Fareed, Joao Paulo Flores, and Anup Kumar Das. "Analysis of UAS-LiDAR ground points classification in agricultural fields using traditional algorithms and PointCNN". In: *Remote Sensing* 15.2 (2023), p. 483.

[171] Hengming Dai et al. "Deep ground filtering of large-scale ALS point clouds via iterative sequential ground prediction". In: *Remote Sensing* 15.4 (2023), p. 961.

[172] Nannan Qin et al. "Towards intelligent ground filtering of large-scale topographic point clouds: A comprehensive survey". In: *International Journal of Applied Earth Observation and Geoinformation* 125 (Dec. 2023), p. 103566.

[173] Sunghwan Yoo et al. "Human vision based 3d point cloud semantic segmentation of large-scale outdoor scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 6576–6585.

[174] Hoàng-Ân Lê et al. "Learning Digital Terrain Models From Point Clouds: ALS2DTM Dataset and Rasterization-Based GAN". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2022), pp. 4980–4989.

[175] Haruki Oshio, Keiichiro Yashima, and Masashi Matsuoka. "Generating DTM from DSM Using a Conditional GAN in Built-Up Areas". In: *IEEE Geoscience and Remote Sensing Letters* (2023).

[176] Ksenia Bittner et al. "DSM2DTM: AN END-TO-END DEEP LEARNING APPROACH FOR DIGITAL TERRAIN MODEL GENERATION". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 10 (2023), pp. 925–933.

[177] Konstantinos Gavriil, Georg Muntingh, and Oliver JD Barrowclough. "Void filling of digital elevation models with deep generative models". In: *IEEE Geoscience and Remote Sensing Letters* 16.10 (2019), pp. 1645–1649.

[178] Guoqing Zhou et al. "Voids filling of DEM with multiattention generative adversarial network model". In: *Remote Sensing* 14.5 (2022), p. 1206.

[179] Sijin Li et al. "Integrating topographic knowledge into deep learning for the void-filling of digital elevation models". In: *Remote Sensing of Environment* 269 (2022), p. 112818.

[180] United States Geological Survey. *What types of elevation datasets are available, what formats do they come in, and where can I download them?* `https://www.usgs.gov/faqs/what-types-elevation-datasets-are-available-what-formats-do-they-come-and-where-can-i-download`. Accessed: 2023-01-15. 2021.

[181] Nannan Qin et al. "Deep learning for filtering the ground from ALS point clouds: A dataset, evaluations and issues". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 202 (2023), pp. 246–261.

[182] Zexiao Xie et al. "Recent advances in conventional and deep learning-based depth completion: A survey". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[183] Lin Bai et al. "DepthNet: Real-time LiDAR point cloud depth completion for autonomous vehicles". In: *IEEE Access* 8 (2020), pp. 227825–227833.

[184] Juan Wang et al. "Self-Prior Guided Pixel Adversarial Networks for Blind Image Inpainting". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[185] Xiaobo Zhang et al. “Image inpainting based on deep learning: A review”. In: *Information Fusion* 90 (2023), pp. 74–94.

[186] Amir Atapour-Abarghouei and Toby P Breckon. “DepthComp: real-time depth image completion based on prior semantic scene segmentation.” In: (2017).

[187] Petteri Alho, Hannu Hyyppä, and Juha Hyyppä. “Consequence of DTM precision for flood hazard mapping: A case study in SW Finland”. In: *Nordic journal of surveying and real estate research* 6.1 (2009).

[188] Heather McGrath et al. “A comparison of rapid DTM based approaches for on-demand flood inundation mapping”. In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 8397–8400.

[189] Henrik Persson et al. “Estimating forest biomass and height using optical stereo satellite data and a DTM from laser scanning data”. In: *Canadian Journal of Remote Sensing* 39.3 (2013), pp. 251–262.

[190] Hossein Arefi and Peter Reinartz. “Multi-Level Building Reconstruction for Automatic Enhancement of High Resolution DSMS”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39 (2012), pp. 11–16.

[191] Deming Kong et al. “A real-time method for DSM generation from airborne LiDAR data”. In: *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE. 2013, pp. 377–380.

[192] Stephan Gehrke et al. “Semi-global matching: An alternative to LIDAR for DSM generation”. In: *Proceedings of the 2010 Canadian Geomatics Conference and Symposium of Commission I*. Vol. 2. 6. 2010.

[193] Shichao Jin et al. “A point-based fully convolutional neural network for airborne lidar ground point filtering in forested environments”. In: *IEEE journal of selected topics in applied earth observations and remote sensing* 13 (2020), pp. 3958–3974.

[194] Ming Huang, Pengcheng Wei, and Xianglei Liu. "An efficient encoding voxel-based segmentation (EVBS) algorithm based on fast adjacent voxel search for point cloud plane segmentation". In: *Remote Sensing* 11.23 (2019), p. 2727.

[195] Zhenyang Hui et al. "Automatic DTM extraction from airborne LiDAR based on expectation-maximization". In: *Optics & Laser Technology* 112 (Apr. 2019), pp. 43–55.

[196] Bowen Li et al. "Terrain-Net: A Highly-Efficient, Parameter-Free, and Easy-to-Use Deep Neural Network for Ground Filtering of UAV LiDAR Data in Forested Environments". In: *Remote Sensing* 14.22 (2022), p. 5798.

[197] Zeynep Akbulut, Fevzi Karsli, and Mustafa Dihkan. "A New Approach for Ground Filtering of Airborne Laser Scanning Data Using PointNet++". In: *Journal of the Indian Society of Remote Sensing* 52.1 (2024), pp. 1–15.

[198] M. Jaritz et al. "Sparse and dense data with CNNs: Depth completion and semantic segmentation". In: (2018), pp. 52–60.

[199] R. Schön, K. Ludwig, and R. Lienhart. "Impact of pseudo depth on open world object segmentation with minimal user guidance". In: (2023), pp. 4808–4818.

[200] Yu Zhang and Qiang Yang. "An overview of multi-task learning". In: *National Science Review* 5.1 (2018), pp. 30–43.

[201] Amir Atapour-Abarghouei and Toby P Breckon. "To complete or to estimate, that is the question: A multi-task approach to depth completion and monocular depth estimation". In: *2019 International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 183–193.

[202] Lina Gao et al. "Depth-aware inverted refinement network for RGB-D salient object detection". In: *Neurocomputing* 518 (2023), pp. 507–522.

[203] Adriano Cardace et al. "Boosting Multi-Modal Unsupervised Domain Adaptation for LiDAR Semantic Segmentation by Self-Supervised Depth Completion". In: *IEEE Access* (2023).

[204] Jean Lahoud et al. “3d instance segmentation via multi-task metric learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9256–9266.

[205] Mykhailo Shvets et al. “Joint depth prediction and semantic segmentation with multi-view sam”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2024, pp. 1328–1338.

[206] Nan Zou et al. “Simultaneous semantic segmentation and depth completion with constraint of boundary”. In: *Sensors* 20.3 (2020), p. 635.

[207] Amir Atapour-Abarghouei and Toby P Breckon. “Veritatem dies aperit-temporally consistent depth prediction enabled by a multi-task geometric and semantic scene understanding approach”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3373–3384.

[208] Phillip Isola et al. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[209] Qingyong Hu et al. “Randla-net: Efficient semantic segmentation of large-scale point clouds”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11108–11117.

[210] D.J. Mulla. “Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps”. In: *Biosystems Engineering* 114.4 (2013), pp. 358–371.

[211] S.C. Popescu, R.H. Wynne, and J.A. Scrivani. “Fusion of small-footprint lidar and multispectral data to estimate plot-level volume and biomass in deciduous and pine forests in Virginia, USA”. In: *Forest Science* 50.4 (2004), pp. 551–565.

[212] F. Biljecki, H. Ledoux, and J. Stoter. “Applications of 3D City Models: State of the Art Review”. In: *ISPRS International Journal of Geo-Information* 4.4 (2015), pp. 2842–2889.

[213] J.P. Wilson and J.C. Gallant. “Digital terrain modeling”. In: *Geomorphology* 37 (2000), pp. 107–121.

[214] Armin Masoumian et al. “Monocular Depth Estimation Using Deep Learning: A Review”. In: *Sensors* 22.14 (2022), p. 5353. DOI: 10.3390/s22145353.

[215] Iman Abaspur Kazerouni et al. “A survey of state-of-the-art on visual SLAM”. In: *Expert Systems with Applications* 205 (2022), p. 117734.

[216] Jun Cheng et al. “A review of visual SLAM methods for autonomous driving vehicles”. In: *Engineering Applications of Artificial Intelligence* 114 (2022), p. 104992.

[217] Egor Aksamentov et al. “Approach to robotic mobile platform path planning upon analysis of aerial imaging data”. In: *Proceedings of 15th International Conference on Electromechanics and Robotics" Zavalishin's Readings" ER (ZR) 2020, Ufa, Russia, 15–18 April 2020*. Springer. 2020, pp. 93–103.

[218] Okusimba George Omusotsi. “Role of GIS as a Toolfor Environmental Planning and Management”. In: (2019).

[219] Mostafa Elhashash, Hessah Albanwan, and Rongjun Qin. “A Review of Mobile Mapping Systems: From Sensors to Applications”. In: *Sensors* 22.11 (2022), p. 4262. DOI: 10.3390/s22114262.

[220] Haruki Oshio, Keiichiro Yashima, and Masashi Matsuoka. “Generating DTM from DSM Using a Conditional GAN in Coastal Urban Areas of Japan”. In: *Authorea Preprints* (2023).

[221] Paul E Utgoff. *Machine learning of inductive bias*. Vol. 15. Springer Science & Business Media, 2012.

[222] P. Rashidi and H. Rastiveis. “Extraction of ground points from lidar data based on slope and progressive window thresholding (SPWT)”. In: *Earth Observation and Geomatics Engineering* 2 (2018), pp. 36–44.

[223] J. Zhang and X. Lin. "Filtering airborne lidar data by embedding smoothness-constrained segmentation in progressive TIN densification". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 81 (2013), pp. 44–59.

[224] Vladimír Kunc and Jiří Kléma. "Three Decades of Activations: A Comprehensive Survey of 400 Activation Functions for Neural Networks". In: *arXiv preprint arXiv:2402.09092* (2024).

[225] Chen Fu et al. "Depth Completion via Inductive Fusion of Planar LIDAR and Monocular Camera". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10843–10848.

[226] Andreas Geiger et al. "Vision meets robotics: The kitti dataset". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[227] Shunping Ji et al. "Panoramic SLAM from a multiple fisheye camera rig". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 159 (2020), pp. 169–183.

[228] Qing Li et al. "3D map-guided single indoor image localization refinement". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 161 (2020), pp. 13–26.

[229] Nathan Silberman et al. "Indoor segmentation and support inference from rgbd images". In: *European conference on computer vision*. Springer. 2012, pp. 746–760.

[230] Yurong You et al. "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving". In: *arXiv preprint arXiv:1906.06310* (2019).

[231] Niclas Zeller, Franz Quint, and Uwe Stilla. "Depth estimation and camera calibration of a focused plenoptic camera for visual odometry". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 118 (2016), pp. 83–100.

[232] François Fleuret. "The little book of deep learning". In: *A lovely concise introduction* (2023), p. 297.

Appendices

A.3 Mapping and Depth Estimation in Panoramic Imaging

A.3.1 Mappings Between 3D Points and Pixels in the Panoramic Image

To establish correspondences from a pixel in the panoramic image to the camera's 3D coordinate system, we utilize the unit sphere projection and the polar coordinate system as intermediaries.

3D to Unit Sphere Projection

Transitioning from the 3D camera coordinate system to the unit sphere for a given 3D point involves calculating the range r, which is the distance from the point to the camera. This is given by:

$$r = \sqrt{X_c^2 + Y_c^2 + Z_c^2} \tag{6.1}$$

Here, (X_c, Y_c, Z_c) denotes the coordinates of the point in the 3D camera coordinate system. The projection onto the unit sphere is then obtained by normalizing these coordinates:

$$(X_u, Y_u, Z_u) = \left(\frac{X_c}{r}, \frac{Y_c}{r}, \frac{Z_c}{r}\right) \tag{6.2}$$

where (X_u, Y_u, Z_u) represents the point's projection on the unit sphere.

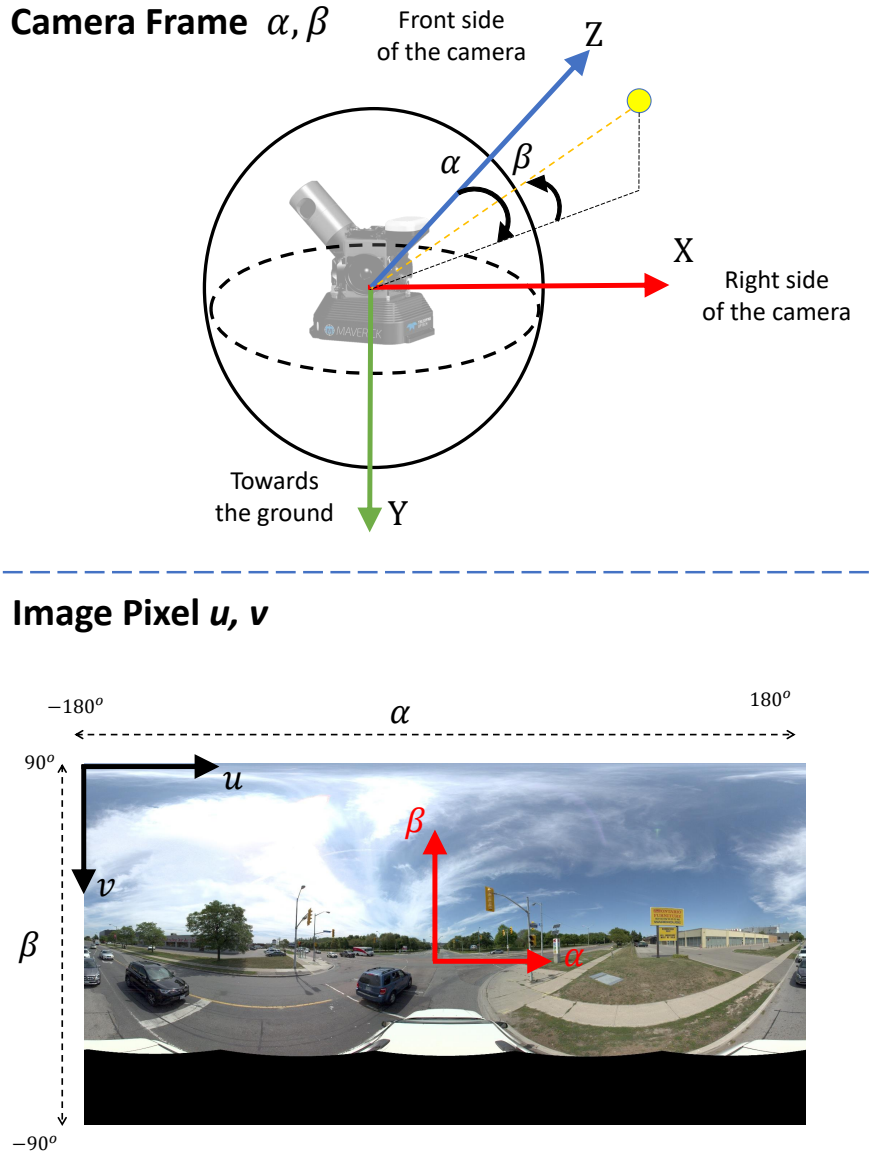


Figure 6.1: Various coordinate systems representation [156].

Conversely, converting from the unit sphere back to the 3D camera coordinate system requires knowing the range r for the 3D point, which can be obtained via methods such as LiDAR point-cloud data or SLAM-based techniques like triangulation. With r known, the conversion is expressed as:

$$(X_c, Y_c, Z_c) = (rX_u, rY_u, rZ_u) \tag{6.3}$$

Unit Sphere to Polar Coordinates

The mapping from the unit sphere to the polar coordinate system is described by:

$$\begin{aligned} \alpha &= \arctan 2(X_u, Z_u) \\ \beta &= -\arcsin(Y_u) \end{aligned} \tag{6.4}$$

where α and β represent the longitude and latitude, respectively, constrained within $\alpha \in [-180°, 180°]$ and $\beta \in [-90°, 90°]$.

The inverse transformation is obtained by:

$$\begin{aligned} X_u &= \cos(\beta) \cdot \sin(\alpha) \\ Y_u &= -\sin(\beta) \\ Z_u &= \cos(\beta) \cdot \cos(\alpha) \end{aligned} \tag{6.5}$$

Polar Coordinates to Image Pixels

The conversion from polar coordinates to pixel coordinates in the image is given by:

$$\begin{aligned} u &= w \cdot \left(0.5 + \frac{\alpha}{2\pi}\right) \\ v &= h \cdot \left(0.5 - \frac{\beta}{\pi}\right) \end{aligned} \tag{6.6}$$

where (u, v) denotes the pixel coordinates in the image, and h and w are the height and width of the image, respectively. The pixel coordinate space is defined as:

$$\phi = \{(u, v) \mid u, v \in \mathbb{W}, 0 \leq u < w, 0 \leq v < h\} \tag{6.7}$$

Using the extrinsic parameter matrix between the panoramic camera and the LiDAR sensor in the MMS, along with Equations 6.1, 6.2, 6.4, and 6.6, we project the 3D points onto the corresponding panoramic image, as shown in Figure 6.2.

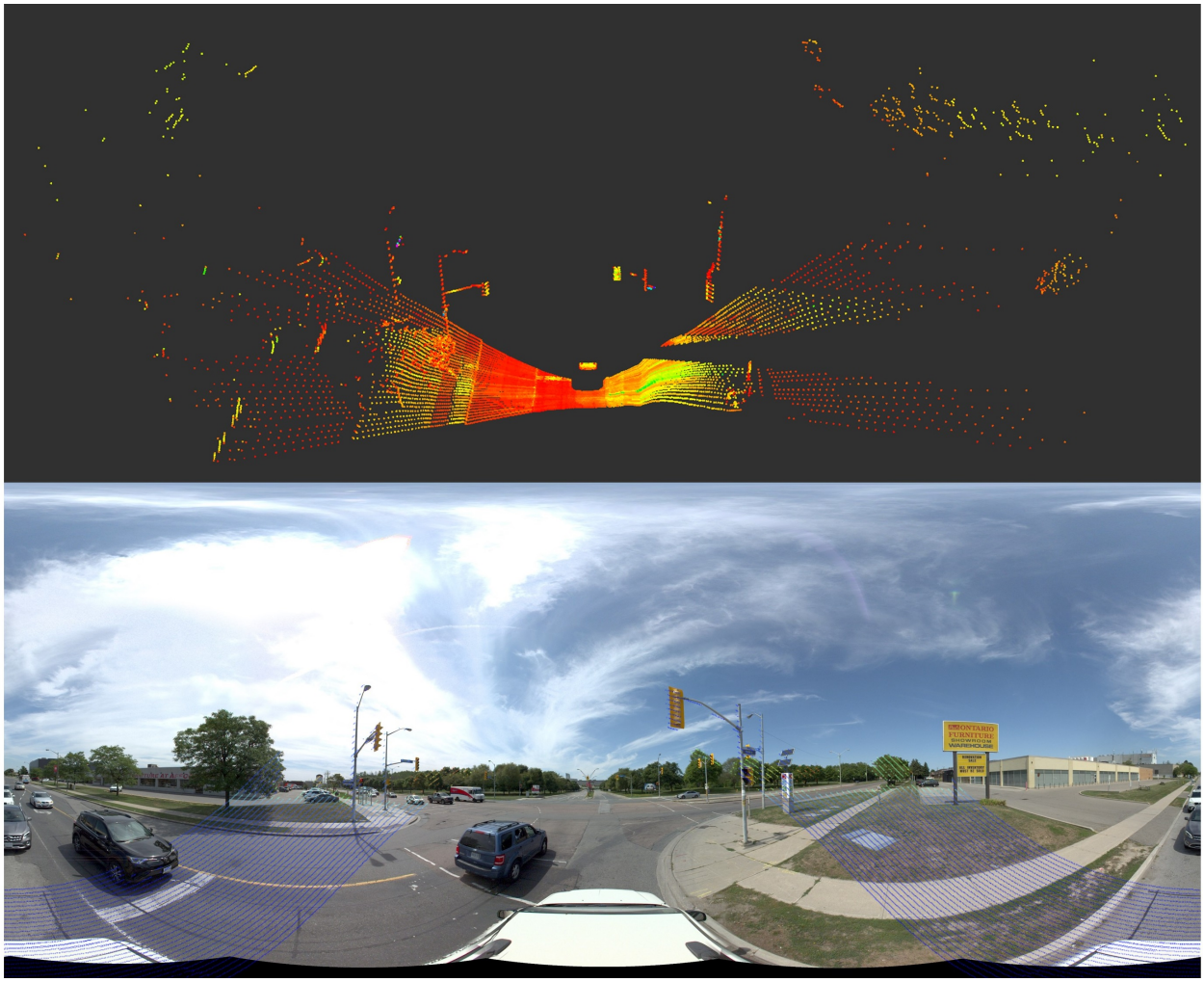


Figure 6.2: The 3D LiDAR points (up) and the points projected on the corresponding panoramic image (down) [156].

A.3.2 Depth Estimation Module

We begin by outlining the method used in our foundational work, RPV-SLAM [156], and then introduce our novel approach, PanoDARS.

Bi-Interpolation

To enhance depth estimation in our system, we first discuss the standard procedure of bi-interpolation used for densifying LiDAR points and creating a more comprehensive depth map. We aim to compare the performance of our deep learning module with this bi-interpolation method in later chapters.

Using the extrinsic parameters matrix and Equations 6.1, 6.2, 6.4, and 6.6, we project the 3D points onto the panoramic image at frame t, denoted by ${}^{l}\varphi_t \subset \phi$. For each point ${}^{l}\varphi_t^i \in {}^{l}\varphi_t$ at time t, we define a function ${}^{l}\Gamma_t$ returning its range value:

$$ {}^{l}\Gamma_t({}^{l}\varphi_t^i) = r_t^i \tag{6.8} $$

Dense Depth Map Creation

The projected LiDAR points ${}^{l}\varphi_t$ are sparse, and their probability of aligning with actual visual features for RGB-D SLAM usage is low. We hypothesize that pixels near ${}^{l}\varphi_t$ in the pixel coordinate space likely have similar depth values to their neighboring pixels. We categorize all pixels into three groups: ${}^{l}\varphi_t$ (pixels corresponding to a specific LiDAR point), ${}^{c}\varphi_t$ (pixels not covered by LiDAR but close to ${}^{l}\varphi_t$), and ${}^{f}\varphi_t$ (pixels not covered by LiDAR and far from ${}^{l}\varphi_t$):

$$ {}^{l}\varphi_t \cup^{c} \varphi_t \cup^{f} \varphi_t = \phi \tag{6.9} $$

Assuming pixels in ${}^{c}\varphi_t$ have similar depths to their neighbors in ${}^{l}\varphi_t$, we perform Delaunay triangulation on ${}^{l}\varphi_t$ and estimate the depth for each pixel ${}^{c}\varphi_t^j \in^{c} \varphi_t$ that falls within a triangle. The depth value is calculated using the range values of the triangle's vertices:

$$ {}^{c}\Gamma_t({}^{c}\varphi_t^j) = w_1 \cdot r_t^{i_1} + w_2 \cdot r_t^{i_2} + w_3 \cdot r_t^{i_3} \tag{6.10} $$

where w_1, w_2, and w_3 are the weights for linear barycentric interpolation.

We define a unified depth function for each pixel $\varphi_t^i \in \phi$:

$$\Gamma_t(\varphi_t^i) = \begin{cases} {}^l\Gamma_t(\varphi_t^i) & \text{if } \varphi_t^i \in {}^l\varphi_t \\ {}^c\Gamma_t(\varphi_t^i) & \text{if } \varphi_t^i \in {}^c\varphi_t \\ Null & \text{if } \varphi_t^i \in {}^f\varphi_t \end{cases} \tag{6.11}$$

This function, defined over the entire pixel coordinate space, is added as an additional channel to the RGB panoramic image to create an RGB-D panoramic image [150].

A.4 ORB Feature Detection and Description

ORB, or Oriented FAST and Rotated BRIEF, is a key algorithm in computer vision for feature detection and description. Initially, ORB applies the FAST (Features from Accelerated Segment Test) algorithm to detect keypoints. FAST operates by considering a circle of sixteen pixels around a candidate pixel p. A pixel p is classified as a corner if there are n contiguous pixels in the circle which are all brighter than the intensity of p plus a threshold T, or all darker than the intensity of p minus T. The value of n is typically set to 12.

Once keypoints are detected, ORB computes their orientation to achieve rotation invariance. The orientation θ is calculated based on the intensity centroid of the patch surrounding the keypoint. The moments m_{pq} of a patch are given by:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{6.12}$$

where $I(x, y)$ is the intensity of the pixel at (x, y). The centroid $C = (c_x, c_y)$ is then determined using the first order moments:

$$c_x = \frac{m_{10}}{m_{00}}, \quad c_y = \frac{m_{01}}{m_{00}} \tag{6.13}$$

The orientation θ is the arctangent of the central moment, defined as:

$$\theta = \arctan 2(m_{01}, m_{10}) \tag{6.14}$$

Following the orientation computation, ORB employs the Rotated BRIEF (Binary Robust Independent Elementary Features) descriptor for keypoint description. BRIEF creates a

binary descriptor by comparing the intensity of pairs of pixels within a smoothed image patch. Given a keypoint k, its descriptor D_k is formed by comparing the intensity of pairs of pixels around k:

$$D_k[i] = \begin{cases} 1 & \text{if } I(p_i) < I(q_i) \\ 0 & \text{otherwise} \end{cases} \tag{6.15}$$

where (p_i, q_i) are the selected pixel pairs. To achieve rotation invariance, the coordinates of these pixel pairs are rotated according to the keypoint orientation θ prior to the intensity comparison.

The efficiency and robustness of ORB make it particularly suitable for real-time applications where computational resources are limited, such as in mobile or embedded systems. However, it is important to note that while ORB is rotation-invariant, it is not scale-invariant, which can be a limitation in scenarios where scale invariance is crucial[150].

A.5 HDPV-SLAM Overview

The HDPV-SLAM system exemplifies advanced localization and mapping by intelligently integrating LiDAR and panoramic camera data. It processes these inputs to produce enriched RGB-D images, foundational for ORB feature extraction critical to the SLAM process. A depth estimation module further augments the system, converting sparse LiDAR data into comprehensive depth maps using deep learning. The system's core functionalities include a Tracking Module that aligns visual features with map points to ascertain camera positioning, and a Mapping Module that generates new map points from keyframes, enhancing map precision. Notably, the Depth Association Module innovatively synchronizes depth information with visual features, even in non-keyframe scenarios, substantially improving map point accuracy. The system culminates with the Loop Closing Module, which corrects trajectory errors through global bundle adjustments, ensuring overall accuracy and robustness of the mapping solution. This cohesive interplay of modules underscores the system's sophistication in delivering precise and reliable spatial awareness [150].

SLAM Input: In this stage, RGB-D images are created by merging panoramic RGB images with LiDAR point clouds. Frames are sequentially generated, each consisting of an RGB image and a projected sparse LiDAR map. These frames are crucial for extracting ORB features used in subsequent SLAM processes. The output includes camera positions and a comprehensive set of map points that are continuously updated with new frames.

Depth Estimation Module: The depth estimation module transforms sparse LiDAR maps into dense depth maps using deep learning methods. It involves projecting LiDAR points onto panoramic images and then densifying the depth map. Pixels are categorized based on their proximity to LiDAR points, with different depth estimation strategies applied to each category.

Tracking Module: This module connects visual features with existing map points. It processes frames in a sequential order, matching features to update camera pose and position. The determination of keyframes is based on the anticipated future performance of the tracking process.

Mapping Module: Upon identifying a keyframe, the mapping module creates new map points from the frame data, thereby updating the map. It also manages local bundle adjustments for improved mapping accuracy.

Depth Association Module: The Depth association module in [32, 150] is innovatively designed to optimally associate depth information with visual features within the observable region, denoted as OR_t, as can be seen in Figure 3.6. This module, activated post the tracking phase where all visible map points in the current frame F_t are tracked and the camera's pose j_t is estimated, operates even when F_t is not a keyframe. It marks a significant advancement from traditional methods by utilizing non-keyframes for depth association, thereby enhancing the accuracy of tracked map points and ensuring consistent mapping quality. Map points are generated either through direct LiDAR-based depth assignment or via triangulation of visual

features, based on their position in relation to OR_t. These methodologies lead to variable depth values for the same map point over time. The module addresses these variances by implementing a depth association task, guided by two policies. If a map point p_t^i is initially created through triangulation and has not been modified later, its position is updated to a new estimate ${}^{new}p_t^i$ based on the latest depth data. Furthermore, if a map point was previously modified or created based on a depth value $\hat{\mathbf{D}}_p(o_p^i)$ greater than the current depth observation $\hat{\mathbf{D}}_t(o_t^i)$, its position is updated to ${}^{new}p_t^i$.

To mitigate inaccuracies in ORB visual feature assignment by the tracking module, a rejection threshold θ is established. This threshold prevents significant trajectory alterations by rejecting substantial positional changes if the distance between the current and new positions of a map point exceeds θ. The detailed algorithm of the association module has been discussed in [150].

Complementing the Depth Association Module, the Loop Closing Module detects loops and performs global bundle adjustment on the trajectory. This optimization involves pose-graph optimization of the camera poses j_t in each frame F_t within the loop and optimization of all map points created in those keyframes using a similarity transformation.

Loop Closing Module: This module detects revisits to previously mapped areas and performs global bundle adjustments to correct accumulated trajectory errors. It enhances the overall accuracy of the SLAM solution, maintaining the integrity of the environment mapping over time.