DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 969

Qianqian Zou

3D mapping with probabilistic uncertainty measures

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978 3 7696 5381 6

DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 969

3D mapping with probabilistic uncertainty measures

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

M. Sc. Qianqian Zou

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978 3 7696 5381 6

Adresse des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften:

DGK

Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:
Vorsitzender: Prof. Dr.-Ing. Jakob Flury
Referentin: Prof. Dr.-Ing. habil. Monika Sester
Korreferenten: Prof. Dr.-Ing. habil. Christian Heipke
Prof. Dr.-Ing. Stefan Hinz

Tag der mündlichen Prüfung: 10.07.2025

:

Diese Dissertation ist auf dem Server des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften, München unter <http://dgk.badw.de/>
sowie unter Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik
der Leibniz Universität Hannover (ISSN 0174-1454), Nr. 417,
<https://repo.uni-hannover.de/handle/123456789/20003>, Hannover 2025, elektronisch publiziert

---

© 2025 Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus zu vervielfältigen.

ISSN 0065-5325
ISBN 978 3 7696 5381 6

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Dissertation selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur sowie Hilfsmittel erstellt zu haben. Ferner erkläre ich, dass

1. mir die Regeln der geltenden Promotionsordnung bekannt sind und eingehalten wurden; mit einer Prüfung nach den Bestimmungen der geltenden Fassung der Promotionsordnung bin ich einverstanden,

2. Dritten weder unmittelbar noch mittelbar geldwerte Leistungen für Vermittlungstätigkeiten oder für die inhaltliche Ausarbeitung der Dissertation erbracht wurden, d.h. die wissenschaftliche Arbeit wurde weder in Teilen noch in Gänze von Dritten gegen Entgelt oder sonstige Gegenleistung erworben oder vermittelt,

3. die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht wurde,

4. weder die vorliegende Dissertation noch eine in wesentlichen Teilen ähnliche Arbeit bei einer anderen Hochschule als Dissertation eingereicht wurde.

Hannover, 15.05.2025

Qianqian Zou

Abstract

Three-dimensional (3D) mapping plays a foundational role across a wide range of applications, including autonomous systems, urban planning, geographic information systems, and immersive digital environments such as augmented and virtual reality. Despite its importance, constructing accurate, efficient, and reliable 3D maps of large-scale urban environments remains a significant challenge due to the inherent complexity of real-world scenes, sensor noise, occlusions, and data sparsity. Traditional mapping methods often focus on accuracy and efficiency but fail to provide uncertainty quantification, which is crucial for safety-critical applications such as autonomous navigation. Without an explicit representation of uncertainty, even highly accurate models may mislead downstream tasks if local data errors or ambiguities are not properly accounted for.

This dissertation addresses the gap by proposing a probabilistic 3D mapping framework that models not only the geometric structure of urban environments but also quantifies the associated uncertainties. The proposed method integrates probabilistic modeling techniques—specifically Gaussian Mixture Models (GMM) and Gaussian Processes (GP)—to construct continuous, uncertainty-aware 3D map representations from Light Detection and Ranging (LiDAR) point clouds. GMM are used to encode informative priors that significantly improve the scalability of GP inference, enabling efficient modeling of large-scale environments. The framework was first developed for probabilistic building models, in which structures are segmented into local regions and modeled using a combination of GMM priors and local GPs. It was then extended to handle arbitrary geo-objects through 3D implicit surface representations. This generalization is achieved through hierarchical mixture models, Gaussian regression, and GP inference incorporating derivative observations. The resulting 3D Gaussian map provides both detailed geometry and well-calibrated spatial uncertainty estimates, which are shown to be beneficial for various tasks.

Beyond mapping, the dissertation explores the application of the proposed representation in two key areas: uncertainty-aware localization and 3D geometry generation. For localization, the Gaussian map allows for robust pose estimation under uncertain conditions by propagating map uncertainty into the localization process. For geometry generation, the integration with deep generative models—specifically diffusion models—demonstrates the utility of the 3D Gaussian representation as a shape prior, showing the potential in probabilistic reconstruction of occluded or missing regions. The framework thus bridges traditional mapping with probabilistic learning, offering a unified approach to both represent and reason about 3D environments under uncertainty.

The proposed methods are evaluated on both real-world and simulated datasets, encompassing large-scale urban environments and object-level 3D models, and are compared against state-of-the-art baselines. The results demonstrate that incorporating uncertainty into 3D map representations not only improves the robustness and interpretability of mapping but also opens new directions for uncertainty-aware generative modeling in 3D vision and robotics.

Keywords: 3D mapping, 3D reconstruction, probabilistic uncertainty estimation, LiDAR point clouds, Gaussian mixture model, Gaussian Process, 3D generation

Kurzfassung

Dreidimensionale (3D) Karten spielen eine grundlegende Rolle in einer Vielzahl von Anwendungen, einschließlich autonomer Systeme, Stadtplanung, geografischer Informationssysteme und immersiver digitaler Umgebungen wie Augmented und Virtual Reality. Trotz ihrer Bedeutung stellt die Erstellung genauer, effizienter und zuverlässiger 3D-Karten von großflächigen städtischen Umgebungen aufgrund der inhärenten Komplexität realer Szenen, des Sensorrauschens, der Verdeckungen und der geringen Datenmenge nach wie vor eine große Herausforderung dar. Herkömmliche Kartierungsmethoden konzentrieren sich oft auf Genauigkeit und Effizienz, bieten aber keine Quantifizierung der Unsicherheit, die für sicherheitskritische Anwendungen wie die autonome Navigation entscheidend ist. Ohne eine explizite Darstellung der Unsicherheit können selbst hochpräzise Modelle nachgelagerte Aufgaben in die Irre führen, wenn lokale Datenfehler oder Mehrdeutigkeiten nicht angemessen berücksichtigt werden.

Diese Dissertation schließt diese Lücke, indem sie ein probabilistisches 3D-Kartierungsverfahren vorschlägt, das nicht nur die geometrische Struktur der städtischen Umgebung modelliert, sondern auch die damit verbundenen Unsicherheiten quantifiziert. Die vorgeschlagene Methode integriert probabilistische Modellierungstechniken – insbesondere Gaußsche Mischmodelle (GMM) und Gaußsche Prozesse (GP) –, um kontinuierliche, unsicherheitsbewusste 3D-Kartendarstellungen aus Light Detection and Ranging (LiDAR)-Punktwolken zu erstellen. Gaußsche Mischmodelle werden verwendet, um informative Prioritäten zu kodieren, die die Skalierbarkeit der GP-Inferenz verbessern und eine effiziente Modellierung von großen Umgebungen ermöglichen. Das Framework wurde zunächst für probabilistische Gebäudemodelle entwickelt, in denen Strukturen in lokale Regionen segmentiert und mithilfe einer Kombination aus GMM-Prioritäten und lokalen GPs modelliert werden. Anschließend wurde es erweitert, um beliebige Geoobjekte durch implizite 3D-Oberflächendarstellungen zu verarbeiten. Diese Generalisierung wird durch hierarchische Mischmodelle, Gauß-Regression und GP-Inferenz unter Einbeziehung abgeleiteter Beobachtungen erreicht. Die resultierende 3D-Gaußkarte liefert sowohl detaillierte Geometrie als auch gut kalibrierte Schätzungen der räumlichen Unsicherheit, die sich für verschiedene Aufgaben als vorteilhaft erweisen.

Über die Kartierung hinaus untersucht die Dissertation die Anwendung der vorgeschlagenen Repräsentation in zwei Schlüsselbereichen: Lokalisierung unter Berücksichtigung der Unsicherheit und 3D-Geometrieerzeugung. Bei der Lokalisierung ermöglicht die Gaußkarte eine robuste Posenschätzung unter unsicheren Bedingungen, indem sie die Kartenunsicherheit in den Lokalisierungsprozess einfließen lässt. Bei der Geometrieerzeugung demonstriert die Integration mit tiefen generativen Modellen - insbesondere mit Diffusionsmodellen - den Nutzen der 3D-Gauß-Darstellung als Form-Prior und zeigt das Potenzial für die probabilistische Rekonstruktion verdeckter oder fehlender Regionen. Der Rahmen überbrückt somit traditionelles Mapping mit probabilistischem Lernen und bietet einen einheitlichen Ansatz, um 3D-Umgebungen unter Unsicherheit darzustellen und zu verstehen.

Die vorgeschlagenen Methoden werden sowohl anhand realer als auch simulierter Datensätze, die großflächige städtische Umgebungen und 3D-Modelle auf Objektebene umfassen, evaluiert und mit den modernsten Ansätzen verglichen. Die Ergebnisse zeigen, dass die Einbeziehung von Unsicherheiten in 3D-Kartendarstellungen nicht nur die Robustheit und Interpretierbarkeit der Kartierung verbessert, sondern auch neue Wege für die generative Modellierung von 3D-Vision und Robotik mit Blick auf Unsicherheiten eröffnet.

Schlüsselwörter: 3D-Mapping, 3D-Rekonstruktion, probabilistische Unsicherheitsschätzung, Punktwolken, Gaußsches Mischmodell, Gaußscher Prozess, 3D-Generierung

Contents

Abstract **v**

1 Introduction **1**
- 1.1 3D Mapping 1
- 1.2 Motivation 3
- 1.3 Goal and Contributions 5
- 1.4 Structure of the Dissertation 6

2 Theoretical Basics **7**
- 2.1 Map Representations 7
  - 2.1.1 Point Clouds 7
  - 2.1.2 Boundary Representation 7
  - 2.1.3 Occupancy Map 8
  - 2.1.4 Implicit Surface Map 10
  - 2.1.5 Explicit Mesh Surface 12
- 2.2 Probabilistic Uncertainty 13
  - 2.2.1 Bayesian Inference 13
  - 2.2.2 Multivariate Gaussian Distribution 14
  - 2.2.3 Gaussian Mixture Model 16
  - 2.2.4 Gaussian Mixture Regression 20
  - 2.2.5 Gaussian Process 21
  - 2.2.6 Statistical Tests for Assessing out-of-distribution data 25
  - 2.2.7 Uncertainty Propagation 26
- 2.3 LiDAR-based Localization 28
  - 2.3.1 Transformation 28
  - 2.3.2 Iterative Closest Point 31
  - 2.3.3 Normal Distribution Transform 32
- 2.4 Probabilistic Generative Diffusion Models 35
  - 2.4.1 Diffuse and Denoise 35
  - 2.4.2 Optimization Loss 36

3 Related Work **37**
- 3.1 Map representations 37
- 3.2 Gaussian Processe-based Mapping 38
- 3.3 Gaussian Mixture Model-based Mapping 40
- 3.4 Neural Implicit Models 41
- 3.5 Gaussian Splatting 42
- 3.6 3D Generation and Reconstruction 43
  - 3.6.1 General Generative Models 43
  - 3.6.2 3D Geometry Generation 43
  - 3.6.3 3D Completion 44

4 Datasets and Preprocessing 47
- 4.1 Datasets 47
  - 4.1.1 LUCOOP Dataset 47
  - 4.1.2 LARD Dataset 49
  - 4.1.3 CARLA Dataset 50
  - 4.1.4 ShapeNet Dataset 50
- 4.2 Data Preparation 51
  - 4.2.1 Preprocessing of LUCOOP Riegl Data for Mapping 51
  - 4.2.2 Preprocessing of LUCOOP Trajectory Data 51
  - 4.2.3 Pre-processing of ShapeNet Data 52
  - 4.2.4 Ground Truth Generation 53

5 Urban Building Models using GP and GMM in a 2.5D Local Frame 55
- 5.1 Overview 55
- 5.2 Façade Segmentation 57
  - 5.2.1 RANSAC Segmentation 57
  - 5.2.2 GMM Segmentation 58
- 5.3 GMM for Planar Surfaces 59
- 5.4 GP Using GMM as Prior 61
  - 5.4.1 Application of GP 61
  - 5.4.2 GMM as Prior 63
  - 5.4.3 GP-block Techniques 65
  - 5.4.4 Outlier Filtering 66
- 5.5 Experiments 66
  - 5.5.1 Experimental Setting 66
  - 5.5.2 Ablation Study 67
  - 5.5.3 Evaluations 67
  - 5.5.4 Computational Time Comparison 71
  - 5.5.5 Discussion 72
- 5.6 Summary 72

6 Uncertain Implicit Surface Mapping Using GP and HGMM 75
- 6.1 Overview 75
- 6.2 4D Hierarchical Gaussian Mixture Regression 76
  - 6.2.1 Hierarchical Structure 76
  - 6.2.2 Surface Distance Regression 77
  - 6.2.3 Data Augmentation 79
- 6.3 GP Inference with HGMM as Priors 80
  - 6.3.1 GP Regression with Non-stationary Priors from HGMM 80
  - 6.3.2 GP with Derivative Observations 81
  - 6.3.3 Integration with LogGP 82
  - 6.3.4 Octree-based GP Block 83
- 6.4 Uncertainty in SDF Estimation 83
- 6.5 Incremental Update 84
- 6.6 Experiments 85
  - 6.6.1 Experimental Setting 85
  - 6.6.2 Evaluation Metrics 86
  - 6.6.3 Ablation Study 87
  - 6.6.4 Surface Reconstruction 88
  - 6.6.5 Quantitative Evaluation 90
  - 6.6.6 Computation Time Comparison 93

6.6.7 Discussion 93
6.7 Summary 94

7 Uncertainty-aware Localization Using Uncertain Maps 97
7.1 Probabilistic Localization with GMMs 97
7.1.1 GMM Transform 98
7.2 Localization with Uncertain Implicit Surfaces 98
7.2.1 Pose Optimization 99
7.2.2 Uncertainty-aware Optimization 101
7.2.3 Uncertainty Propagated from Maps 101
7.3 Experiments 104
7.3.1 Experimental Setting 104
7.3.2 Accuracy Evaluation 104
7.3.3 Uncertainty Investigation 104
7.3.4 Discussion 107

8 Probabilistic Diffusion Models for 3D Gaussian Generation 109
8.1 3D Geometry Generation 109
8.1.1 Diffusion Model for 3D Gaussian 110
8.1.2 Diffusion Model for 4D Gaussian 112
8.1.3 Loss Functions 113
8.1.4 Denoising Backbones 113
8.2 Reconstruction with Partial Measurements 114
8.2.1 Posterior Sampling 114
8.2.2 Optimization-based Completion 115
8.3 Experiments 117
8.3.1 Experimental Setting 117
8.3.2 Diffusion Generation 117
8.3.3 Completion 121
8.3.4 Discussion 125

9 Conclusion and Outlook 127
9.1 Summary and Conclusion 127
9.2 Outlook 128

List of Figures 132

List of Tables 133

List of Abbreviations 137

Bibliography 139

Acknowledgements 151

Curriculum Vitae 151

Chapter 1

Introduction

1.1 3D Mapping

Three-dimensional (3D) mapping is a technique that captures and generates detailed representations of unknown environments. It serves as a critical foundation across diverse research domains, including geoinformatics and remote sensing, architecture and construction, robotics, computer graphics, and computer vision. This technology supports a wide range of applications, such as urban planning, architectural design, infrastructure management, autonomous navigation, and the creation of digital twins for real-time monitoring, simulation, and analysis of physical environments.

Mapping begins with the acquisition of data from an unknown physical environment using various sensing techniques. For instance, photogrammetry utilizes aerial, satellite, or terrestrial images, while LiDAR methods, such as Terrestrial Laser Scanning (TLS) and 3D Mobile Mapping, generate dense point clouds. Each method has its strengths and is chosen based on the required resolution and accuracy, as well as the environmental conditions. Once data is collected, it undergoes a comprehensive processing pipeline to construct a 3D model. This involves aligning and merging different data sources and creating a certain map representation that reflects the geometry and structure of the real-world environment at the required level of detail and quality.

3D mapping technologies encompass a variety of representations, each suited to different applications and levels of detail required in modelling environments. Figure 1.1 illustrates several representative examples. In the field of Geographical Information Science (GIS), traditional models such as Digital Elevation Models (DEM) and Triangulated Irregular Networks (TIN) are commonly employed to represent terrain elevation, supporting geospatial analysis and topographic studies. For urban environments, 3D City Models-often structured by levels of detail (e.g, CityGML Level of Detail 2 (LoD2) models)-are widely adopted to visualize urban environments with buildings, roads, and infrastructure, enabling applications in urban planning and smart city development. In the architectural and construction domains, Building Information Models (BIM) offer highly detailed and semantically rich 3D representations specifically for buildings, with metadata such as materials, structural elements, plumbing, and electrical systems.

Additionally, raw 3D point clouds obtained through Light Detection and Ranging (LiDAR) or photogrammetry can be simply used as a 3D map. While these point-based models provide dense and accurate geometric information, they often lack structure and semantic annotation, requiring further processing for downstream tasks. Other than the aforementioned representations, advanced mapping techniques have been developed in the fields of robotics and computer vision to meet the demands of perception and reasoning in complex environments; for example, 3D occupancy maps represent spatial information probabilistically, encoding the likelihood of occupancy in each voxel rather than relying on explicit geometric descriptions. In contrast to discrete representations, implicit surface maps, such as distance fields and neural radiance fields, define 3D geometry using continuous functions rather than explicit vertices or voxels, enabling smooth and resolution-independent modeling of surfaces. These representations are the primary focus of this dissertation, providing a foundation for scalable and expressive 3D mapping with uncertainties.

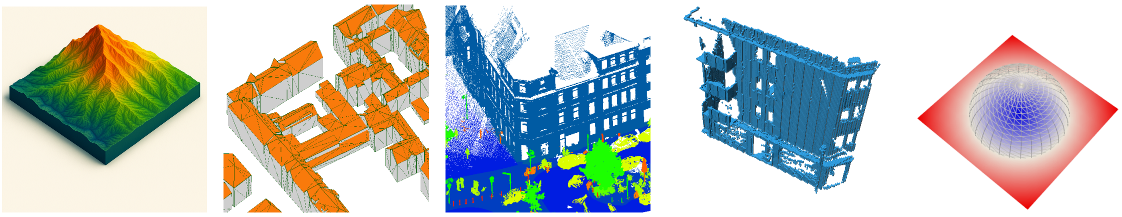

(a) DEM (b) LoD2 model (c) Point clouds (d) Occupancy map (e) Implicit field

FIGURE 1.1. 3D maps

The automated reconstruction of the physical world through 3D modeling from point cloud data, particularly in urban settings, is an area of active research within geoinformatics, remote sensing, and robotics. These fields focus on developing methods that efficiently and accurately transform large-scale, unstructured sensor data into detailed, structured 3D representations of geo-objects. In autonomous systems, mapping technology plays a critical role in assisting systems to comprehend unknown and complex scenes. Autonomous vehicles rely on 3D models to perceive and interpret their surroundings accurately, which includes static objects like roads and buildings, as well as dynamic elements such as other vehicles and pedestrians. The accurate mapping is crucial for localization, collision avoidance, path planning and maneuvering the vehicle safely through complex environments.

Beyond mapping and reconstructing the physical world in the domain of robotics and autonomous driving, a proper 3D mapping framework can also make digital or virtual contents more interactive, immersive, and useful in practical applications, e.g., gaming or simulation using Augmented Reality (AR) and Virtual Reality (VR). 3D mapping is fundamental for blending digital content with the real world seamlessly. With a proper 3D representation compatible with various hierarchies of resolutions and detail, encoding high-fidelity 3D information—where "high-fidelity" denotes a high degree of accuracy and realism in reproducing or simulating the real environment—digital 3D objects can be placed to interact naturally with physical spaces. This capability is further enhanced by recent advances in machine learning and deep generative models, which are trained on massive datasets, learning complex statistical distributions of real-world objects and scenes. This helps them to synthesize content that closely resembles real data and enables more realistic and context-aware spatial interactions.

In both physical-world reconstruction and digital content creation, the uncertainty of mapping is critically important. Understanding and managing the mapping uncertainty significantly impacts the effectiveness and safety of autonomous systems and digital systems. Misalignments or mapping errors in spatial understanding can lead to physical collisions or incorrect actions, potentially causing harm. Uncertainty quantification in 3D mapping provides a measure of confidence in the data and the models derived from it. By understanding the potential errors and the reliability of the 3D models, users can make informed decisions, particularly in scenarios where safety and precision are paramount.

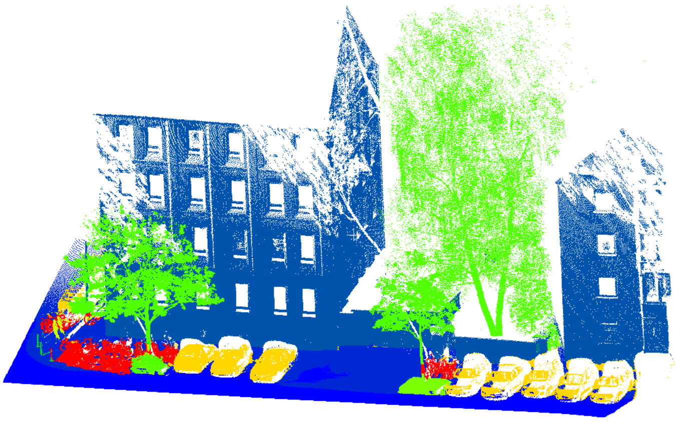


FIGURE 1.2. Typical LiDAR point clouds captured in urban environments: colors indicate the semantic labels

1.2 Motivation

As introduced, mapping plays a pivotal role in various disciplines, notably in the advancement of modern autonomous systems. These technologies heavily rely on maps that are both accurate and computationally efficient, enabling fast processing, low-latency decision-making, and safe operation in complex environments. However, mapping large urban areas accurately and efficiently remains a significant challenge.

This difficulty stems from several factors inherent to urban environments and the limitations of current sensor technology. Firstly, urban scenes are highly unstructured and heterogeneous, containing a diverse group of objects like buildings, vehicles, trees, and other infrastructural elements, as shown in Figure 1.2. Each component introduces variability that complicates the mapping process. Secondly, the vast amounts of data required to map large areas can lead to efficiency issues. Processing, storing, and analyzing these large datasets demand substantial computational resources and optimized algorithms to maintain workflow efficiency. Thirdly, commonly used sensors such as LiDAR are susceptible to various sources of noise, which can degrade the quality of the data. Factors like atmospheric conditions, sensor calibration errors, and technical limitations affect the accuracy of the measurements. In urban mapping, LiDAR and other sensing technologies often encounter occlusions caused by obstacles such as buildings or trees, which block the sensor's line of sight to certain areas. As illustrated in Figure 1.2, the building façades are occluded by trees in front of them. Additionally, the collected point clouds can be sparse, with uneven distribution of data points, especially in complex or cluttered areas.

The inherent challenges of mapping large urban environments invariably introduce imperfections into the resulting maps. These imperfections compromise not just the accuracy but also introduce potential spatial or semantic uncertainties in the reconstructed city models. A comprehensive mapping framework should prioritize not only accuracy and efficiency but also robustly address the uncertainties of mapping outputs. While plenty of the methods focus on efficiency and accuracy, the confidence information associated with the results, affected by data quality and the presence of uncertain or unclear information, should also be indicated. A model without a quality measure can be risky for safety-critical scenarios. Some popular mapping frameworks (Hornung et al. 2013; Mildenhall et al. 2021; Kerbl et al. 2023) emphasize accuracy and efficiency, while the uncertainty of mapping outputs is not sufficiently addressed. Since large errors can also appear in a generally accurate model due to data errors and incompleteness, it is necessary to quantify the uncertainty of the mapping results as well.

The uncertainties associated with maps also significantly impact the performance of downstream uncertainty-aware tasks, including localization, navigation, and collision avoidance of an autonomous system. When the map is treated as an error-free reference in localization and path-planning tasks without considering its quality, this can result in severe errors, e.g., the wrongly modelled occluded buildings might cause a bias in the estimated pose. Thus, an appropriate representation or quantification of the map uncertainties should be defined to help convey the information about the map quality to downstream applications.

Additionally, one may use multiple sensors to capture the environment, or may have the demand to update an existing map with new data. In those scenarios, a good uncertainty measure of the maps created by different sensors and agents is essential for data fusion and motion planning purposes (Y. Huang and Gupta 2008).

Although many existing studies on uncertainty estimation are found in the domain of autonomous driving, they primarily focus on uncertainties in pose estimation, with comparatively less attention given to the accuracy and precision of the generated maps. There is a lack of sufficient research describing the accuracy and precision of maps. Incorporating probabilistic uncertainty measures into maps is essential and holds significant potential to enhance performance in map-based localization and Simultaneous Localization and Mapping (SLAM).

Some existing methods have demonstrated that even simple representations of map uncertainty can improve localization accuracy or make the results more robust (Biber and Strasser 2003; Ehambram et al. 2022; Javanmardi et al. 2019). For example, Biber and Strasser 2003 proposed to represent the environment using Normal Distribution Transform (NDT) maps, where the map uncertainty is represented by a grid of distributions. However, it assumes independent discontinuous distributions of the neighboring cells and it has been found to result in a higher uncertainty at cell boundaries (Srivastava and Michael 2018). Javanmardi et al. 2019 introduced the idea of building abstract vectors and planar surface maps of buildings and ground, where the uncertainty is given by the normal distribution generated from vectors and planes. Building maps with fixed interval uncertainties has been used in hybrid interval-probabilistic localization (Ehambram et al. 2022). Occupancy maps that incorporate uncertain occupancy states are widely used in localization tasks, where uncertainty is explicitly represented through probabilistic occupancy values assigned to each cell.

To enable well-calibrated uncertainty estimation, where the predicted uncertainty accurately corresponds to the true likelihood of errors, probabilistic techniques such as Gaussian Mixture Model (GMM) and Gaussian Process (GP) can be integrated into spatial mapping frameworks. These methods support a probabilistic formulation of the 3D mapping process and facilitate the integration of Bayesian uncertainty quantification. Additionally, these models also possess continuous characteristics, allowing for smooth and resolution-independent representations of spatial structures. While several GMM- and GP-based mapping approaches have been proposed (Marriott et al. 2018; Srivastava and Michael 2018; O'Callaghan and F. T. Ramos 2012; J. Wang and Englot 2016; B. Lee et al. 2019), the inherent probabilistic properties and the associated output uncertainties have not been thoroughly explored (Pearson et al. 2022).

The existing studies prove the importance of accurately quantifying the uncertainty in mapping. Yet, it remains a substantial challenge to effectively map continuous spaces from noisy, incomplete point clouds while also assigning appropriate uncertainty measures. Despite its relevance, the topic of uncertainty in 3D mapping has not been sufficiently explored, revealing a clear research gap, particularly in the context of spatial uncertainty quantification for urban environments. There are many man-made structures with relatively regular geometrical shapes, where buildings are often the most important ones for localization, e.g., as used in the previous work (Javanmardi et al. 2019; Ehambram et al. 2022). Motivated by this, this dissertation begins with the development of an uncertainty-aware map representation tailored for buildings in urban settings and subsequently extends the model to accommodate more general structures.

Moreover, incomplete models resulting from partial or occluded measurements are a significant challenge in 3D mapping. Addressing this issue requires not only enhancing the reconstruction using prior knowledge of the environment but also quantifying the uncertainty associated with the reconstructed regions to maintain an uncertainty-aware mapping framework. With probabilistic mapping, those unknown regions can be inferred from the prior distribution. This capability further motivates the adoption of probabilistic methodologies for more informative 3D map representations.

Recent developments in 3D geometry generation underscore the potential of a 3D mapping representation that integrates effectively with generative models. Such models not only aid in enhancing 3D reconstruction from partial incomplete data by leveraging prior information provided by generative models but also improve the fidelity of simulations and virtual content generation. These advantages highlight the potential of exploring novel 3D mapping frameworks in combination with deep generative models, representing a promising research direction with substantial implications for both real-world applications and immersive virtual environments.

1.3 Goal and Contributions

To address the above-mentioned issues, this dissertation aims to model geo-objects in large-scale urban environments with a focus on accuracy, efficiency, and uncertainty quantification. Specifically, continuous 3D models and reconstructions are developed using probabilistic methodologies, enabling the representation of spatial structures alongside well-calibrated measures of uncertainty. Building upon this foundation, the dissertation also explores the use of the proposed map representation for uncertainty-aware localization and 3D geometry generation.

To achieve these objectives, the framework leverages both GMMs and GPs as tools for probabilistic inference. GPs are widely recognized for their ability to provide reliable uncertainty estimates through posterior inference; however, their scalability is limited, particularly when applied to large datasets typical of urban mapping. To address this, a hybrid approach is introduced, which incorporates informative priors derived from parametric probabilistic models—specifically, GMMs—to reduce the training complexity and enhance computational efficiency, while preserving well-calibrated uncertainty in the inference process.

The resulting framework integrates GPs and GMMs to construct a 3D Gaussian map from LiDAR point cloud data in large, static urban scenes. The method is applied to a range of urban structures, from man-made façades to more general surface geometries. The proposed representation is evaluated and benchmarked against state-of-the-art methods on both real-world and synthetic datasets, demonstrating competitive performance.

A central contribution of this work is its emphasis on the uncertainty of the mapping outputs. The framework provides uncertainty estimates that serve as indicators of model confidence, allowing unreliable regions, particularly those with sparse or occluded measurements, to be identified and treated accordingly. For such unobserved areas, the probabilistic inference enables coarse Gaussian estimations with high uncertainty, which can serve as a warning signal for downstream applications such as localization or path planning.

Building on the 3D Gaussian representation, the dissertation further investigates two key applications: uncertainty-aware localization and probabilistic 3D geometry generation. In both cases, the proposed mapping framework demonstrates its potential to support robust inference in the presence of uncertainty. In the case of geometry generation, the integration with deep generative models enables uncertainty-informed surface simulation and highlights the plausible reconstructions of missing geometry, with the potential utility of the approach for partial point cloud completion.

1.4 Structure of the Dissertation

Following the presentation of the motivation and objectives in this introductory chapter, the structure of the dissertation is outlined as follows:

- **Chapter 2: Theoretical Basics** - This chapter lays the groundwork for our study by discussing the theoretical concepts pertinent to our research. Topics covered include standard mapping representations, probabilistic methods, the fundamentals of localization, and an introductory overview of diffusion models.

- **Chapter 3: Review of Related Work** - This chapter examines existing studies relevant to our work, focusing on advancements in 3D mapping, localization techniques, and the generation of 3D point clouds.

- **Chapter 4: Datasets and Data Preparation** - This chapter introduces the datasets used in the subsequent experiments and details the corresponding data preprocessing procedures..

- **Chapter 5: Uncertain building models using GMM and GP** - This chapter introduces uncertain building models employing GMM priors and GP framed with local 2.5D frames.

- **Chapter 6: Uncertain distance field modeling using hierarchical GMM and GP** - This chapter extends the previous building models introduced in Chapter 5, employing 4D Hierarchical GMM priors and GP inferences with derivative observations. This approach is used to model uncertainty-aware 3D maps for more complex shapes.

- **Chapter 7: Application in Uncertainty-Aware Localization** - This chapter illustrates how the proposed methods are applied to enhance localization, the corresponding uncertainty propagation from map to pose estimation is also discussed.

- **Chapter 8: Application in 3D Geometry Generation Using Gaussian Maps** - This chapter explores the application of the 3D Gaussian map representations in generating 3D geometries based on diffusion models.

- **Chapter 9: Conclusions and Outlook** - The dissertation concludes with a summary of the findings and a discussion of potential directions for future research.

Chapter 2

Theoretical Basics

2.1 Map Representations

A map is a structured representation of the spatial properties of an environment, capturing geometric, topological, and semantic information to support tasks such as geospatial analysis, localization, navigation, perception, and decision-making. Map representations serve as structured models that facilitate the visualization, analysis, and interpretation of spatial data. In this section, the commonly used map representations in the field of autonomous systems are introduced, including 3D point clouds, vector-based boundary representations, occupancy maps, implicit surface maps, and explicit mesh surfaces.

2.1.1 Point Clouds

Point clouds are a collection of data points defined in a three-dimensional coordinate system. Each point in a point cloud is expressed as x, y and z. These points represent the external surface of an object or a scene and are typically generated by 3D scanners, such as LiDAR, camera images, or other remote sensing technologies. Point clouds are fundamental in various fields, including computer graphics, robotics, geospatial analysis, and virtual reality, providing detailed representations of real-world environments.

Point clouds are characterized by several key attributes, with density, intensity, and color being the most significant. Density, determined by the spacing between points, directly influences both the detail captured in the data and the overall size of the data file—higher density provides more detail but increases file size. Intensity measures the strength of the sensor's return signal and is typically available from LiDAR data, indicating surface reflectivity and texture. Color information, usually represented in RGB values, is generally acquired from photographic sources (e.g., camera images) and enriches visual fidelity. These attributes constitute the fundamental geometric and radiometric properties of the point cloud, which in turn dictate its utility for subsequent processing and analysis.

2.1.2 Boundary Representation

Boundary representation is a geometric modeling approach that defines a 3D object by explicitly describing its boundaries, such as surfaces, edges, and vertices, rather than its interior volume. In these models, objects are typically represented as a collection of polygonal faces (often triangles or quadrilaterals) that form closed surfaces enclosing the object. This representation is particularly well-suited for man-made environments where geometries are structured and well-defined.

In the context of urban modeling, CityGML is a widely adopted standard that uses a boundary representation to encode the geometric, semantic, and topological information of city objects such as buildings, roads, and vegetation. CityGML supports multiple Levels of Detail (LoD), allowing models to scale from simple block models (LoD1) to

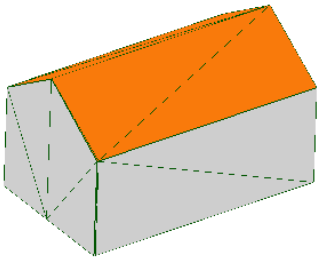


FIGURE 2.1. LoD2 with roof surface (orange) and wall surface (grey)

detailed architectural representations including façades and roofs (LoD2 and above) (Kolbe 2009). Figure 2.1 illustrates an example of boundary representation, showing the semantic components of a building, with the roof surface shown in orange and the wall surface in grey.

Boundary representation in standards like CityGML offers several key advantages. It enables accurate reconstruction of object geometries with explicit surface definitions and supports rich semantic information by linking geometric components to semantic classes (e.g., roof surface, wall surface). Additionally, as an XML-based standard, CityGML ensures high interoperability across software platforms and supports hierarchical structuring of urban elements, facilitating scalable and semantically meaningful urban modeling.

However, boundary representation models typically do not encode uncertainty or volumetric information, and they require topologically consistent data, which can be a limitation in noisy or incomplete reconstructions.

2.1.3 Occupancy Map

Occupancy maps are a fundamental mapping representation used in robotics and autonomous systems, particularly for applications involving navigation, mapping, and environment understanding. These maps provide a probabilistic framework to represent the presence or absence of obstacles in an environment, facilitating safe and efficient movement for robots.

An occupancy map is defined as a grid (2D)- or voxel (3D)-based representation of a space where each cell in the grid contains a probability that specifies whether a particular area is occupied by an obstacle or is free. The probability ranges typically from 0 (free) to 1 (occupied), allowing a robot to make decisions based on it.

An occupancy map is often generated by ray tracing, incorporating sensor data, usually from LiDAR, sonar, or stereo camera. The scanner and sonar emit the rays in various directions and can measure the distance to the nearest obstacle in the direction of the ray. As each ray travels through space, it may either hit an obstacle or travel up to its maximum range without encountering any. The point where a ray hits an obstacle is marked as occupied, while cells traversed by these rays are seen as free space. Although stereo cameras do not actively emit rays, they estimate a depth map that can be reprojected into 3D space to form a point cloud. Similar to LiDAR, each point is considered a "hit" (occupied cell), and the space along the ray from the camera origin to the point is considered "free".

Figure 2.2 presents a 2D illustration of an occupancy map for clarity. In this example, the blue element represents a sensor, and the green lines denote the emitted scanning rays. Cells traversed by these rays are shown in white, indicating free space (with an occupancy value of 0), while cells where the rays terminate—suggesting contact with an object—are shown in red, indicating occupied space (with an occupancy value of 1).

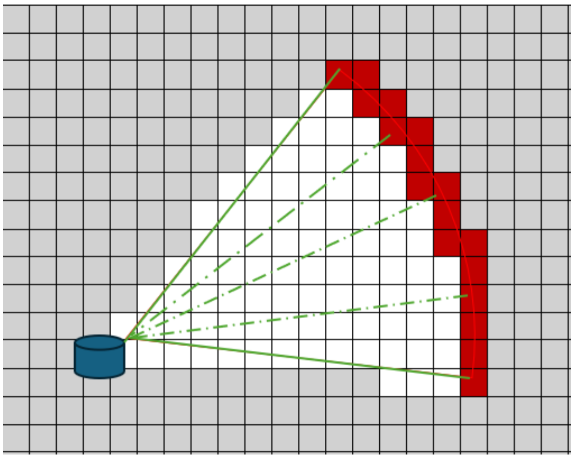


FIGURE 2.2. Occupancy map

In probabilistic occupancy maps, these binary values are replaced with occupancy probabilities ranging continuously from 0 to 1. For instance, a cell passed through by a ray might be assigned a low occupancy probability (e.g., 0.3), reflecting a high likelihood of being free, whereas a cell where a ray terminates may be assigned a high occupancy probability (e.g., 0.9), indicating it is likely occupied. This probabilistic representation provides a more nuanced and robust modeling of the environment, especially in the presence of sensor noise and partial observations.

The occupancy map can be incrementally updated as new sensor data becomes available. Each cell in the path of a ray is updated based on whether the ray passes through it freely or ends in it. The **Bayesian update rule** is used to compute the probability that a cell is occupied given the current measurement and the previous state's occupancy probability. The updating equation is given by:

$$P(o|z, X) = \frac{P(z|o, X) \cdot P(o)}{P(z|X)}, \tag{2.1}$$

where:

- $P(o|z, X)$ is the posterior probability of the cell being occupied, given the sensor measurement z and the map state X.
- $P(z|o, X)$ is the likelihood of observing z, assuming the cell is occupied.
- $P(o)$ is the prior probability of the cell being occupied.
- $P(z|X)$ is the normalizing constant that ensures the probabilities sum to 1.

This equation is used to update the occupancy probability based on new sensor data, reflecting changes in the environment.

In practice, the occupancy map utilizes the **log-odds** formulation to make the computation more numerically stable and efficient. By transforming occupancy probabilities $p \in [0, 1]$ into the logarithmic domain, the method operates in an unbounded real-valued space, mitigating floating-point precision artifacts that arise when probabilities approach 0 or 1. Also, Bayesian updates simplify to addition with the log-odds, replacing multiplicative probability updates with faster, more stable log-likelihood accumulation. The log-odds l of a cell being occupied is updated as follows:

$$l_t(o) = l_{t-1}(o) + \log\left(\frac{P(z_t|o)}{P(z_t|\neg o)}\right) - l_0(o) \tag{2.2}$$

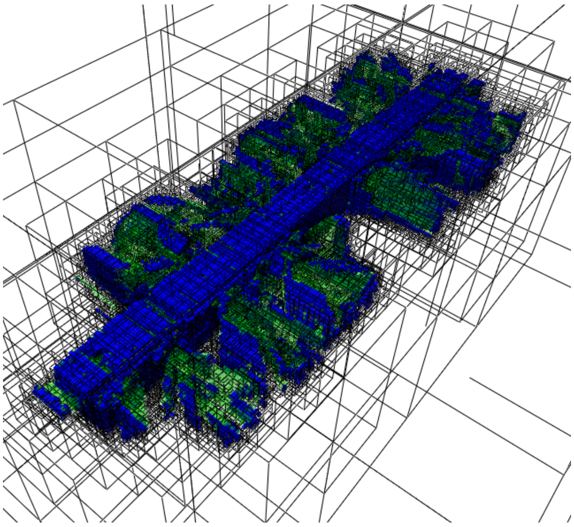


FIGURE 2.3. An example of the OctoMap (Hornung et al. 2013)

where:

- $l_t(o)$ represents the log-odds of occupancy at time t,
- $l_{t-1}(o)$ is the log-odds of occupancy at the previous time step $t-1$,
- $\log\left(\frac{P(z_t|o)}{P(z_t|\neg o)}\right)$ is the log-odds update resulting from new measurement z_t,
- $l_0(o)$ is the initial log-odds value, typically based on the prior probability of occupancy.

Octree-based Occupancy Map (OctoMap) (Hornung et al. 2013) stands out among occupancy mapping techniques, primarily due to its enhanced efficiency and reduced storage requirements compared to the original uniform grid version. It utilizes an octree, a tree data structure where each node has exactly eight children. This structure is effective for 3D space partitioning because it divides the space recursively into eight octants. At each level of the tree, the space is divided into smaller cubes (voxels), allowing for progressively finer resolutions. This hierarchical structure can be exploited to adjust the resolution depending on the need, such as higher resolution in areas of interest and lower resolution elsewhere. This enables OctoMap to efficiently handle large datasets with improved precision, while optimizing both memory usage and computational resources.

Figure 2.3 illustrates an example of an OctoMap, where blue voxels represent occupied space and black wireframes indicate the hierarchical structure of the underlying octree. Regions with a higher density of measurements or larger variation in occupancy states are recursively subdivided into smaller voxels, allowing for adaptive resolution and more detailed representation where needed.

OctoMap's capacity to efficiently generate detailed and updatable 3D maps makes it a powerful tool for any application requiring a detailed understanding of complex environments.

2.1.4 Implicit Surface Map

Implicit surface maps represent surfaces in a three-dimensional space not through explicit coordinates of surface points but through a function whose zero-crossing set defines the

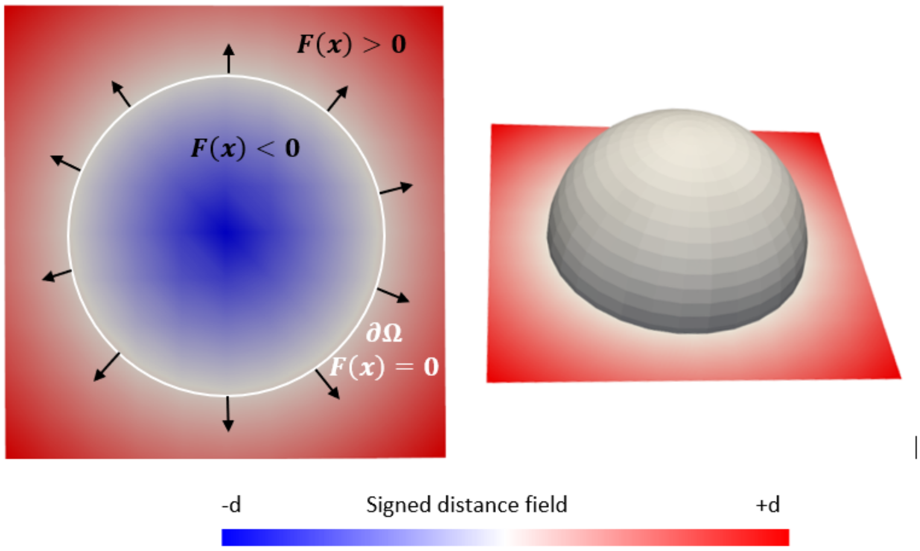


FIGURE 2.4. SDF of a sphere

surface. This approach offers unique advantages in terms of flexibility, continuity, and the ease of merging multiple surfaces.

An implicit surface in three-dimensional space is typically defined by an equation of the form: $F(\boldsymbol{x}) = 0$ where F is a scalar field defined over $\boldsymbol{x} \in \mathbb{R}^3$. The set of all points $(\boldsymbol{x})$ satisfying this equation constitutes the implicit surface. Unlike explicit representations that directly list vertex coordinates (like polygon meshes), implicit surfaces are defined by their relationship to a function.

A popular representation is the Signed Distance Field (SDF), where the function $F(\boldsymbol{x})$ represents the shortest distance to the surface. Figure 2.4 illustrates an example of a SDF for a sphere, where $F(\boldsymbol{x})$ denotes the distance field. The blue regions inside the circle represent negative SDF values, corresponding to space inside the sphere, while the red regions outside indicate positive values for space outside the object. The magnitude of the distance field increases smoothly with the distance to the nearest surface. The white circle marks the zero-level set, corresponding to the object's surface where the signed distance is exactly zero. Mathematically, it is defined by the following functions:

$$d = F(\boldsymbol{x}) \begin{cases} = 0 & : \text{ on the surface,} \\ < 0 & : \text{ inside object,} \\ > 0 & : \text{ outside object.} \end{cases} \tag{2.3}$$

where d is the signed distance. Positive distances $(F(\boldsymbol{x}) > 0)$ indicate the points are outside the surface, while negative values $(F(\boldsymbol{x}) < 0)$ define the interior volume, denoted as Ω. The zero-value distance denotes the boundary $\partial\Omega$ of the object's surface:

$$\partial\Omega = \{\boldsymbol{x} : F(\boldsymbol{x}) = 0\} \tag{2.4}$$

It should be noted that in scenarios where the sign information is disregarded, one obtains the Euclidean Distance Field (EDF) as a result. EDF defines spatial relationships purely based on the magnitude of distances, without incorporating the directional or inside-outside context provided by signed distance measures.

Implicit surfaces inherently provide smooth and continuous representations without requiring high-resolution grids, which is beneficial in applications needing fluid and organic shapes. This smoothness comes from the mathematical properties of the functions defining the surfaces, typically facilitating easier generation of smooth transitions and deformations. Implicit surface equations can compactly represent complex shapes with relatively simple mathematical expressions. This compactness often results in reduced storage requirements compared to discrete grids or explicit mesh representations. Additionally, the use of mathematical functions in implicit surface representations provides valuable analytical properties, such as normals and curvatures, which can be derived directly from the equations. This capability is particularly useful in scientific simulations where these properties are required for computations.

2.1.5 Explicit Mesh Surface

An explicit mesh is a representation of a three-dimensional shape through discrete geometric elements like vertices, edges, and faces, commonly used in computer graphics. This method directly specifies the shape by defining its surface as a collection of polygons, making it particularly suitable for rendering, simulation, and physical analysis. The most common type of explicit mesh is the triangular mesh, where the basic surface element is a triangle. This format is favoured due to its computational simplicity and flexibility in handling complex surfaces. Compared to implicit surface representations, explicit meshes directly outline the shape's surface, making them intuitive and straightforward for visualization and interaction. Manipulating the shape by adjusting the vertices, edges, or faces is relatively straightforward, facilitating tasks like editing, deformation, and simulation.

While explicit meshes are effective for representing geometric surfaces, they also pose several challenges that require careful management. Firstly, high-resolution meshes can be computationally intensive, requiring substantial memory and processing power. Secondly, the property of the mesh (e.g., element shape, size, and distribution) significantly affects the outcomes of simulations and analyses. Poorly constructed meshes can lead to unreliable or inaccurate results. Additionally, different systems and applications may use various mesh formats, requiring robust conversion tools to ensure compatibility.

Marching cubes is a widely used computer graphics algorithm, which constructs an explicit polygonal mesh from implicit surface fields in 3D. It was developed by William E. Lorensen and Harvey E. Cline in 1987 and has since become a fundamental technique in the field of computer graphics for rendering complex surfaces. The marching cubes algorithm works by dividing the volume of data into a discrete grid of cubes. Each corner of these cubes (also referred to as vertices) can have one of two possible states: inside or outside the isosurface, often determined by whether the data value at that vertex is larger than or smaller than a user-defined threshold (the iso-value).

The process includes cube classification, edge interpolation, triangulation and mesh generation.

- Cube classification: Each cube in the grid is examined independently. The state of the cube's eight vertices (inside or outside the isosurface) determines its configuration. There are 256 possible configurations (2^8, since each vertex has two possible states), but due to symmetry and rotation, this can be reduced to 15 unique cases by the original marching cubes algorithm.

- Edge interpolation: The algorithm interpolates the edges of each cube to find the exact points where the surface intersects the cube. If a cube's edge connects an "inside" vertex to an "outside" vertex, the surface must intersect that edge. The exact point of intersection is estimated based on the scalar values at the cube's vertices, usually by linear interpolation.

- Triangulation: Once the intersection points are determined, the cube is triangulated by creating polygons (triangles) from these points. The specific triangulation for a cube depends on its configuration.
- Mesh generation: Triangles generated from all the cubes are combined to create the final mesh that approximates the isosurface.

2.2 Probabilistic Uncertainty

Probabilistic uncertainty quantification focuses on quantifying the uncertainty inherent in predictions and models through probabilistic means, which is vital for making informed decisions under uncertainty. It involves using probability distributions to describe and analyze the uncertainty in model outputs, parameters, or predictions. This contrasts with deterministic methods, which provide specific values or outcomes without accounting for variability. A key technique in uncertainty quantification is the use of Bayesian statistical methods, which offer a structured probabilistic framework that integrates prior knowledge with empirical data, facilitating dynamic updating and refinement of uncertainty models as new information becomes available. In practice, with probabilistic uncertainty, it is possible to derive confidence intervals, which quantify the range within which the true value of a variable is likely to lie with a specified level of confidence.

In the context of mapping in this work, techniques such as Bayesian inference, Multivariate Gaussian distributions, Gaussian Mixture Models, and Gaussian Processes are employed to quantify uncertainty within a Bayesian framework. These methods enable predictive modeling while improving the reliability and robustness of spatial analysis. This section introduces these techniques as follows.

2.2.1 Bayesian Inference

Bayesian inference is a statistical method that applies the principles of probability to update an uncertain estimation based on observing new data. It is based on Bayes' Theorem, which describes the probability of an event based on prior knowledge of conditions that might be related to the event. Bayes' Theorem is the cornerstone of Bayesian inference, and it mainly consists of prior probability, evidence, likelihood, and posterior probability. The update rule reads:

$$p(\theta|\boldsymbol{D}) = \frac{p(\boldsymbol{D}|\theta) \cdot p(\theta)}{p(\boldsymbol{D})} \tag{2.5}$$

where $p(\theta)$ is the prior probability of the target model parameters θ, which represents the belief about θ before observing the data. $p(\boldsymbol{D})$ is the evidence of the observed data. $p(\boldsymbol{D}|\theta)$ is the likelihood of θ, also known as the conditional probability of observing the data given the model parameters θ. $p(\theta|\boldsymbol{D})$ is the updated posterior probability of the parameters θ given data $\boldsymbol{D}$.

The prior probability distribution reflects what is known about the parameters before any data is observed. This can be based on previous experiments, or expert knowledge, or could be non-informative if little is known a priori. Unlike probability, which measures how probable a future outcome is under certain conditions, likelihood measures how plausible a particular parameter set is for observed samples. It is central to fitting models to data in Bayesian inference. Note that we emphasize that the likelihood function $p(\boldsymbol{D}|\theta)$ is not a probability distribution over θ and is not normalized.

The posterior distribution combines the likelihood function with the prior distribution. In the context of uncertainty estimation, the posterior probability captures the model's state of knowledge: The more “narrow" this distribution, i.e., the more concentrated the probability mass in a small region, the less uncertain the model is.

Bayesian inference often involves calculating the expected values of statistics or making decisions based on the posterior distribution. This can include prediction, estimating parameters, or comparing different models. For a given query sample $\boldsymbol{x}_*$, we average over all possible predictive parameter values, weighted by their posterior probability. Thus, the predictive posterior distribution of the target value y_* reads:

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{D}) = \int p(y_*|\boldsymbol{x}_*, \theta)p(\theta|\boldsymbol{D})d\theta \tag{2.6}$$

This is in contrast to non-Bayesian schemes, where a single parameter estimation is typically chosen.

2.2.2 Multivariate Gaussian Distribution

The Gaussian distribution, also referred to as the normal distribution, is one of the most important probability distributions in statistics due to its widespread applications across various scientific and engineering disciplines. It is characterized by its "bell-shaped" curve, which is symmetrical about the mean. A univariate Gaussian distribution is fully described by two parameters: the mean (μ) and the variance (σ^2). The Probability Density Function (PDF) for a single variable x is defined by:

$$\mathcal{N}(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{2.7}$$

In the case of multivariate Gaussian, the variable is a d-dimensional random vector and the PDF for the Multivariate Normal Distribution (MVN) is a bit more involved than the univariate case due to the inclusion of a covariance matrix. The distribution can be written in the form:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.8}$$

where:

- $\boldsymbol{\mu}$ is the d-dimensional mean vector,
- $\boldsymbol{\Sigma}$ is the $d \times d$ covariance matrix, and
- $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

The normalization constant $\frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}$ in Equation (2.8) ensures that the density function integrates to 1.

The Gaussian distribution possesses many important analytical properties, some of which are applied to our work that will be introduced in the subsequent chapters. Thus, we explain these properties in detail here.

We start with the geometric property of the Gaussian distribution, i.e., the geometric shape of the level sets. The level sets can be seen as density contours of the PDF, where the density function values $p(x)$ of all samples in a certain set equal a constant value.

As shown in Equation (2.8), the probability density function depends on the **Mahalanobis distance** Δ between $\boldsymbol{x}$ and $\boldsymbol{\mu}$, through the form:

$$\Delta = \sqrt{(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2.9}$$

where:

- $\mathbf{\Sigma}^{-1}$ is the inverse of the covariance matrix.

It can be observed that the Gaussian probability density will be constant on surfaces in **X**-space where this Mahalanobis distance is constant, i.e., the contours/surfaces of probability density are equivalent to the contours/surfaces of constant Mahalanobis distance. Therefore, the shape of level sets is dependent on the covariance matrix Σ.

Now since Σ is a symmetric positive definite matrix, its eigendecomposition yields real, positive eigenvalues and mutually orthogonal eigenvectors. This relationship can be mathematically expressed as follows:

$$\Sigma = Q\Lambda Q^{-1} = Q\Lambda Q^{\top} \tag{2.10}$$

where:

- Q is the matrix of eigenvectors of Σ.
- Λ is a diagonal matrix whose diagonal elements are the eigenvalues of Σ.

The Equation (2.10) can be further rewritten as:

$$\Sigma = \sum_{i=1}^{D} \lambda_i \mathbf{q_i} \mathbf{q_i}^{\top}, \tag{2.11}$$

where:

- $\mathbf{q_i}$ is the i-th column of Q, denoting the i-th normalized eigenvector.
- λ_i is the i-th diagonal element of Λ, denoting the i-th eigenvalues of Σ.

Thus, the inverse of the Σ is:

$$\Sigma^{-1} = Q\Lambda^{-1}Q^{\top} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{q_i} \mathbf{q_i}^{\top}. \tag{2.12}$$

Substituting Equation (2.12) into (2.9), we can write the squared Mahalanobis distance as

$$\Delta^2 = \sum_{i=1}^{D} \frac{1}{\lambda_i} (\boldsymbol{x} - \boldsymbol{\mu})^{\top} \mathbf{q_i} \mathbf{q_i}^{\top} (\boldsymbol{x} - \boldsymbol{\mu}) \tag{2.13}$$

$$= \sum_{i=1}^{D} \frac{1}{\lambda_i} x_i'^2, \tag{2.14}$$

where:

- $x_i' = (\boldsymbol{x} - \boldsymbol{\mu})^{\top} \mathbf{q_i} = \mathbf{q_i}^{\top} (\boldsymbol{x} - \boldsymbol{\mu})$ represents the i-th dimension of a new transformed Euclidean coordinates. In the coordinate frame, the axes are defined by the orthogonal eigenvectors with the origin at μ.

Given a constant Δ, one can derive the level sets from the Equation (2.14). For Gaussian distributions, these level sets manifest themselves as ellipses in 2D and ellipsoids in 3D, corresponding to the definition equations of ellipses and ellipsoids. The principal axes of these ellipses/ellipsoids align with the eigenvectors of the covariance matrix, and their radii are proportional to the square roots of the respective eigenvalues.

Conditional and marginal rules are two important properties of Multivariate Gaussian distributions that enable them to work in many tasks, for example, regression. Both the conditional and marginal distributions of a subset of variables from a multivariate Gaussian are also themselves Gaussian.

Suppose there is a joint Gaussian composed of two variables $\mathbf{X} = \{x_1, x_2\}$, the original mean vector is then:

$$\mu = \{\mu_1, \mu_2\}, \tag{2.15}$$

and the covariance matrix is given:

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{bmatrix}. \tag{2.16}$$

The probability density of the marginal distribution of one variable x_1 is defined as:

$$p(x_1) = \int p(x_1, x_2) dx_2 \tag{2.17}$$

Its mean and covariance matrix are the corresponding subset and submatrix of the original mean vector and covariance matrix respectively, i.e., μ_1 and $\mathbf{\Sigma}_{11}$, which is the marginal distribution of x_1.

The conditional distribution of the variable x_1 given the other vector x_2 in a multivariate Gaussian can be represented as $x_{1|2} \sim \mathcal{N}(\mu_{1|2}, \mathbf{\Sigma}_{1|2})$, with the parameters as:

$$\mu_{1|2} = \mu_1 + \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{\mathbf{22}}^{\mathbf{-1}}((x_2 - \mu_2) \tag{2.18}$$

$$\mathbf{\Sigma}_{1|2} = \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{\mathbf{22}}^{\mathbf{-1}}\mathbf{\Sigma}_{21}. \tag{2.19}$$

If we denote the corresponding precision matrix Λ:

$$\mathbf{\Lambda} = \mathbf{\Sigma}^{\mathbf{-1}} = \begin{bmatrix} \mathbf{\Lambda}_{11} & \mathbf{\Lambda}_{12} \\ \mathbf{\Lambda}_{21} & \mathbf{\Lambda}_{22} \end{bmatrix}, \tag{2.20}$$

The conditional rule can be expressed as:

$$\mu_{1|2} = \mu_1 - \mathbf{\Lambda}_{\mathbf{11}}^{\mathbf{-1}}\mathbf{\Lambda}_{\mathbf{12}}(x_2 - \boldsymbol{\mu}_{\mathbf{2}}), \tag{2.21}$$

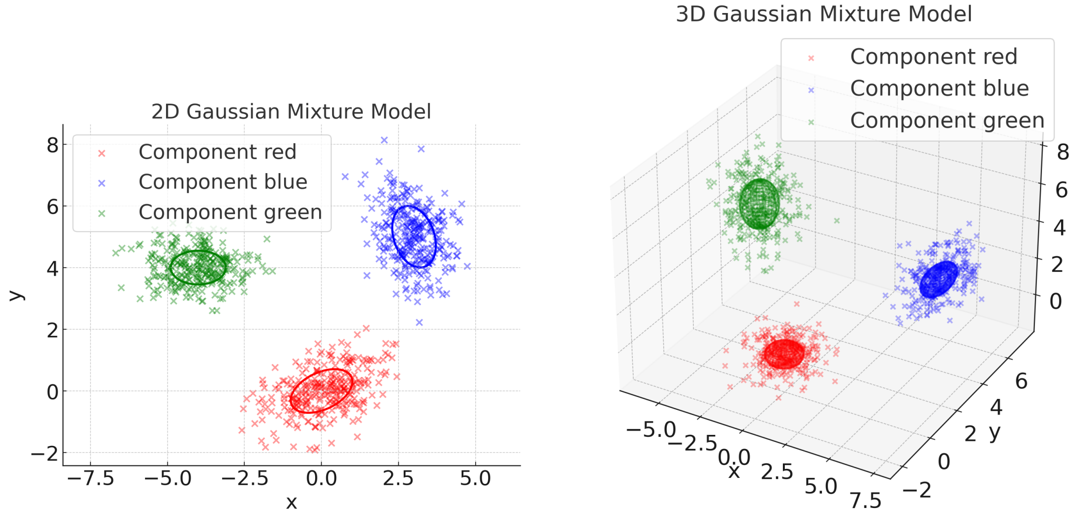
$$\mathbf{\Sigma}_{1|2} = \mathbf{\Lambda}_{\mathbf{11}}^{\mathbf{-1}}. \tag{2.22}$$

It can be easily observed that the conditional mean is a linear function of the other variable x_2, and the conditional covariance is a constant matrix that is independent of the conditioned variable x_2.

2.2.3 Gaussian Mixture Model

While the Gaussian distribution offers notable analytical properties, it often falls short in modelling complex real-world data sets due to its simplicity. However, by employing a GMM, which combines a sufficient number of Gaussian distributions, nearly any continuous density can be approximated with high accuracy. GMMs are particularly useful in applications like clustering, density estimation, and as components of more complex probabilistic models involving latent variables.

Figure 2.5 illustrates the GMM examples for clustering in 2D and 3D. Figure 2.5a shows a mixture of three 2D Gaussian components, each represented by colored scatter points and the ellipses representing the 1-standard-deviation contour of each Gaussian component's covariance matrix. Each ellipse encodes the mean (center) and covariance



(a) 2D GMM ellipses

(b) 3D GMM ellipsoids

FIGURE 2.5. GMM examples

(shape and orientation) of one Gaussian distribution. Likewise, Figure 2.5b visualizes a mixture of three 3D Gaussian distributions. The center location, spatial orientation and elongation of ellipsoids reflect the Gaussian mean and covariance.

GMMs assume that the data originate from several Gaussian sources, and each component in a GMM is defined by its unique parameters - mean, covariance, and mixture coefficient, allowing for precise adjustments to fit diverse datasets. The probability distribution function of a standard GMM is described by a set of parameters $\boldsymbol{\theta} = \{(\pi_k, \boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})\}_{k=1}^{K}$, as written in:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}), \tag{2.23}$$

$$\sum_{k=1}^{K} \pi_k = 1, 0 \leq \pi_k \leq 1 \tag{2.24}$$

where π_k is the mixing weight of the k-th component, which affects the contribution of each Gaussian component to the overall mixture; $\boldsymbol{\mu_k}$ and $\boldsymbol{\Sigma_k}$ are the mean and covariance of the k-th Gaussian. Given a perfect K, the components will properly approximate the object density.

To view GMM in a Bayesian formulation, a latent variable $\mathbf{z}$ is introduced and the GMM is interpreted as the marginal distribution $p(\boldsymbol{x})$ considering a joint distribution of both samples and latent variables $p(\boldsymbol{x}, \mathbf{z})$.

The latent variable $\mathbf{z}$ denotes which Gaussian component originates the sample. With K Gaussian components, $\mathbf{z}$ is a K-dimensional binary random variable, each k-th dimensional z_k is either 0 or 1, i.e., $z_k \in \{0, 1\}$ and only one dimension equals to 1, and all others are 0:

$$z_k = \begin{cases} 1 & : k = k^* \\ 0 & : k \neq k^* \end{cases}, \tag{2.25}$$

$$\sum_{k}^{K} z_k = 1 \tag{2.26}$$

where k^* denotes the particular dimension that equals 1 in a latent variable. There is a probability of each component being k^*, i.e., $p(z_k = 1)$. It quantifies the relative contribution of each k-th Gaussian component in the mixture, and namely, it is the mixing weight π_k of the GMM. Therefore, $p(z_k = 1)$ can be expressed mathematically as:

$$p(z_k = 1) = \pi_k, \tag{2.27}$$

$$p(\mathbf{z}) = \prod_{k=1}^{K} p(z_k = 1)^{z_k} = \pi_k \tag{2.28}$$

where $p(\mathbf{z})$ represents the prior probability of the latent variable. $p(\boldsymbol{x} \mid z_k = 1)$ is a conditional probability, denoting that once the variable $\mathbf{z}$ is given, the distribution is a single Gaussian component. Thus, the GMM density function, as the marginal distribution $p(\boldsymbol{x})$ of a joint distribution $p(\boldsymbol{x}, \mathbf{z})$, can be written as:

$$p(\boldsymbol{x}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\boldsymbol{x}|\mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.29}$$

Now we consider the question: how can we optimize the parameters of the GMM to represent the data distribution properly? In a similar case of a single Gaussian, Maximum Likelihood Estimation (MLE) is utilized to determine the optimal parameters. The likelihood function - discussed previously in Section 2.2.1, is the probability of the observed data given the parameters $\boldsymbol{\theta}$. For a unimodal Gaussian, they are denoted as $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\Sigma})$. If we assume each sample in data $\mathbf{X}$ is independent, the parameters can be optimized as follows:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \; p(\mathbf{X} \mid \boldsymbol{\theta}) \tag{2.30}$$

$$p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i}^{N} p(\boldsymbol{x}_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i}^{N} \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu})\right) \tag{2.31}$$

where

- $\boldsymbol{x}_i$ is the i-th sample of the data.
- d is the dimensionality of the data.
- N is the number of total samples.

To enhance computational efficiency and avoid the challenges associated with the cumulative multiplication of probabilities, the optimization of the likelihood function in exponential models is commonly transformed into a log-likelihood maximization problem:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \; \log p(\mathbf{X} \mid \boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \; \sum_{i}^{N} \log p(\boldsymbol{x}_i \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.32}$$

However, in the context of a Gaussian mixture model, the presence of multiple Gaussian components, each associated with a distinct weighting factor, introduces complexity to the optimization process. The MLE optimization technique has to solve the logarithm of a sum:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}} \; \sum_{i}^{N} \log \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2.33}$$

The necessity to compute the logarithm of the weighted summation of these Gaussian components poses analytical challenges and can not be solved in closed form. The

Expectation-Maximization (EM) algorithm is typically used as a powerful tool for GMM parameter optimization.

The main idea behind EM is also maximizing the likelihood functions; however, an iterative scheme is employed to overcome computational challenges posed by the direct optimization of complex likelihood functions. The EM algorithm in the context of GMM involves two key phases: the Expectation step (E-step) and the Maximization step (M-step). These steps are iteratively repeated until convergence, typically determined when the change in the log-likelihood function or the parameter values falls below a predefined threshold.

During the Maximization step, the parameters are derived based on the previous Equation 2.33, by setting the derivatives of the log-likelihood with respect to these parameters to zero. After simplifying and reformulating the equations, the mean vector, the covariance matrix and the mixing weight can be succinctly represented as follows:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik}) x_i, \tag{2.34}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma(z_{ik})(x_i - \boldsymbol{\mu}_k)(x_i - \boldsymbol{\mu}_k)^T, \tag{2.35}$$

$$\pi_k = \frac{N_k}{N}, \tag{2.36}$$

and $\gamma(z_{ik})$ and N_k are calculated as:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{2.37}$$

$$N_k = \sum_{i=1}^{N} \gamma(z_{ik}) \tag{2.38}$$

where

- $\gamma(z_{ik})$ represents the "responsibility" that each Gaussian component k takes for each observation $\boldsymbol{x}_i$, reflecting how likely x_i is to come from the k-th component relative to all possible components in the model.
- N_k represents the effective number of data points assigned to the k-th component.

Note that, as addressed earlier, the above equations can not be calculated in closed form, since the responsibility $\gamma(z_{ik})$ is unknown and correlated to the other parameters.

In the iteration scheme, we can initialize the parameters, and the Expectation step estimates the expected responsibilities based on the current parameters, according to Equation 2.37. The M-step can then update the parameters based on the expected γ. This optimization process can be summarized as follows:

1. Initialization: the parameters $\boldsymbol{\theta} = \{(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^{K}$ are initialized. This can be done randomly or based on some heuristic, such as the results of a clustering algorithm like k-means.

2. **Expectation** step: calculate the posterior probability $p(z_k = 1 \mid \mathbf{X})$, which is also called responsibilities $\gamma(z_k, \boldsymbol{x}_i)$; it approximates the true probability distribution of latent variables $p(z_k = 1)$.

3. **Maximization** step: maximize the joint distribution $p(x, \mathbf{z})$ to optimize the parameters.
4. Convergence: the algorithm iterates between the E-step and M-step until the changes in parameters or the improvement in the log-likelihood are below some threshold.

Other than EM, an alternative approach to optimize the GMM parameters is to use a gradient-based method. These methods are advantageous, particularly when leveraging modern GPU-based automatic differentiation and optimization techniques for large datasets or complex models.

2.2.4 Gaussian Mixture Regression

Leveraging the conditional and marginal properties of the Multivariate Gaussian distribution, the joint Gaussian distribution can be incorporated into regression analysis. This approach can be further developed within the framework of GMM, resulting in what is known as Gaussian Mixture Regression (GMR). GMR is adept at modelling complex datasets characterized by multiple underlying trends, nonlinear relationships, or heterogeneous variances. This methodology allows for a sophisticated analysis that accounts for the intrinsic variability and potential non-linearity in the data, making it highly suitable for applications where traditional regression models might fail to capture the underlying dynamics of the system effectively.

Suppose there is a joint distribution of the variables $(\mathbf{X}, \mathbf{Y})$, if both variables are multivariate Gaussian distributions, the conditional distribution $p(\boldsymbol{y} \mid \boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{y|x}, \boldsymbol{\Sigma}_{y|x})$ is also multivariate Gaussian, and its mean and covariance matrix can be computed according to Equation 2.18 and 2.19. $\mathbf{X}$ is called input variable and $\mathbf{Y}$ is the target variable or response variable. $m(\boldsymbol{x}) = \boldsymbol{\mu}_{y|x}$ is the regressed target value, and $\boldsymbol{\Sigma}_{y|x}$ describes the uncertainty of the regression.

If we consider the distribution of $(\mathbf{X}, \mathbf{Y})$ as a mixture of Gaussians, the regression can be computed as the weighted sum of regressions of multiple components. The overall regression inherits the regressed results from each single Gaussian component. First, its k-th component as a single Gaussian is denoted by

$$\boldsymbol{\mu} = (\boldsymbol{\mu}_{kX}, \boldsymbol{\mu}_{kY}), \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{kX} & \boldsymbol{\Sigma}_{kYX} \\ \boldsymbol{\Sigma}_{kXY} & \boldsymbol{\Sigma}_{kY} \end{bmatrix}, \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}_{11} & \boldsymbol{\Lambda}_{12} \\ \boldsymbol{\Lambda}_{21} & \boldsymbol{\Lambda}_{22} \end{bmatrix} \tag{2.39}$$

Considering that a joint distribution can be seen as a product of two distributions, the mixture of the joint distribution is given

$$p(\boldsymbol{y}, \boldsymbol{x}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\mu}_{kY|X}, \boldsymbol{\Sigma}_{kY|X}) p(\boldsymbol{x}; \boldsymbol{\mu}_{kX}, \boldsymbol{\Sigma}_{kX}), \tag{2.40}$$

and the marginal distribution $p(\boldsymbol{x})$ can be written as

$$p(\boldsymbol{x}) = \int p(\boldsymbol{y}, \boldsymbol{x}) dy = \sum_{k=1}^{K} \pi_k p(\boldsymbol{x}; \boldsymbol{\mu}_{kX}, \boldsymbol{\Sigma}_{kX}). \tag{2.41}$$

Therefore, the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ reads

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x})} = \sum_{k=1}^{K} \underbrace{\frac{\pi_k p(\boldsymbol{x}; \boldsymbol{\mu}_{kX}, \boldsymbol{\Sigma}_{kX})}{\sum_{j=1}^{K} \pi_j p(\boldsymbol{x}; \boldsymbol{\mu}_{jX}, \boldsymbol{\Sigma}_{jX})}}_{w_k(x)} p(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\mu}_{kY|X}, \boldsymbol{\Sigma}_{kY|X}). \tag{2.42}$$

where

- $w_k(\boldsymbol{x}) = \frac{\pi_k p(\boldsymbol{x};\boldsymbol{\mu}_{kX},\boldsymbol{\Sigma}_{kX})}{\sum_{j=1}^{K}\pi_j p(\boldsymbol{x};\boldsymbol{\mu}_{jX},\boldsymbol{\Sigma}_{jX})}$ denotes the mixing weight of the k-th component of the GMR.

$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\mu}_{kY|X},\boldsymbol{\Sigma}_{kY|X})$ can be computed by the conditional rule, same as in the single Gaussian case:

$$\mu_{kY|X} = \boldsymbol{\mu}_{kY} - \boldsymbol{\Lambda}_{kYY}^{-1}\boldsymbol{\Lambda}_{kYX}(\boldsymbol{x} - \boldsymbol{\mu}_{kX}), \tag{2.43}$$

$$\Sigma_{kY|X} = \boldsymbol{\Lambda}_{kYY}^{-1} \tag{2.44}$$

where $\mu_{Y|X}$ can be denoted as $m_k(\boldsymbol{x})$, representing the expected target value for the queried variable $\boldsymbol{x}$, and $\Sigma_{Y|X}$ is the associated uncertainty.

According to Equation 2.42, given the input variable $\boldsymbol{x}$, the target variable $\boldsymbol{y}$ can be regressed as the expectation of the distribution $p(\boldsymbol{y}|\boldsymbol{x})$, which is the weighted sum of each component's regression $m_k(\boldsymbol{x})$. The regression function and the associated uncertainty (variance $v(\boldsymbol{x})$) are expressed as

$$m(\boldsymbol{x}) = \sum_{k}^{K} w_k(\boldsymbol{x}) m_k(\boldsymbol{x}), \tag{2.45}$$

$$v(\boldsymbol{x}) = \sum_{k}^{K} w_k(\boldsymbol{x})\{\Sigma_k(\boldsymbol{x}) + m_k(\boldsymbol{x})m_k(\boldsymbol{x})^{\mathbf{T}}\} - m(\boldsymbol{x})m(\boldsymbol{x})^{\mathbf{T}}. \tag{2.46}$$

Although the regression function $m(\boldsymbol{x})$ resembles a kernel estimator in form, a crucial distinction exists: the weight function $w_k(\boldsymbol{x})$ is determined by the components of a global Gaussian mixture model rather than by the local structure of the data. Consequently, GMR constitutes a global parametric model that incorporates nonparametric flexibility (Sung 2004). Additionally, the availability of the covariance within the regression model is particularly advantageous for tasks requiring uncertainty quantification.

2.2.5 Gaussian Process

A GP is another powerful, probabilistic model used in machine learning for regression and classification tasks. It is defined as a collection of random variables, any finite number of which have a joint Gaussian distribution. Essentially, a GP generalizes multivariate Gaussian distributions to infinite dimensionality, allowing for the modelling of complex data sets with potentially infinite observations. The strength of GPs lies in their ability to provide a predictive distribution (not just point estimates) for outputs, offering not only the mean prediction at any point but also the uncertainty of that prediction.

A GP model can be seen as an infinite multivariate Gaussian distribution, and the target values $\mathbf{Y}_\infty = (y_1, y_2, y_3, ..., y_i, ...)$ represent different dimensions of this distribution with infinite dimensionality. If N dimensional values $\mathbf{Y_N} = (y_1, y_2, y_3, ..., y_N)$ are given, and any other target dimensions are $\mathbf{Y}_* = (y_t, y_t + 1, y_t + 2, ...)$, the joint distribution can be expressed as

$$\begin{bmatrix} \mathbf{Y_N} \\ \mathbf{Y}_* \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \boldsymbol{\mu}_N \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_N & \boldsymbol{\Sigma}_{N*} \\ \boldsymbol{\Sigma}_{*N} & \boldsymbol{\Sigma}_* \end{bmatrix}) \tag{2.47}$$

The targets $\mathbf{Y}_*$ can be inferred using the conditional rule of the Gaussian distribution, resulting in the posterior probability :

$$p(\mathbf{Y}_* \mid \mathbf{Y_N}) \sim \mathcal{N}(\boldsymbol{\mu}_{*|N}, \boldsymbol{\Sigma}_{*|N}). \tag{2.48}$$

In the context of GP, the target variable $\mathbf{Y}$ is defined over a continuous input space $\mathbf{X}$, e.g., over time or spatial coordinates, with $y = f(\boldsymbol{x})$ representing the corresponding target values at the input points. Both the mean and covariance in Equation 2.47 are functions defined over these input variables.

The GP model is thus formulated as $f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$, representing a distribution of functions. It is fully defined by two key functions: a mean function and a covariance function (kernel). Given the input data $\boldsymbol{x}$, the mean function defines the expectation of the target value $\mathbb{E}(y) = \mu(\boldsymbol{x})$, while the kernel function determines the variance $k(\boldsymbol{x}, \boldsymbol{x})$ of this target value and its covariance $k(\boldsymbol{x}, \boldsymbol{x}')$ with another variable y' at any other input location $\boldsymbol{x'}$.

Therefore, in the perspective of Bayesian inference, the marginal distribution $p(y)$ from the joint distribution 2.47 is seen as the prior probability of y at any input $\boldsymbol{x}$. Supposing a dataset with input values $\mathbf{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^N$ and the corresponding target variables $\boldsymbol{y} = \{y_i\}_{i=1}^N$, the prior probability of $\boldsymbol{y} = f(\mathbf{X})$ is denoted as

$$p(\boldsymbol{y} \mid \mathbf{X}) \sim \mathcal{N}(\mu_0(\mathbf{X}), k_0(\mathbf{X}, \mathbf{X})), \tag{2.49}$$

where

- $\mu_0(\boldsymbol{x})$ is denoted as the prior mean function.
- $k_0(\boldsymbol{x}, \boldsymbol{x}')$ represents the prior covariance (kernel) function.

The posterior probability 2.48 of the predictive target values $\boldsymbol{y}_*$ can be written as

$$p(\boldsymbol{y}_* \mid \mathbf{X}_*, \mathbf{X}, \boldsymbol{y}) \sim \mathcal{N}(\mu_0(\mathbf{X}_*) + \mathbf{k}_*^T \mathbf{K}_N^{-1}(\boldsymbol{y} - \mu_0(\mathbf{X})), k_0(\mathbf{X}_*, \mathbf{X}_*') - \mathbf{k}_*^{\mathbf{T}}(\mathbf{K_N})^{-\mathbf{1}}\mathbf{k}_*'), \tag{2.50}$$

where

- the posterior mean and covariance functions are calculated using conditional properties 2.18 and 2.19.
- $\mathbf{K}_N = k_0(\mathbf{X}, \mathbf{X})$ is the covariance matrix of the N input points calculated by the prior kernel function.
- $[\mathbf{k}_*]_N = k_0(\mathbf{X}, \boldsymbol{x}_*)$ and $[\mathbf{k}_*']_N = k_0(\mathbf{X}, \boldsymbol{x}_*')$ are the prior covariance between the N input training points and the predicting points.

The above inference considers the simple scenario when the provided training data $\boldsymbol{y}$ are measured without noise. When the target variables $\boldsymbol{y}$ include additive noise $\boldsymbol{\eta} \sim \mathcal{N}(0, \sigma_\eta^2\mathbf{I})$, the model must account for this noise in its inference. The relationship becomes $\boldsymbol{y} = f(\mathbf{X}) + \boldsymbol{\eta}$, and the prior covariance of the observed data becomes $\mathbf{K}_N + \sigma_\eta^2\mathbf{I}$. The inference of the predictive inputs $\mathbf{X}_*$ can be estimated as [Rasmussen and C. K. I. Williams 2006]:

$$\mu(\boldsymbol{x}_*) = \mu_0(\boldsymbol{x}_*) + \mathbf{k}_*^T(\mathbf{K}_N + \sigma_\eta^2\mathbf{I})^{-1}(\boldsymbol{y} - \mu_0(\mathbf{X})), \tag{2.51}$$

$$k(\boldsymbol{x}_*, \boldsymbol{x}_*') = k_0(\boldsymbol{x}_*, \boldsymbol{x}_*') - \mathbf{k}_*^T(\mathbf{K}_N + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{k}_*' + \sigma_\eta^2, \tag{2.52}$$

In most practical applications of GP, there is typically no a priori knowledge about the mean function $\mu_0(\boldsymbol{x})$, leading to its common specification as a constant zero. The selection of the kernel is critical to the GP's flexibility, allowing it to adapt to varying data complexity by specifying how points in the input space are related.

Selection of Kernel Functions

The core assumption of the GP model is the basic similarity assumption that points with close inputs x are likely to have similar target values y. Furthermore, a test point near the training data is expected to yield predictions with lower uncertainty due to higher informative neighbours.

The covariance function defines the nearness or similarity between data samples. Therefore, it is the key component of the GP framework, encapsulating the assumptions about the underlying function that the model aims to learn. Common choices for the covariance function include linear, polynomial, or more commonly, Radial Basis Functions (RBF).

The **RBF kernel**, also known as the Squared Exponential kernel, is perhaps the most widely used kernel due to its properties of smoothness and infinite differentiability. It is defined as:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2 \cdot l^2}\right) \tag{2.53}$$

where:

- $\mathbf{x}$ and $\mathbf{x}'$ are vectors in the input space.
- σ^2 is the variance parameter, controlling the overall variance of the kernel. In practical applications, it is typically treated as a hyperparameter and is selected based on empirical evaluation or according to the noise characteristics of the observations.
- l is the length scale parameter, which determines how quickly the correlation between points decays with distance.

The **Automatic Relevance Determination (ARD)** kernel is a sophisticated extension of the standard RBF kernels. It is particularly designed to handle high-dimensional data efficiently by determining the relevance of each input feature. Unlike the simpler RBF kernel, which uses a single length scale l for all input dimensions, the ARD kernel introduces a distinct length scale l_i for each input dimension i. This allows the kernel to adjust its sensitivity to variations in different dimensions of the input space independently. The ARD kernel is defined as:

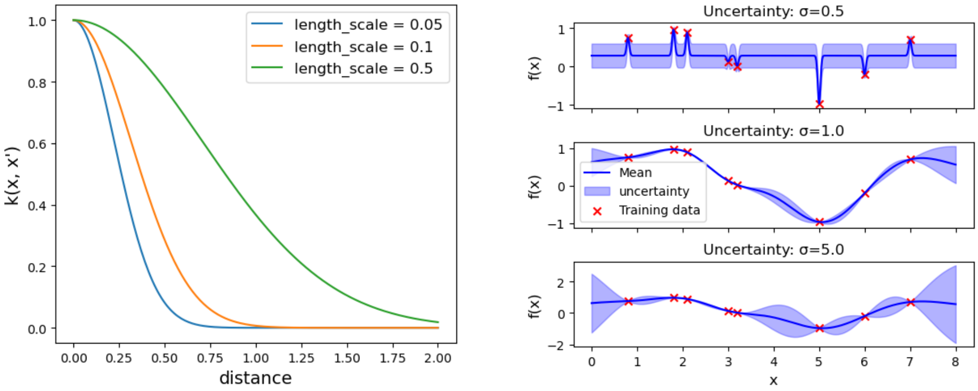
$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2} \sum_{i=1}^{d} \frac{(x_i - x_i')^2}{l_i^2}\right) \tag{2.54}$$

where:

- d is the number of dimensions in the input space.
- x_i and x_i' are the i-th components of $\mathbf{x}$ and $\mathbf{x}'$, respectively.
- l_i are the length scales corresponding to each dimension, allowing the kernel to adaptively measure the relevance of each dimension by scaling the contribution of differences along that dimension.

The ARD kernel is particularly useful in high-dimensional spaces where not all dimensions are equally important. By learning which features are most relevant, the GP model can potentially improve its predictive accuracy and also provide insights into the underlying processes that generate the data.

The **Matérn kernel** is a generalization of the RBF kernel that introduces an additional parameter ν to control the smoothness of the resulting function. The kernel is particularly useful for modelling physical processes with a known degree of smoothness. Its formulation is:



(a) Length scale l impacting distance (b) Prior variance σ impacts

FIGURE 2.6. Impacts of varying hyperparameters: l and σ

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} \|\mathbf{x} - \mathbf{x}'\|}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} \|\mathbf{x} - \mathbf{x}'\|}{l} \right) \tag{2.55}$$

where:

- ν is the smoothness parameter and the smoothness of the function increases with higher ν. Common choices are $\nu = 3/2$ and $\nu = 5/2$, yielding functions that are once and twice differentiable, respectively.
- $\Gamma(\nu)$ is the gamma function, a generalization of the factorial function to complex and real number arguments.
- K_{ν} is a modified Bessel function of the second kind, which handles the behaviour of the kernel function at different scales of ν.

The Matérn kernel's primary advantage is its control over the smoothness of the function. Unlike the RBF kernel, which is infinitely differentiable and thus very smooth, the Matérn kernel can model functions that are less smooth or even non-differentiable when ν is small. This is crucial when dealing with real-world data that might exhibit abrupt changes or is not inherently smooth.

Hyper-parameters Selection

Typically, the covariance function families that we use will have some free parameters whose values also need to be determined. For example, the ARD covariance function is affected by the variance parameter σ and the characteristics length scale $\boldsymbol{l} = \{l_1, l_2, ..., l_i, ...\}$. Also, the additive noise $\boldsymbol{\eta}$ in the noisy observations can be varied and affect the covariance function. In general, these free parameters are known as **Hyper-parameters** in GP inference.

Examples of the effects of varying the hyperparameters on GP inference are shown in Figure 2.6. The length scale l determines how relevant an input is, or how far two inputs become uncorrelated, as shown in Figure 2.6a:

- If the length scale has a very small value, the kernel function decays quickly as two inputs move apart. This means that points close to each other are highly correlated, while points that are even slightly further apart have low correlations. The GP will model the data with a high sensitivity to small variations in the input. This can lead to jaggy inference and overfitting, where the GP model captures noise or very fine-grained details that may not generalize well. Also, the variance of the inference grows rapidly away from the training points.

- In contrast, if the length scale is large, the kernel function decays slowly, meaning that points far apart still have a high correlation. The estimated function will be smoother and less sensitive to local variations. However, it may cause inaccuracy in the small-scale structures.

The variance parameter σ controls the vertical scale or amplitude of the kernel function, also indicating the prior variance if there is no correlated input for the predictive input, as shown in Figure 2.6b.

- If the variance parameter is small, the output of the GP will be heavily restricted around the mean. The GP predictions are unlikely to deviate much from the prior mean function. The model will produce predictions close to the prior mean, even in regions with training points, making the model less flexible. It may also underestimate the uncertainty of the inference, particularly for regions without training data.
- If the variance parameter is large, the GP is more flexible and tolerant to the training data largely deviating from the prior mean. The model will allow larger variations in its predictions, but it can also lead to overfitting if the variance is too high. It could overestimate the uncertainty of the inference, particularly for regions without training data.

Regarding the noise parameter $\boldsymbol{\eta}$ of observations, it affects how confident the GP is in the observations. If set incorrectly, it can lead to overfitting or underfitting and yield inaccurate uncertainty estimates.

There are different methods for optimizing the hyper-parameters from training data. In practice, the hyperparameters are often optimized through MLE or cross-validation to find values that best balance fitting the data and generalization.

Comparison of GP with GMM

Both Gaussian Processes and Gaussian Mixture Models harness the power of Gaussian statistics, albeit in different ways, to provide robust, flexible, and probabilistically sound models that are invaluable across a wide range of applications from robotics to bioinformatics. GMMs adapt to complex data distributions through the mixture components, each with its own mean and covariance, while GPs adapt to data through the choice of the covariance function, which governs the smoothness and other properties of the function being modeled. Both models excel at quantifying uncertainty—GMM through the component variances and GP through its predictive confidence intervals.

2.2.6 Statistical Tests for Assessing out-of-distribution data

Statistical tests designed to determine whether a dataset follows a Gaussian distribution are critical in many areas of research. It is crucial to check if the data follows the estimated distribution or is out of the distribution. A commonly used approach for the multivariate Gaussian distribution is the Chi-squared test.

The Chi-squared test, commonly used for goodness-of-fit, independence tests in categorical data, and homogeneity tests, can also play a role in the context of a multivariate Gaussian (normal) distribution, particularly through its connection to the Mahalanobis distance. If the data indeed follow this distribution, the squared Mahalanobis distance (Δ^2) follows a Chi-squared distribution with degrees of freedom equal to the number of dimensions (variables) in the data:

$$\Delta^2 \sim \mathcal{X}_k^2, \tag{2.56}$$

where:

- $\mathcal{X}_k^2$ is the Chi-squared distribution with k degrees of freedom.

The primary testing strategy is: Given a p-value, if the computed statistics (Δ^2 in the case of MVN) is larger than the corresponding value regarding this p-value and degree of freedom, the testing sample will be seen as out of this distribution.

The testing process can be implemented as follows:

- Calculate the Mahalanobis Distance of the testing sample.
- Compare this distance to the critical value for the p-value of the Chi-squared distribution with k degrees of freedom.
- Hypothesis Testing:

  1) Null Hypothesis (H0): The statistic is smaller than the critical value. The data follow this multivariate normal distribution.

  2) Alternative Hypothesis (HA): The statistic is larger than the critical value. The data do not follow this distribution.

Instead of the binary decision of in or out of distribution, the Mahalanobis distance can also provide an insight into how close the sample is to the mean of the distribution, indicating the uncertainty of the prediction of the sample; a larger Mahalanobis distance implies the sample lies in a low-probability region, i.e., higher uncertainty. Conversely, a smaller distance indicates the sample is close to the mean in a high-confidence region.

2.2.7 Uncertainty Propagation

Uncertainty propagation, also known as error propagation, refers to the process of determining the uncertainty or error in a result that is computed from multiple measurements, each with its own associated uncertainty. This is crucial in scientific and engineering contexts where precise measurements are critical.

We categorize the uncertainties into two types: 1) Random errors, which are unpredictable statistical fluctuations that affect the precision of measurements and can be minimized (though not fully eliminated) by repeated measurements. They are zero-mean and unbiased. 2) Systematic errors, which consistently skew measurements in one direction, arise from biased instrumentation, flawed experimental design, or other sources that cause a repeated, predictable inaccuracy. Correcting systematic errors often requires calibration or refining the experimental setup. Note that although they are similar to the aleatoric uncertainty and epistemic uncertainty in the deep-learning framework, they are not equivalent.

Although aleatoric uncertainty and random error both stem from inherent variability, they differ in how they can be addressed. Random error refers to unpredictable fluctuations in measurements, such as sensor noise, which can be statistically mitigated by aggregating multiple observations. Through averaging, the influence of random errors can be reduced, although not entirely eliminated from individual measurements. In contrast, aleatoric uncertainty reflects fundamental stochasticity in the data-generating process itself—such as the unpredictable motion of dynamic objects or irreducible noise in sensor inputs. This type of uncertainty cannot be reduced even with more data, as it represents intrinsic randomness rather than sampling variation.

The epistemic uncertainty is conceptually similar to systematic errors in that both arise from issues that could, in theory, be resolved with better knowledge or techniques. However, while epistemic uncertainty encompasses broader knowledge limitations, systematic errors are specifically tied to biases in the data collection or experimental setup.

As in our studied problem, no prior information is considered for the systematic error reduction; mainly, the statistical random errors are discussed. The most general way of characterizing statistical uncertainty is by specifying its probability distribution. If the probability distribution of the variable is known or can be assumed, in theory, it is possible to get any of its statistics. Most commonly, the uncertainty of a quantity is quantified in terms of variance or covariance. When combining multiple measurements through various mathematical operations, the uncertainty needs to be propagated accordingly. In the following, we introduce popular methods of uncertainty propagation.

Analytical Methods are employed in cases where the uncertainty propagation calculation can be done through algebraic procedures. Considering scenarios where the target variable is a function of several input variables arranged in a linear system, expressed as $\mathbf{f} = \mathbf{A}\mathbf{x}$, the covariance matrix of the input $\mathbf{x}$ allows for the direct computation of the propagated uncertainty in $\mathbf{\Sigma_f}$, which can be represented as:

$$\mathbf{\Sigma_x} = \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^{\mathbf{T}}] \tag{2.57}$$

$$\begin{aligned}\mathbf{\Sigma_f} &= \mathbb{E}[(\mathbf{Ax} - \mathbb{E}[\mathbf{Ax}])(\mathbf{Ax} - \mathbb{E}[\mathbf{Ax}])^{\mathbf{T}}] && (2.58)\\ &= \mathbb{E}[\mathbf{A}(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^{\mathbf{T}}\mathbf{A}^{\mathbf{T}}] && (2.59)\\ &= \mathbf{A}\mathbf{\Sigma_x}\mathbf{A}^{\mathbf{T}}. && (2.60)\end{aligned}$$

In the case of a non-linear system, the function $\mathbf{f}$ can be linearised by approximation to a first-order Taylor series expansion. The first-order Taylor expansion for a multivariable function of $\mathbf{x}$ in $\mathbb{R}^3$ is :

$$f(\mathbf{x}) \approx f(\mathbf{a}) + \nabla f(\mathbf{a})^T(\mathbf{x} - \mathbf{a}), \tag{2.61}$$

where

- $\nabla f(\mathbf{a})$ denotes the gradient of f at a point $\mathbf{a}$ near $\mathbf{x}$, which is a vector of partial derivatives.

Since we are considering the uncertainty of the expected value $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$, the expansion is evaluated at $\boldsymbol{\mu}$. Given the N dimensional input $\mathbf{x}$ and K dimensional output $\mathbf{f}$, the first-order Taylor expansion of the target $\mathbf{f}$ can be written in matrix notation:

$$\mathbf{f} \approx \mathbf{f}(\boldsymbol{\mu}) + \mathbf{J}(\boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu}) \tag{2.62}$$

where

- $\mathbf{J}$ is the $K \times N$ Jacobian matrix. Each row of this matrix contains the gradient of the component function f_k evaluated at $\boldsymbol{\mu}$.

Therefore, the propagation of error follows the linear case, but replacing the linear coefficients $\mathbf{A}$ with the Jacobian matrix:

$$\mathbf{\Sigma_f} = \mathbf{J}\mathbf{\Sigma_x}\mathbf{J}^{\mathbf{T}}. \tag{2.63}$$

Numerical methods, such as Monte Carlo (MC) simulations, are used for more complex relationships or when analytical solutions are not feasible. MC simulation involves repeatedly sampling the input parameters from their probability distributions, computing the model output for each set of samples, and then analyzing the distribution of these outputs to assess the overall uncertainty and sensitivity. This technique is versatile, accommodating a wide range of applications.

Bayesian Statistical Methods treat model parameters as random variables and use Bayes' Theorem to update the probability distributions of these parameters based on observed data, as introduced in the section 2.2.1. This approach provides a probabilistic description of parameter uncertainty, which can be propagated through the model to predict the uncertainty in the output. It is useful in situations where prior knowledge about parameters is available or where parameters can be updated with new information.

2.3 LiDAR-based Localization

Localization is the process of determining an object's pose within a predefined coordinate system relative to its environment. This encompasses both the position, specified by coordinates x, y and z, and the orientation, typically expressed in terms of pitch, roll, and yaw angles. The primary objective of localization is to accurately establish a moving object's location at any given moment, which is crucial for tasks such as path planning and navigation.

Accurate **Point Cloud Registration** is crucial for the tasks of localization with point clouds. It involves the alignment of two point clouds to find the optimal transformation that makes them coincide. This transformation typically involves translations and rotations that best match the point cloud captured from a sensor to a reference point cloud or a pre-existing model.

2.3.1 Transformation

Rigid Transformations are the most common transformations applied in point cloud registration, preserving the distances between points (i.e., no scaling), including translation and rotation. Note that Affine Transformations are sometimes used, including rotations, translations, scaling, and shearing, offering a more flexible model adjustment compared to rigid transformations.

Rotation is often represented by a rotation matrix R, which is a 3x3 orthogonal matrix with a determinant of +1. Translation is represented by a translation vector $\boldsymbol{t}$, which is a 3x1 vector. The combined rigid transformation can be represented in homogeneous coordinates as

$$T = \begin{bmatrix} R & \boldsymbol{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.64}$$

This 4x4 matrix applies both rotation and translation to a point cloud.

3D Rotation and Quaternions

3D rotations can be typically represented by Euler angles, rotation matrices, rotation vectors, or quaternions.

Euler angles represent rotation by specifying three angles that rotate an object around the axes of a coordinate system. Typically, these angles are about the x-axis, y-axis, and z-axis in a specified order, denoted by the angles ϕ, θ, and ψ, corresponding to roll, pitch, and yaw, respectively. While intuitive, Euler angles are susceptible to gimbal lock—a phenomenon that causes a loss of one degree of rotational freedom when two of the three rotation axes align.

Rotation matrices are 3x3 matrices used to directly rotate vectors in 3D space without changing their magnitude. A rotation matrix is an orthogonal matrix with a determinant of +1. Rotation matrices can be derived from Euler angles by multiplying three basic rotation matrices corresponding to the rotations around the principal axes. In the common ZYX convention, the rotation matrix $\mathbf{R} \in \mathcal{SO}(3)$, the Special Orthogonal group representing 3D rotations, is constructed as:

$$\mathbf{R} = \mathbf{R}_z(\psi) \cdot \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi)$$

where:

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Rotation vectors provide a concise method for encoding a rotation in 3D space using the axis-angle representation. Unlike Euler angle representation, it is not a collection of rotation angles around the x, y, and z axes; rather, it amalgamates the direction of the axis and the magnitude of rotation into a single vector. The direction of the vector specifies the axis of rotation, while its magnitude, measured in radians, quantifies the rotation angle around this axis. This representation is not only compact but also avoids the complexities and limitations associated with Euler angles, such as gimbal locks.

Mathematically, if θ is the angle of rotation and $\mathbf{u} = (u_x, u_y, u_z)$ is a unit vector along the axis of rotation, the rotation vector $\mathbf{r}$ can be expressed as:

$$\mathbf{r} = \theta\mathbf{u} \tag{2.65}$$

Note that the norm $||\mathbf{u}||$ is one.

Quaternions extend the axis-angle representation further as a mathematical tool used for encoding 3D rotations. Unlike traditional complex numbers, quaternions are four-dimensional, comprising one real part and three imaginary parts, and are generally written as

$$q = w + xi + yj + zk \tag{2.66}$$

where

- w, x, y, and z are real numbers.
- i, j, and k are the quaternion units with the properties: $i^2 = j^2 = k^2 = -1$.

Given a rotation vector with the angle θ and the rotation axis vector $\mathbf{u}$, its quaternion is defined as

$$q = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(u_x i + u_y j + u_z k) \tag{2.67}$$

This formulation ensures that the quaternion remains normalized, maintaining its magnitude at 1. This normalization is crucial as it keeps q as a unit quaternion, essential for accurately representing rotations without scaling.

Conversion to Rotation Matrices: Both rotation vectors and quaternions can be converted into rotation matrices, which are more directly applicable for transforming vectors in space. The conversion can be done using Rodrigues' rotation formula:

$$\mathbf{R} = \mathbf{I} + (\sin\theta)\mathbf{K} + (1 - \cos\theta)\mathbf{K}^2 \tag{2.68}$$

where:

- **I** is the identity matrix.
- **K** is the skew-symmetric matrix derived from the unit vector **u** of the rotation axis, given by:

$$\mathbf{K} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

Skew-symmetric matrix and 3D rotation

In linear algebra and vector calculus, a skew-symmetric matrix is a square matrix equal to the negative of its transpose. This property makes skew matrices an interesting subject of study, especially in the context of rotations and angular velocities. A matrix **A** is defined as the skew-symmetric matrix if

$$\mathbf{A}^{\mathbf{T}} = -\mathbf{A}. \tag{2.69}$$

For example, a 3D skew-symmetric matrix can be written as:

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} = \begin{bmatrix} -a_{11} & -a_{12} & -a_{13} \\ -a_{21} & -a_{22} & -a_{23} \\ -a_{31} & -a_{32} & -a_{33} \end{bmatrix} \tag{2.70}$$

The definition yields some important properties:

1. All entries on the diagonal of a skew-symmetric matrix must be zeros, as $a_{ii} = -a_{ii}$.
2. $a_{ij} = -a_{ji}$ means that the matrix is symmetric with respect to the main diagonal but with signs reversed.
3. The eigenvalues of a real skew-symmetric matrix are purely imaginary or zero. This is because the characteristic polynomial of the matrix, which determines the eigenvalues, includes terms that lead to imaginary results due to the antisymmetric properties of the matrix.
4. The determinant and trace of a real skew-symmetric matrix are both zero. The trace is zero because it is the sum of the diagonal elements, which are all zero. The determinant is more nuanced but can be shown through the properties of the eigenvalues and the block structure in larger matrices.
5. For a real skew-symmetric matrix, the matrix exponential e^A (defined by the power series for the exponential function) is an orthogonal matrix.

Thus, the 3D skew-matrix is sufficiently defined by 3 parameters and can be simplified as:

$$\mathbf{A} = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix} \tag{2.71}$$

In the context of 3D rotations, skew-symmetric matrices play a key role in representing angular velocity vectors $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ or rotation vectors $\boldsymbol{\phi} = (\phi_x, \phi_y, \phi_z)$, which can be used to perform cross products via matrix multiplication. For example, the matrix representation of $\boldsymbol{\phi}$ is given by:

$$[\boldsymbol{\phi}]_\times = \begin{bmatrix} 0 & -\phi_z & \phi_y \\ \phi_z & 0 & -\phi_x \\ -\phi_y & \phi_x & 0 \end{bmatrix} \tag{2.72}$$

This matrix is known as the cross-product matrix or the skew-symmetric matrix associated with $\boldsymbol{\phi}$. When this matrix multiplies any vector $\mathbf{v}$, the result is the cross product of $\boldsymbol{\phi}$ and $\mathbf{v}$, i.e., $[\boldsymbol{\phi}]_\times \mathbf{v} = \boldsymbol{\phi} \times \mathbf{v}$.

The skew-symmetric matrix $[\boldsymbol{\phi}]_\times$ is crucial in the dynamics of rotating bodies and robotics, where it is often necessary to compute changes in orientation. For example, in rigid body transformation, the differential of a rotation matrix $\mathbf{R}$ is a skew-symmetric matrix and an infinitesimal rotation matrix can be expressed in the form:

$$\mathbf{R}(d\boldsymbol{\phi}) \approx \mathbf{I} + [d\boldsymbol{\phi}]_\times \tag{2.73}$$

where

- $\mathbf{I}$ is the identity matrix.
- $d\boldsymbol{\phi}$ is a rotation vector with an infinitely small norm, which can also be written in terms of the axis of rotation $\mathbf{u}$ and the magnitude of the rotation angle $d\theta$: $d\boldsymbol{\phi} = \mathbf{u}d\theta$.

2.3.2 Iterative Closest Point

Iterative Closest Point (ICP) is a fundamental technique widely used for aligning or registering two shapes or datasets. It iteratively refines the rotation and translation by minimizing an error metric, typically the sum of squared differences between the coordinates of the matched pairs. This method is widely used due to its simplicity and effectiveness, but requires a good initial alignment to converge to the correct solution. The key steps involved in the ICP algorithm using point clouds include:

- Start with an initial guess of the rotation $\mathbf{R}$ and translation $\mathbf{t}$.
- Select corresponding points in the two point clouds.
- Estimate the transformation to minimize the distance between these points.
- Update the source point cloud by applying the estimated transformation.
- Repeat the process until convergence.

There exist many ICP variants, from which point-to-point and point-to-plane are the most popular. Figure 2.7 illustrates the distances minimized in both point-to-point and point-to-plane ICP variants. The black dots represent the map point cloud, assumed to lie on a plane, while the blue dot corresponds to a point from the source point cloud to be

(a) Point-to-point distance (b) Point-to-plane distance

FIGURE 2.7. Distances to be minimized in both cases

aligned. The red dashed line indicates the distance that should be minimized to estimate the optimal rotation and translation, thereby registering the source point cloud to the map. Point-to-point ICP seeks to minimize the Euclidean distances between corresponding point pairs in the two point clouds, as shown in Figure 2.7a. This is the simplest and most direct form of the ICP algorithm. Given a source point cloud $\mathcal{S}$ and a target point cloud $\mathcal{T}$, where point $\boldsymbol{s}_i \in \mathcal{S}$ corresponds to point $\boldsymbol{t}_i \in \mathcal{T}$, the transformation, $(\mathbf{R}, \mathbf{t})$ that minimizes the sum of squared distances between each pair is computed iteratively as the objective function:

$$\mathbf{R}, \mathbf{t} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{R}\boldsymbol{s}_i + \mathbf{t} - \boldsymbol{t}_i\|^2 \tag{2.74}$$

Point-to-plane ICP minimizes not just the direct distances between points, but the distances from points in the source cloud to the tangent planes at their corresponding points in the target cloud. This is typically more robust and converges faster than point-to-point ICP, especially in cases where the surfaces are approximately planar. The distance minimized is the perpendicular distance from each source point to the plane defined by its corresponding target point and the normal at that point. Thus, the objective function is:

$$\mathbf{R}, \mathbf{t} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|(\mathbf{R}\boldsymbol{s}_i + \mathbf{t} - \boldsymbol{t}_i) \cdot \mathbf{n_i}\|^2 \tag{2.75}$$

where

- $\mathbf{n_i}$ is the normal to the surface at target point t_i

2.3.3 Normal Distribution Transform

NDT is a statistical localization method that was introduced to overcome some of the limitations associated with the ICP algorithm, particularly its dependency on initial alignment and its computational intensity when dealing with large datasets. NDT can also be applied to the process of building a map, but its primary strength lies in localization.

NDT represents the point clouds not as sets of individual points, but as continuous probability density functions. The key difference of NDT is that it optimizes the transformation parameters by maximizing the likelihood of all source points in the reference distributions, rather than minimizing the distances between nearest points.

This process begins with the probabilistic modelling of the reference point cloud. To handle large datasets efficiently, NDT partitions the space into a grid of cells, each modeled by a local Gaussian distribution, as shown in Figure 2.8. Within each cell, the empirical mean $\boldsymbol{\mu}$ and covariance Σ are calculated based on the points that fall into that cell. In Figure 2.8, the shape and orientation of each ellipse represent the covariance structure of

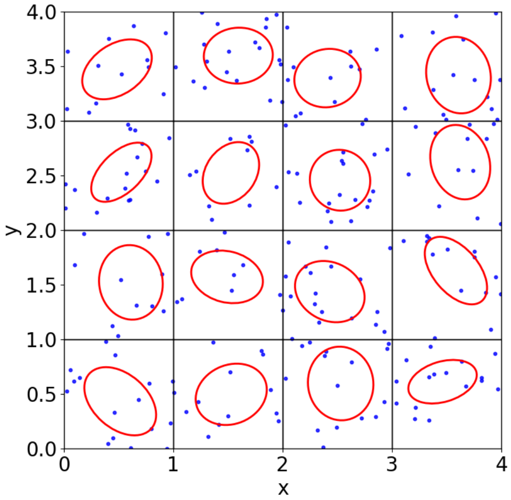


FIGURE 2.8. 2D NDT grid with fitted Gaussians.

the Gaussian. The probability density for a point in each cell is then characterized by the Gaussian distribution defined as:

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.76}$$

where D denotes the dimension of $\boldsymbol{x}$, either 2D or 3D.

With the cell division, points from the source cloud are then assigned to the respective cells where they reside. After applying the transformation, the PDFs of the transformed points are computed based on the Gaussian component of each cell. The negative log-likelihood is then minimized to estimate the transformation parameters. However, outliers in the scan data may significantly impact the negative log-likelihood under a normal distribution. To make the optimization robust to outliers, a mixture of a normal distribution and a uniform distribution is used instead of relying on the original single Gaussian model (Biber and Strasser 2003; Magnusson et al. 2007):

$$p(\boldsymbol{x}) = c_1 \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) + c_2 p_0 \tag{2.77}$$

where p_0 is the expected ratio of outliers. The constants c_1 and c_2 can be determined by requiring that the probability mass of $p(\boldsymbol{x})$ equals one within the space spanned by a cell.

NDT (Biber and Strasser 2003; Magnusson et al. 2007) further approximates the negative log-likelihood of the above mixture distribution by a Gaussian. The approximation retains derivatives that are less complex than those of the log-likelihood of the original mixture model but preserves the same general properties necessary for optimisation. Supposing a source point $\boldsymbol{s}$, the final approximation of the negative log-likelihood of a transformed point ($\boldsymbol{x} = \mathbf{R}\boldsymbol{s} + \mathbf{t}$) reads:

$$\tilde{p}(\boldsymbol{x}) = -d_1 \exp\left(-\frac{d_2}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{2.78}$$

where the constants d_1 and d_2 are specified by:

$$\begin{aligned} d_3 &= -\log(c_2), \\ d_1 &= -\log(c_1 + c_2) - d_3, \\ d_2 &= -2\log\left(\frac{(-\log(c_1 \exp(-\frac{1}{2}) + c_2) - d_3)}{d_1}\right). \end{aligned} \tag{2.79}$$

Note that the d_3 term is omitted in the score function as it is a constant offset and will not change the optimization. The score function is the summation of the negative log-likelihood of all points:

$$L(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N} \tilde{p}(\mathbf{R}\boldsymbol{s}_i + \mathbf{t}) \tag{2.80}$$

The rotation matrix $\mathbf{R}$ is represented by Euler angles in NDT, and thus, the parameters can be encoded using the six-dimensional vector $\boldsymbol{\theta} = \{t_x, t_y, t_z, \phi_x, \phi_y, \phi_z\}$. Newton's algorithm is used to iteratively optimize these parameters in the score function, solving the equation:

$$\mathbf{H}\Delta\boldsymbol{\theta} = -\boldsymbol{g} \tag{2.81}$$

where $\mathbf{H}$ is the Hessian matrix and $\boldsymbol{g}$ is the gradient vector of the score function.

This approach transforms the problem of point cloud registration into the problem of aligning these PDFs, which is typically more robust to variations in point density and noise. In summary, the overall process of NDT is:

- Cell Division: The reference point cloud is divided into a grid of cells. Each cell contains a subset of the point cloud.
- Mapping: Within each cell, the points are assumed to represent samples from a normal distribution. The mean and covariance of the distribution are calculated based on the points within the cell. This essentially models the point cloud as a collection of overlapping Gaussian distributions.
- Transformation Estimation: The transformation needed to align a second point cloud with the reference is estimated by maximizing the likelihood of the points from this cloud under the Gaussian models established from the reference cloud. This process involves iteratively adjusting the transformation parameters (translation and rotation) to improve the fit.
- Optimization: The registration process typically uses gradient descent or other optimization techniques to find the transformation parameters that maximize the likelihood.

NDT offers several advantages that make it well-suited for robust and efficient point cloud registration. One key strength is its robustness to initial misalignment. Unlike ICP, NDT does not require a close initial alignment because it utilizes a probabilistic representation of the point cloud through local Gaussian distributions. This allows it to consider broader spatial information, making it more tolerant of initial pose errors. Additionally, NDT is computationally efficient when dealing with large datasets. By converting the point cloud into a statistical grid-based representation, it avoids costly point-to-point comparisons, enabling faster convergence. Another advantage of NDT is its ability to handle varying point densities naturally. Since each voxel or cell models its own local distribution independently, the algorithm remains effective even in regions with non-uniform sampling, leading to more reliable performance across diverse scanning conditions.

Despite its advantages, the NDT also has several notable limitations. A primary concern is its dependency on cell size, which significantly affects both performance and accuracy. If the cells are too large, important local geometric details may be lost; if too small, the statistical estimates become unreliable, leading to poor registration results. Additionally, while NDT is often more efficient than direct point-to-point methods, it still involves substantial computational overhead. Specifically, calculating Gaussian parameters for each cell and evaluating the likelihood function can become computationally intensive, particularly when applied to large-scale point cloud datasets. Another limitation arises from the underlying assumption that the points within each cell follow a Gaussian distribution. This assumption may not hold for complex or highly structured geometries, making NDT less effective in environments where the data exhibit strong non-Gaussian characteristics.

2.4 Probabilistic Generative Diffusion Models

Diffusion models (Ho et al. 2020; J. Song et al. 2021; A. Q. Nichol and Dhariwal 2021) are a class of generative models in machine learning that have gained prominence for their ability to produce high-quality synthetic data, particularly images. Inspired by non-equilibrium thermodynamics, they work by modeling the process of data generation as a gradual transformation from simple Gaussian noise to complex data structures, effectively "diffusing" and "denoising" data over time. The diffusion model can also be interpreted as one of the Stochastic Differential Equations (SDE) (Y. Song et al. 2020), which builds the connection to the score-based generative methods. In our work introduced in Chapter 7, we developed a diffusion model for 3D Gaussian generation. Here are some theoretical basics about the diffusion model.

2.4.1 Diffuse and Denoise

The model is called diffusion because it gradually diffuses the real data sample with a specific distribution to Gaussian noise. In this diffusing process, noise is incrementally added to these samples over a series of time steps. Given a real data sample x_0 from the complex data distribution $q(x_0)$, the data with added noise at each step is denoted as x_t. At each step, the current, less noisy data will be diffused to a noisier one:

$$q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}x_{t-1}, \sqrt{\beta_t}I), \tag{2.82}$$

where β_t is the noise scheduling variance, and $\alpha_t = 1 - \beta_t$.

With the widely used reparametrization trick, Eq. (2.82) can also be expressed as:

$$x_t = \sqrt{\alpha_t}x_{t-1} + \sqrt{\beta_t}\epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{2.83}$$

where ϵ is the noise sampled from a normal distribution.

The entire forward diffuse process from the clean data to the final Gaussian noise can be represented by a Markov Chain:

$$q(x_{0:T}) = q(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1}), \tag{2.84}$$

Additionally, by integrating out intermediate timesteps, the initial data sample can be transitioned directly to the variable at the $t-$th timestep:

$$q(x_t|x_0) = \sqrt{\prod_{i=1}^{t} \alpha_i} \cdot x_0 + \sqrt{1 - \prod_{i=1}^{t} (\alpha_i)} \cdot \epsilon \tag{2.85}$$

$$= \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon, \tag{2.86}$$

In contrast, the reverse process, often referred to as the denoising process, involves generating new data samples from a state of random noise. This process operates through a series of steps, each aimed at progressively reducing the noise, which can be expressed as:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \tag{2.87}$$

where $p(x_T)$ is the standard Gaussian distribution. Each step $p_\theta(x_{t-1}|x_t)$ in the reverse process can be mathematically represented as follows:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}|\mu_\theta(x_t, t), \sigma_t I) \tag{2.88}$$

$$= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t \epsilon, \tag{2.89}$$

where θ are the parameters of the denoising model that predicts the noise, and ϵ_θ represents the predicted noise.

2.4.2 Optimization Loss

As a probabilistic generative model, diffusion models seek to approximate an unknown "true" data distribution $p(x_0)$ with a model distribution $q(x_0)$. The ultimate goal is that samples drawn from $q(x_0)$ faithfully reproduce the characteristics of the real data (e.g., realistic images of dogs) and thus lie close to $p(x_0)$ in distributional terms.

To achieve this, diffusion models define a forward noising process that gradually corrupts the original sample x_0, and then train a neural network with certain neural parameters θ to predict the exact noise added at each timestep in the diffusing process. Equivalently, this is viewed as learning the score function of the intermediate distributions.

Optimal neural parameters can be obtained by minimizing the cross-entropy between two distributions, which corresponds to maximizing the evidence lower bound. Alternatively, from the perspective of reverse SDE, the estimated score function $\nabla_x \log p(x)$, defined as the derivative of the log-likelihood of data over data samples, should match the score of the true target distribution. This way, the learned reverse process will exactly recover the data distribution.

Both perspectives provide the same loss:

$$\mathcal{L} = \mathbb{E}_{t \sim \mathcal{U}(0,1), x_0 \sim p(x_0), \epsilon \sim \mathcal{N}(0,I)} \left[||\epsilon - \epsilon_\theta(x_t, t)||^2 \right], \tag{2.90}$$

The model is trained to optimize the neural parameters θ for the denoising model. By construction, minimizing the noise-prediction error drives q_θ closer to the true data distribution p. In the reverse denoising process, this predicted noise can be substituted into Equation (2.89) to generate a new data sample from the trained diffusion distribution, starting from the random noise $x_T \sim \mathcal{N}(0, I)$.

Chapter 3

Related Work

3.1 Map representations

The efficacy of environmental representation significantly influences downstream tasks that are sensitive to uncertainty. Traditional occupancy maps (Moravec and Elfes 1985; Elfes 1989; Hornung et al. 2013) represent the space as discrete structured cells, and the occupancy of each cell is modelled independently by ray-casting techniques. These methods have been pivotal in localization and collision avoidance applications. However, the inherent independence of cell modeling can lead to inconsistencies, particularly when dealing with the incomplete, noisy, and sparse data typically obtained from LiDAR systems in urban environments.

The NDT is a widely utilized technique in spatial representation, initially proposed for 2D scan registration (Biber and Strasser 2003). It was subsequently extended to 3D applications by Magnusson et al. 2007. NDT discretizes the space into fixed-size cells, within each of which a Gaussian distribution is fitted to model the spatial data. This approach allows NDT to achieve higher fidelity than traditional occupancy grids at comparably coarser resolutions. Further development led to the creation of the NDT Occupancy Mapping (NDT-OM) framework, as introduced by Saarinen et al. 2013. NDT-OM enhances the NDT framework by incorporating a log-likelihood update mechanism for each cell, facilitating real-time dynamic updates that capture both the normal distribution parameters and probabilistic occupancy within each cell. Despite its advantages, NDT assumes independence between cells, which can introduce high uncertainty at cell boundaries (Srivastava and Michael 2018; Dhawale and Michael 2020).

As an alternative, continuous implicit models (O. Williams and Fitzgibbon 2007; Dragiev et al. 2011; B. Lee et al. 2019; Curless and Levoy 1996; L. Wu, K. M. B. Lee, Gentil, et al. 2023; Gentil et al. 2024; Ivan et al. 2022; Oleynikova, Burri, et al. 2016; Oleynikova, Taylor, et al. 2017; Pan et al. 2022) have demonstrated substantial benefits in addressing these challenges. Signed-Distance Fields (SDF) and Euclidean Distance Fields (EDF), for instance, have been suggested to improve the accuracy and robustness of surface modelling (Curless and Levoy 1996). Many existing systems leverage SDF for scene reconstruction (e.g., Newcombe et al. 2011; Bylow et al. 2013), demonstrating its efficacy in capturing complex geometries. Moreover, by providing dense spatial information on traversable space, SDF is invaluable for collision checks and planning (Ratliff et al. 2009; Oleynikova, Millane, et al. 2016; L. Wu, K. M. B. Lee, Gentil, et al. 2023). Another benefit of SDF lies in its ability to naturally provide direct, informative distance features useful in applications such as ICP localization, where these fields facilitate direct distance computations, eliminating the need for costly nearest-neighbor searches (B. Lee et al. 2019).

In some existing work, distance fields are derived from discrete maps. For example, Oleynikova, Burri, et al. 2016 generate fixed-size EDFs based on discrete OctoMaps (Hornung et al. 2013). FIESTA (Han et al. 2019) obtains EDFs by utilizing the occupancy-based wavefront propagation. Enhancing efficiency, the Voxblox (Oleynikova, Taylor, et al. 2017) and Voxfield (Pan et al. 2022) compute EDFs from Truncated Signed Distance Fields (TSDF) rather than OctoMaps. It involves computing the TSDF first and then propagating the EDF from the TSDF, leveraging a distance propagation technique to improve computation speed. Nonetheless, these methods are limited to estimating discrete distance fields within voxel structures and lack the capability to interpolate distance fields in areas devoid of data, which can be crucial for ensuring continuity in environmental models.

This review has thus highlighted a critical gap: the need for representations that are truly continuous and can manage uncertainty in a proper manner. To address this, we will first explore continuous probabilistic mapping paradigms in the next sections, such as GP and GMM mapping, which perform inference over the entire map space. We will then examine the rise of modern rendering-based representations. Finally, we will discuss how this diverse suite of representations serves as the foundation for advanced 3D generative models. These generative models can be employed as deep priors to reconstruct uncertain or unobserved areas.

3.2 Gaussian Processe-based Mapping

Gaussian Processes offer a continuous, non-parametric, and probabilistic framework ideal for learning and inference within spatial mapping. Leveraging their data-driven nature, GPs adeptly generate accurate and reliable 3D maps from noisy sensor data, effectively encapsulating probabilistic uncertainties. This approach not only enhances the precision of the maps but also quantitatively describes their uncertainty and completeness. This section reviews the advancements in the application of GP-based methods to mapping, highlighting their unique advantages and the challenges they address.

Utilizing GPs in occupancy mapping, GP occupancy map (GPOM) (O'Callaghan et al. 2009; O'Callaghan and F. T. Ramos 2012) has been proposed to introduce the spatial correlation to the neighbouring points to generate continuous maps. This flexible continuous representation facilitates the creation of occupancy maps at arbitrary resolutions, contrasting with and surpassing traditional occupancy grid maps (Moravec and Elfes 1985; Elfes 1989). In GP-mapping, the occupancy states are predicted by GPs and "squashed" into probabilities in the range of [0, 1] by a logistic regression, which is robust with sparse data. Later, more GP-based mapping emerged (Jadidi, Miro, Valencia, et al. 2014; Jadidi, Miro, and Dissanayake 2018). Jadidi, Miro, Valencia, et al. 2014 proposed GP frontier maps to regress a gradient field of occupancy probability distribution as frontier boundaries between known and unknown areas for further exploration. Jadidi, Miro, and Dissanayake 2018 further advance this work by incorporating mutual information-based exploration strategies and developing a more computationally tractable GP-based mapping system. These applications of GPs primarily focused on 2D environments. GP-mapping also paved the way for more complex 3D mapping applications (O. Williams and Fitzgibbon 2007; S. Kim and J. Kim 2014; S. Kim and J. Kim 2015; J. Wang and Englot 2016; B. Lee et al. 2019; L. Wu, K. M. B. Lee, L. Liu, et al. 2021; Ivan et al. 2022; L. Wu, K. M. B. Lee, Gentil, et al. 2023) for both occupancy voxel inference and distance fields inference, where the need for handling larger datasets and more intricate structures became apparent.

The continuous GP inference gives the generalizability to build a map of desired resolution of occupancy cells for diverse applications. However, the drawback of GP lies in the high computational cost due to matrix inversion operations, and this drawback limits the applicability of GPs for large datasets. The scalability of GPs to large datasets, as required in dense 3D mapping, has been a significant area of research. Approximation techniques such as sparse kernels of Gaussian Processes (Melkumyan and F. Ramos 2009), local GP partitioning of the spatial world into subsets using Octree-based data partition (S. Kim and J. Kim 2014; S. Kim and J. Kim 2015; J. Wang and Englot 2016; B. Lee et al. 2019) or OpenVDB(Museth 2013)-based data structure (L. Wu, Le Gentil, et al. 2025), the fusion of local GPs with Bayesian Committee Machines (BCM) (S. Kim and J. Kim 2014; Khan et al. 2020), and low-rank approximation (Smola and Bartlett 2000; Snelson and Ghahramani 2005; Titsias 2009; Hensman et al. 2013; L. Wu, Gentil, et al. 2023) have been developed to manage the computational complexities associated with large-scale 3D environments.

Alternatively, Hilbert Maps (Senanayake and F. Ramos 2017; Guizilini and F. Ramos 2016), as another continuous mapping technique, have been introduced to efficiently map the complex real world by operating on a high-dimensional feature vector. With efficient stochastic gradient descent optimization, it can achieve comparable performance to GP mapping with less time. However, while Hilbert maps can be efficient in making predictions, they typically do not provide as rich information on the uncertainty of those predictions compared to GPs.

Bayesian generalized kernel inference (Vega-Brown et al. 2014; Doherty et al. 2017; Shan et al. 2018; Doherty 2019; Gan et al. 2020) is also proposed to search for a comparable accuracy as GP, but with faster prediction time. For instance, to avoid the unreliable prediction from successive BCM updates, the Bayesian generalized kernel OctoMap (BGKOctoMap) (Doherty 2019) leverages Bayesian kernel inference and sparse kernels to perform a stable inference-based occupancy mapping. Nevertheless, the quality of the approximation is degraded at the price of faster speed.

While GPs offer a continuous regression framework for occupancy mapping, facilitating the generation of occupancy maps at various resolutions, the fundamental nature of occupancy maps remains discrete. Consequently, there has been a shift towards GP implicit surface models that utilize continuous distance field functions to bridge this gap between discrete and continuous spatial representations. Recently, B. Lee et al. 2019 proposed a Gaussian Process Implicit Surface (GPIS) map with derivative observations, where the continuous distance fields are regressed instead of occupancy values. In their work, 2D GP is applied first to obtain the derivatives, and 3D GP with derivative observations is followed to infer the surface distances. GPIS accurately estimates the distance fields that are close to the surface, but it loses accuracy in regions far from the surface. Log-GPIS (L. Wu, K. M. B. Lee, Gentil, et al. 2023) is then proposed to address this issue by applying the logarithmic transformation to a GP formulation, but it faces the trade-offs between accuracy and interpolation abilities. Gentil et al. 2024 solved this problem by leveraging the relation between the kernel and the distance fields. However, the training time of those methods still scales badly with the increasing data size.

There also exist methods that exploit the geometric priors for GPIS inferences. Martens et al. 2017 introduced geometric priors into GPIS to enhance the probabilistic reconstruction of objects. Recently, Ivan et al. 2022 extended the work of GPIS and Log-GPIS with distance field priors extracted from geometry features to reduce the time/memory complexity. This is very close to our work. Instead of a feature extraction, this dissertation employs GMMs to capture the priors, which compactly parameterize point clouds, and the uncertainties are embedded in the covariance matrices of the Gaussian components.

Despite the advantages of the existing GP mapping approaches, GP-based methods for 3D mapping still face challenges, particularly in balancing the accuracy with computational efficiency in large outdoor scenes. Furthermore, the uncertainty inherent in a GP map, which can be exploited to distinguish unknown areas without enough information, has not been sufficiently investigated. This highlights the need to evaluate the reliability of the inferred occupancy probabilities. This issue has been addressed in recent work by Pearson et al. 2022. Additionally, occasional discontinuities in the environment can pose challenges for GP-based mapping, as GPs are often used for continuous targets. This effect is not well-studied in the current literature on GP mapping either. These limitations, therefore, motivate our research into developing scalable, uncertainty-aware, and discontinuity-tolerant GP-based methods for 3D mapping.

3.3 Gaussian Mixture Model-based Mapping

Another group of research is learning a continuous compact map representation with probabilistic parametric or semi-parametric approaches, e.g., GMMs. While the GP inference, with its data-driven nature, captures local surface features, the parametric GMM approach favours the smooth generalization of the global measurement distribution.

In early work, Thrun et al. 2004 fitted a set of rectangular flat surfaces to compose 3D maps, with a group of parameters optimized by EM and the component number estimated by a Bayesian prior. In a similar spirit, but with higher fidelity in approximating diverse arbitrary environments, GMMs have been extensively utilized in 3D mapping. Marriott et al. 2018 estimated Gaussian mixtures instead of 3D planar parameters to extract surfaces. GMMs with large numbers of components have been leveraged to compactly generate a continuous **probabilistic representation of 3D point clouds** (Eckart, K. Kim, Troccoli, et al. 2016; Srivastava and Michael 2016; Goel, Michael, et al. 2023; Goel and Tabib 2023; Gao and W. Dong 2023; Dhawale and Michael 2020), to infer high-fidelity **occupancy maps** of sensor observations (Srivastava and Michael 2018; O'Meadhra et al. 2018; Tabib, Goel, et al. 2019; P. Z. X. Li et al. 2024), perform incremental mapping (Srivastava and Michael 2016; Srivastava and Michael 2018; Dhawale and Michael 2020; Goel and Tabib 2023), and **point cloud registration and localization** (Eckart, K. Kim, and Kautz 2018; H. Dong et al. 2022; Tabib, O'Meadhra, et al. 2018; H. Huang et al. 2020) in the prior work.

Nevertheless, mixture model selection remains a challenge in 3D mapping using GMMs, i.e., it is difficult to determine the proper number of components for various complex scenes. Most conventional methods for determining the number of components are based on the likelihood function and some information criteria (Tao Huang et al. 2017), including Akaike's Information Criterion (AIC) (Akaike 1974), Bayesian Information Criterion (BIC) (Schwarz 1978), Minimum Description Length (MDL) (Hansen and B. Yu 2001), and Kullback–Leibler (KL) divergence between two GMMs (Srivastava and Michael 2016). These criteria help in balancing the model fit against model complexity, preventing overfitting while still capturing significant data patterns. However, the criteria-based method is time-intensive as it has to test various candidate numbers to find the elbow point, which refers to the optimal trade-off point between model complexity and performance.

Bayesian approaches have also been explored as a means to address this issue. One notable example is the Variational Bayesian Gaussian Mixture Model (VB-GMM) (Bishop 2006; Blei and Jordan 2006), which employs a Maximum A Posteriori (MAP) estimator for model selection. In this framework, the number of components can effectively decrease when certain mixing weights become negligible, thereby eliminating unnecessary components. Despite its practical appeal, VB-GMM currently lacks a strong theoretical justification, as its objective function experiences discontinuous changes (Tao Huang et al. 2017). Moreover, variational methods introduce additional parameters compared to EM optimization, leading to higher computational overhead. VB-GMM also requires specifying priors for posterior estimation. Poorly chosen priors can bias the model or slow convergence, limiting the method's reliability and scalability.

Recent research has introduced adaptive, hierarchical structures (H. Dong et al. 2022; Srivastava and Michael 2018; Eckart, K. Kim, Troccoli, et al. 2016; Eckart, K. Kim, and Kautz 2018) and adaptive information-theoretic approaches (Goel, Michael, et al. 2023;

Goel and Tabib 2023) to find a suitable number of GMM components. For instance, Eckart, K. Kim, Troccoli, et al. 2016 employ a curvature threshold for splitting, creating a top-down hierarchical structure in which the point cloud is iteratively partitioned into Gaussian leaf nodes. In contrast, Srivastava and Michael 2018 propose a bottom-up hierarchy that starts with an overestimation of components and then merges them successively based on KL divergence to obtain lower complexity models.

Despite their advantages, these hierarchical methods are sensitive to hyperparameters governing the splitting thresholds in top-down approaches, and they also face trade-offs between accuracy and efficiency in bottom-up schemes (H. Dong et al. 2022). Additionally, adaptive information-theoretic algorithms such as SOGMM (Goel, Michael, et al. 2023) can struggle in complex outdoor scenarios characterized by diverse object scales and shapes, particularly due to potential requirements of high computational resources when tuning the necessary bandwidth parameter.

Additionally, a GMM with high model complexity with large datasets continues to face challenges in terms of scalability. While the above methods utilize GMM as a semi-parametric approach, which would need a large number of components in outdoor scenes, using the traditional EM algorithm to compute GMM parameters becomes extremely time-consuming because it updates Gaussian parameters by traversing all samples in each iteration. The scalability of GMMs to 3D mapping scenarios has seen significant enhancements through the use of hierarchical structures with a top-down strategy (Eckart, K. Kim, Troccoli, et al. 2016; Eckart, K. Kim, and Kautz 2018). However, similar to the issue discussed in model selection, the mapping accuracy of this approach remains sensitive to the hyperparameter of splitting. Dhawale and Michael 2020 proposed a pixel space search-based "region growing" strategy for Gaussian fitting, relaxing the optimization from the expensive EM algorithm, which improves the computational efficiency.

3.4 Neural Implicit Models

Recently, the group of parametric methods based on deep neural networks has provided mapping solutions in the form of neural implicit models; the resulting map can be represented in SDF (Park et al. 2019; Genova et al. 2020; Sitzmann et al. 2020; Gropp et al. 2020; Chibane, Mir, et al. 2020; Ortiz et al. 2022; Wiesmann et al. 2023) and occupancy maps (Mescheder et al. 2019; Jiang et al. 2020). These models are limited by their requirement of access to ground truth 3D geometry for training (Mildenhall et al. 2021), such as the precomputed SDF based on the ground truth mesh.

Relaxing this requirement of ground truth shapes with differentiable rendering techniques, the concept of a Neural Radiance Field (NeRF) (Mildenhall et al. 2021) has been proposed to model the environment with a continuous implicit function of radiance and occupancy, which shows larger potential in mapping, localization, and planning. It is a continuous volumetric function parameterized by a Multilayer Perceptron (MLP). NeRFs are able to produce photorealistic renderings that exhibit detailed geometry and view-dependent effects. Many NeRF-like approaches have been proposed for scene reconstruction and SLAM (Sucar et al. 2021; Pumarola et al. 2021; Barron et al. 2022; Tancik et al. 2022; Zhu et al. 2022; X. Zhong et al. 2023; Yariv et al. 2023; P. Wang et al. 2021; Isaacson et al. 2023). However, as NeRF must learn a large neural network that maps 3D coordinates and viewing directions to radiance and density values, it often requires high computational and memory costs for training and high-quality rendering (Yariv et al. 2023). The inherent neural architecture is also specialized to the particular scene it was trained on and requires retraining or complex modifications to the network structure for a new scene (Goel, Michael, et al. 2023), limiting the model's generalization. Further, NeRF encodes geometry and appearance in a continuous network representation, which is excellent for rendering but less straightforward for tasks like mesh extraction. In uncertainty-aware mapping, the standard NeRF does not provide inherent uncertainty estimates, and thus lacks well-calibrated measures that reflect the reliability of the mapping results.

3.5 Gaussian Splatting

Gaussian splatting (GS) (Kerbl et al. 2023) is a point-based blending approach that uses 3D Gaussians with associated positions and opacities to blend and render the color information. Compared to NeRF, this method offers better efficiency and flexibility in representing complex scenes. GS uses 3D Gaussians to explicitly render images, which focuses more on the occupied space and its neighborhoods rather than full volumetric rendering along the ray compared with NeRF-based approaches. While it efficiently renders the color information, it can not guarantee the precise spatial density and geometries.

Unlike GMM-mapping, GS serves more as point-based blending and is not explicitly calibrated to represent the 3D point cloud distribution. One limitation of 3D GS is the inaccurate estimation of underlying scene geometry and structure, e.g., Gaussians do not precisely match the actual surface (Guédon and Lepetit 2024; B. Huang et al. 2024). The geometry information is modelled by rendering depth images. However, rendering from different viewpoints might lead to inconsistency in depth and thus result in inconsistent geometry (B. Huang et al. 2024; H. Zhou et al. 2024), instead of being directly regressed by Gaussians. Also, 3D GS does not natively model surface normal, which is essential for high-quality surface reconstruction (B. Huang et al. 2024). Unlike methods that produce explicit meshes or consistent SDF representations, Gaussian splatting does not yield a directly manipulable 3D geometry. This complicates downstream tasks like physical simulation, collision checking, or precise shape editing.

Additional enhancements or modifications might be required for good geometry and surface reconstruction. For instance, in the work of C. Zhao et al. 2024, an additional 3D mesh derived from LiDAR is applied to enhance the geometry estimation. To generate the continuous SDF representations, recent research (Guédon and Lepetit 2024; M. Yu et al. 2024; Dai et al. 2024; Z. Yu et al. 2024; Lyu et al. 2024) developed geometry-aware GS variants.

As both Gaussian Splatting and GMM-based mapping employ 3D Gaussians, they share certain conceptual similarities. However, in GMM-based mapping, 3D Gaussians directly model the spatial density of surfaces or point clouds, and the underlying geometry can be inferred through Gaussian mixture regression. In contrast, GS-based methods typically derive geometry via training and rendering of depth images, rather than directly regressing geometry from Gaussians. Moreover, the inherent uncertainty in GS's Gaussian representations remains largely unexplored.

Despite the demonstrated efficacy of GS, its scalability poses a major challenge, particularly in large-scale scene applications where both computational and memory requirements can become prohibitive. Consequently, there is a pressing need to optimize memory usage for model training and storage (G. Chen and W. Wang 2024).

The GMM-based mapping approach proposed in this dissertation leverages 3D Gaussians, sharing similarities with GS, but with substantially reduced memory and computational overhead, alongside a different initialization and splitting strategy. Specifically, while GS typically initializes one Gaussian per point and later splits or prunes Gaussians in an adaptive control step, our method begins with a small set of Gaussians and then splits according to the features of the Gaussian covariance. This significantly lowers the computational burden and eliminates the need for heavy GPU optimization during both training and inference. Moreover, although GS can achieve satisfactory time efficiency on GPU hardware, it can easily involve millions of Gaussian parameters for even modest-sized scenes or objects, leading to stringent memory demands.

3.6 3D Generation and Reconstruction

In this study, we extend the proposed uncertainty-aware mapping representation to applications in 3D generation and completion. The motivation stems from the challenges posed by occlusions and incomplete measurements in outdoor 3D sensing. We hypothesize that deep generative models, trained on large datasets, can provide informative priors to infer and reconstruct missing or occluded regions in such environments. To contextualize our approach, this section reviews recent advancements in 3D generation and reconstruction.

3.6.1 General Generative Models

In this research, we emphasize the importance of probabilistic uncertainty in spatial environment mapping. Thus, many mapping approaches we introduced earlier can be seen as generative models within the domain of traditional machine learning, such as GP inference (B. Lee et al. 2019) and GMM regression (Srivastava and Michael 2018). These techniques are fundamentally oriented towards estimating probabilistic distributions for spatial data. They focus on the inference of mapping expectations of SDF or occupancy, and if possible, quantifying the uncertainty as the distributions of these generative models.

Recently, the term "parametric generative models" has become more commonly associated with deep learning-based mapping frameworks. Unlike traditional models, these modern approaches primarily aim to learn implicit distributions of data through neural networks. Among these, Variational Auto-encoders (VAE) (Kingma and Welling 2014; van den Oord, Vinyals, et al. 2017), Generative Adversarial Networks (GAN) (Goodfellow et al. 2014; Radford, Metz, et al. 2015; Karras et al. 2019; Arjovsky et al. 2017), normalizing flows (Papamakarios et al. 2021), autoregressive models (van den Oord, Kalchbrenner, Vinyals, et al. 2016; van den Oord, Kalchbrenner, and Kavukcuoglu 2016), Transformer-based models (Brown et al. 2020; Ramesh et al. 2021), and Diffusion-based models (Sohl-Dickstein et al. 2015; Ho et al. 2020; Rombach et al. 2022) stand out as prominent methods. The principal objective of these models is not merely to infer statistical parameters but to enable the generation of new data samples that mimic real-world data distributions. Here, we are particularly interested in the generative models in 3D mapping.

3.6.2 3D Geometry Generation

Deep generative models have become pivotal in the domain of 3D geometry, catering to both generation and reconstruction tasks. These models leverage a variety of representational forms, including voxels (Brock et al. 2016; J. Wu et al. 2016), point clouds (C.-L. Li et al. 2018; Gadelha et al. 2018; Achlioptas et al. 2018; Yang et al. 2019; Cai et al. 2020; Luo and Hu 2021; Xie et al. 2021; L. Zhou et al. 2021; Zeng et al. 2022; Z. Ren et al. 2024), and implicit representations (Park et al. 2019; Mescheder et al. 2019; Zhiqin Chen and Zhang 2019; Nam et al. 2022; Shim et al. 2023; Chou et al. 2023; M. Liu, Shi, et al. 2024), with seminal works demonstrating their effectiveness across these modalities. For example, the auto-decoder-based approach - DeepSDF (Park et al. 2019) learns a continuous latent representation for a category of 3D shapes and decodes SDFs from the learned latent codes. During inference, a new latent code is optimized to fit input SDF samples, which are often partial or noisy, to enable shape completion, denoising, and compressive representation. In recent years, the advent of differentiable rendering techniques such as NeRF and Gaussian

Splatting have catalyzed the development of advanced 3D generative models with new 3D representations. These models are specifically designed to generate encoded latent outputs, representing the parameters for NeRF (Jun and A. Nichol 2023; Metzer et al. 2023; Müller et al. 2023) and GS representations (Roessle et al. 2024; Mu et al. 2025; Jiaxiang Tang, Zhaoxi Chen, et al. 2025).

From a methodological perspective, diffusion models have become increasingly prominent in 3D generation. The Denoising Diffusion Probabilistic Models (DDPM) (Ho et al. 2020), originally developed for 2D image generation, has been adapted to 3D by replacing the 2D U-Net architecture with appropriate 3D denoising neural networks, enabling the diffusion process to operate effectively on 3D data. The primary deep learning architectures employed in 3D include MLPs for unstructured point representations (Charles R Qi et al. 2016; Charles R. Qi et al. 2017), 3D Convolutional Neural Networks (CNN) for voxel-based processing (Y. Li et al. 2018; Çiçek et al. 2016), hybrid models combining point and voxel grids (Z. Liu et al. 2019), Graph Neural Networks (GNN) (Y. Wang et al. 2019), and transformer-based models (Vaswani et al. 2017; H. Zhao et al. 2021). These architectures form the backbone of denoising networks in 3D diffusion models. For instance, PVD employs PVCNN as its backbone (L. Zhou et al. 2021), while the emerging Diffusion Transformer (DiT) in 3D adopts transformers to replace traditional U-Net architectures (Chou et al. 2023; Mo et al. 2023). Recent SDFusion (Cheng et al. 2023), L3DG (Roessle et al. 2024) and One-2-3-45++ (M. Liu, Shi, et al. 2024) applied 3D-UNet as backbones. Moreover, some studies underscore the impact of latent diffusion as a highly effective strategy for generative modeling (Rombach et al. 2022; Metzer et al. 2023), where latent space is used instead of directly performing denoising on the features. In those methods, VAE is often used to incorporate the latent space (Zeng et al. 2022; Roessle et al. 2024). Nam et al. 2022 instead apply the auto-decoder to estimate latents.

Additionally, the rise of large vision-language models, such as Contrastive Language-Image Pre-training (CLIP) (Radford, J. W. Kim, et al. 2021), has catalyzed novel approaches to 3D generation, focusing on text-to-3D and image-to-3D transformations based on NeRF or GS representations. A new wave of studies harnesses pre-trained 2D diffusion models like Stable Diffusion (Rombach et al. 2022) and Imagen (Saharia et al. 2022), utilizing fixed models to generate 3D outputs from multi-view 2D images through score distillation techniques (Poole et al. 2022; H. Wang et al. 2023; Junshu Tang et al. 2023; R. Liu et al. 2023; M. Liu, Xu, et al. 2024; Jiaxiang Tang, J. Ren, et al. 2023; Yi et al. 2023; Zilong Chen et al. 2024; Y. Zhong et al. 2024). While these novel methods effectively bridge the gap between 2D and 3D representations (e.g., leveraging 2D generative models to optimize 3D model parameters) and utilize pre-trained large models for various generative tasks, they often struggle to ensure geometric consistency in 3D due to inconsistencies across multiple views. In contrast, other efforts directly generate 3D structures from 3D diffusion models (A. Nichol et al. 2022; Jun and A. Nichol 2023; Mu et al. 2025; Metzer et al. 2023; M. Liu, Shi, et al. 2024), inherently maintaining the robust consistency and geometrical integrity of the learned distributions. These innovations underscore a dynamic and rapidly evolving field, merging cutting-edge generative modeling techniques with advanced rendering technologies to push the boundaries of 3D digital content creation.

3.6.3 3D Completion

The preceding discussion highlighted recent advances in deep generative models for 3D generation, underscoring their applicability not only in direct 3D generation and sampling but also in 3D reconstruction and completion tasks. These models leverage trained generative models as prior knowledge, guided by partial measurements or limited-view images to achieve complete reconstructions.

Earlier approaches such as Point Completion Network (PCN) (Yuan et al. 2018) and IF-Net (Implicit Feature Network) (Chibane, Alldieck, et al. 2020) utilize an encoder-decoder architecture, trained as unified models across various shapes, such as cars, chairs, planes, etc. The encoder-decoder setup enables the models to adapt to diverse shapes, enhancing their versatility in handling different object types. PCN, building on the principles of PointNet (Charles R Qi et al. 2016), completes geometric structures from partial point clouds (Yuan et al. 2018). IF-Net introduces implicit functions within feature spaces to refine details and accommodate a broader modality of 3D inputs (Chibane, Alldieck, et al. 2020). Despite their effectiveness, these methods sometimes fail to reconstruct finer local details.

Decoder-only methods such as DeepSDF (Park et al. 2019) can improve to retain more details, which offers richer latent codes unrestricted by an encoder. It is also claimed to use computational resources more effectively by avoiding the need for training encoders. However, they are inherently more prone to overfitting due to direct latent space optimization. Without an encoder to generalize from broad input patterns, auto-decoders might struggle with generalizing to unseen data. E.g., DeepSDF is often trained on a certain category. Moreover, when compared to recent diffusion-based models, auto-decoders like DeepSDF generally demonstrate a diminished capacity to capture finer details.

More recently, a new wave of studies using score distillation sampling (SDS) loss (Poole et al. 2022) lifts the 2D diffusion priors for 3D completion tasks. Some models use partial measurements for geometrical guidance by applying additional geometrical losses to ensure reconstruction accuracy. For instance, Kasten et al. 2024 utilize 2D priors from pre-trained diffusion models to optimize a complete NeRF representation coupled with an SDF for 3D shapes, facilitating mesh extraction via Marching Cubes (Lorensen and Cline 1998). Similarly, Tianxin Huang et al. 2025 complete the shapes with more efficient GS representations, also derived from 2D priors.

Further innovations have emerged through the use of image-conditioned and view-conditioned diffusion priors for shape completion. Zero-1-to-3 (R. Liu et al. 2023) introduces an image-conditioned and view-conditioned 2D diffusion which learns to control the camera viewpoint of the image generation. This method improves geometrical consistency and enables the reconstruction of neural fields conditioned on the input images without additional loss. Building upon this, M. Liu, Xu, et al. 2024 developed it with a single feed-forward pass and reduced the computational time.

Addressing the challenges posed by inconsistencies in multi-view images generated from 2D priors, some approaches have turned to 3D diffusion models as priors. For example, PVD (L. Zhou et al. 2021) proposed the geometry-conditional 3D diffusion where the partial points are treated as the input condition and remain unchanged in denoising to guide the generation of missing parts. PC2 (Melas-Kyriazi et al. 2023) trains an image projection-conditioned 3D diffusion model, capable of directly reconstructing full 3D point clouds from given images. SDFusion (Cheng et al. 2023) employs a multi-modal 3D diffusion model trained for SDF completion and reconstruction, which can be conditioned on partial shapes, images, and text. One-2-3-45++ (M. Liu, Shi, et al. 2024) enhances the geometrical consistency of One-2-3-45 (M. Liu, Xu, et al. 2024) by incorporating a 3D conditioned diffusion model.

With the guidance of the observation loss based on limited views of images, Liao et al. 2024 apply 3D diffusion models trained for NeRF as the 3D prior to reconstruct shapes while concurrently optimizing the poses. Additionally, adopting a Bayesian perspective, Mu et al. 2025 develop a 3D diffusion model for GS and address the reconstruction of incomplete shapes by treating it as a diffusion posterior sampling problem (Chung et al. 2023). The partial information is integrated as the conditional probability in probabilistic inference with prior knowledge.

These approaches are very meaningful to solve the common challenge of incomplete data acquisition and perspective alteration. Yet, there is still a research gap for uncertainty-aware map generation and reconstruction.

Chapter 4

Datasets and Preprocessing

4.1 Datasets

In this dissertation, large-scale real-world datasets and synthetic datasets are both used to validate and evaluate the proposed methods, including real-world large-scale outdoor point clouds, synthetic point clouds for urban scenes, and synthetic instance-level datasets, as shown in Table 4.1.

TABLE 4.1. List of datasets used in this dissertation.

Dataset	Type	Scenario	Application
LUCOOP: Riegl (Axmann et al. 2023)	Real-world	Urban scene	Mapping
LUCOOP: Velodyne	Real-world	Urban scene	localization
LARD (Senanayake and F. Ramos 2017)	Real-world	Urban scene	Mapping
CARLA simulator (Dosovitskiy et al. 2017)	Synthetic	Urban scene	Mapping
ShapeNet (Chang et al. 2015)	Synthetic	Instance-level	Generation

4.1.1 LUCOOP Dataset

The LUCOOP dataset is a comprehensive, real-world, multi-sensor dataset, collected in an urban area of Hanover, Germany. In this thesis, we mainly use the dense 3D LiDAR point clouds, acquired by Riegl sensors and the sparse 3D point clouds scanned by a single Velodyne scanner for our experiments.

Reigl Data

The dense LiDAR point clouds were acquired by a Riegl VMX-250 mobile mapping system, as illustrated in Figure 4.1. The LiDAR system provides dense point clouds with a measurement precision of $5\,mm$ and an accuracy of $10\,mm$. However, the onboard GPS sensor exhibits much lower positional accuracy compared to the laser scanner. To mitigate this limitation, point cloud alignment techniques are employed during pre-processing to correct the GPS-derived trajectories. The residual alignment error introduces positional noise, which is characterized by a standard deviation of approximately $2\,cm$, as reported in Brenner 2016.

The dataset provides highly dense point clouds, collected from five separate measurement campaigns conducted on different dates over the same urban areas. The aggregated data yield a mean nearest neighbor distance of approximately $5\,mm$. This high density supports the reliable construction of ground-truth maps. Out of the complete dataset, approximately 2.2 million points from one single campaign were used to model the uncertain map in our subsequent experiments, while the remainder (approximately 11.6 million

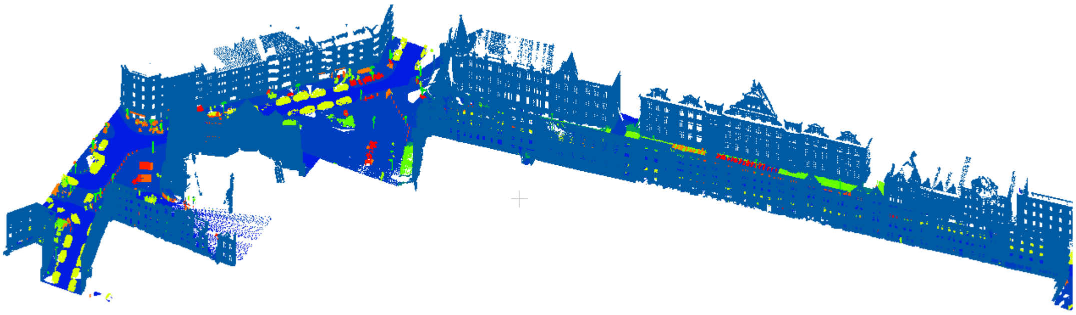
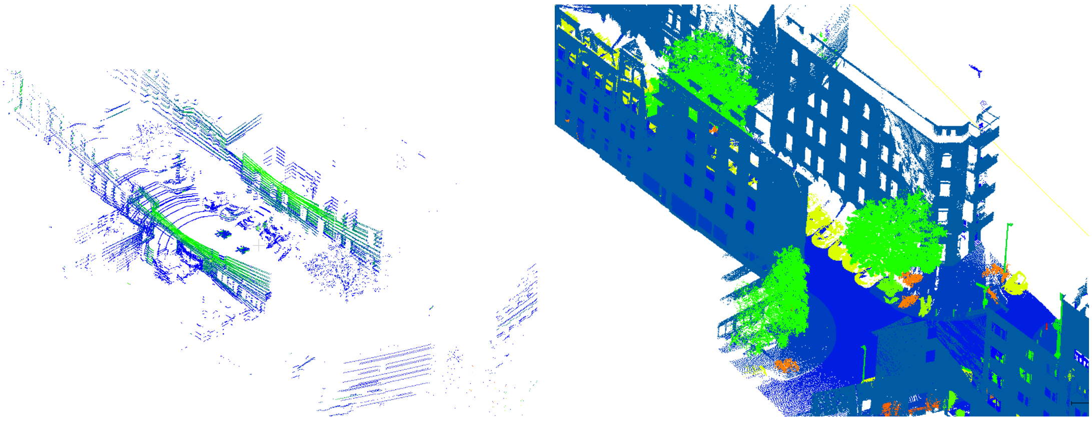


FIGURE 4.1. Real world data



(a) One single Velodyne scan (b) Riegl dense point clouds

FIGURE 4.2. Comparison of the Velodyne scan and the Riegl point clouds

points) served as a ground truth for validation and testing. During a single measurement campaign, the number of points for a single building façade typically exceeded 10,000, with some reaching up to 450,000 in our dataset. The spatial dimensions of the surveyed urban area are approximately $167 \times 380 \times 41\, m^3$.

This dense point cloud is utilized in both Chapter 5 and Chapter 6 to evaluate the proposed uncertainty-aware 3D mapping frameworks. Additionally, it serves as the reference point clouds in Chapter 7 for the localization experiments based on the proposed map representations.

Velodyne Data

The Velodyne dataset provides: (1) LiDAR scans, consisting of sparse point cloud data collected using a Velodyne VLP-16 sensor equipped with 16 laser beams; and (2) post-processed vehicle trajectories, which serve as ground-truth poses corresponding to each LiDAR scan. This dataset is utilized for localization tasks in Chapter 7, where the LiDAR scans are registered to a reference map through point cloud registration to refine pose estimates.

The Velodyne VLP-16 scanner provides a typical range accuracy of approximately $3\, cm$. At a distance of $10\, m$, the vertical spacing between beams is about $35\, cm$, while the horizontal spacing is approximately $3 - 5\, cm$. Figure 4.2a shows an example of a single Velodyne scan, with colors indicating the intensity. For comparison, Figure 4.2b provides the Riegl data in the same area, which is significantly denser than the Velodyne data.

Figure 4.3 displays the vehicle's trajectories and the point clouds used for generating maps within an urban canyon environment, which are evaluated in the experiment. The

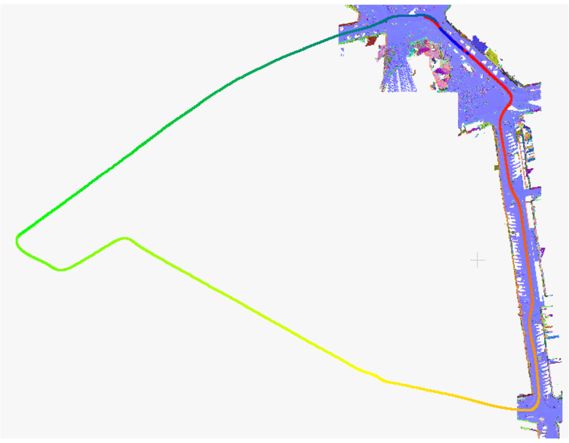


FIGURE 4.3. Trajectories

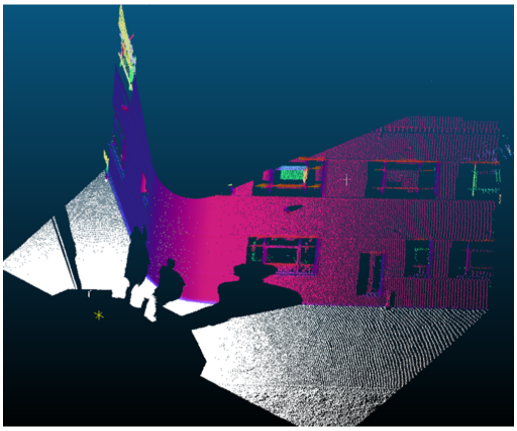
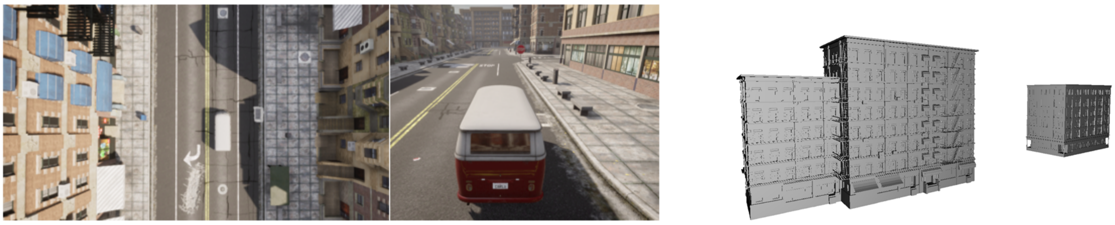


FIGURE 4.4. LARD Hilbert Map Dataset

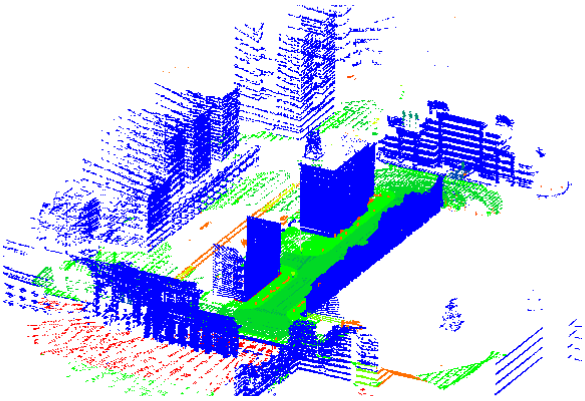
color along the trajectories encodes the temporal progression of the vehicle's motion. The displayed point clouds represent the dense map data from the LUCOOP dataset, with point colors indicating surface normals. This scenario presents a challenge for localization, as most LiDAR measurements are concentrated along the sides of the driving route, forming parallel structures, while relatively few measurements are available in the forward and backward directions. As a result, pose estimation tends to be more accurate in the direction perpendicular to the vehicle's motion, but it becomes more difficult to estimate the pose accurately along the driving direction.

4.1.2 LARD Dataset

The LARD dataset is a real-world dense point cloud used in a benchmark mapping approach known as the local automatic relevance determination (LARD) Hilbert map (Senanayake and F. Ramos 2017). According to the reference, the dataset is composed of about 1.4 million points spread over an area of 400 m^2. The accuracy is about 2 cm. We have also conducted experiments of the uncertainty-aware building models based on this alternative dataset (Figure 4.4). In this dataset, we utilized 20% of the building points for training purposes. The remaining data is used to generate the ground-truth map for evaluation purposes. The dataset is evaluated in Chapter 5 to assess the performance of the proposed uncertain building models.



(a) Simulated scenario (b) Synthetic building mesh



(c) Simulated point clouds

FIGURE 4.5. Simulation in urban scenario

4.1.3 CARLA Dataset

In the experiments, we also evaluate a simulated dataset generated using the Car Learning to Act (CARLA) Simulator—an open-source platform developed for autonomous driving research. Built on Unreal Engine, CARLA provides high-fidelity simulation of urban environments and supports the generation of large-scale, labeled datasets with precise ground truth, including LiDAR point clouds. For this thesis, we utilized predefined scenarios available within CARLA. Specifically, as illustrated in Figure 4.5a, simulations were conducted using a vehicle navigating through the "Synthetic Urban Scenario" to simulate realistic urban driving conditions. During these simulations, LiDAR measurements were collected to capture the 3D structure of the vehicle's surroundings, as shown in Figure 4.5c. Additionally, CARLA provides access to the ground-truth mesh models of the geo-objects in the scene, as shown in Figure 4.5b, enabling accurate evaluation of mapping performance.

Note that the simulated LiDAR measurements include Gaussian noise with a standard deviation of $0.02\,m$. The dataset is used in the experiments presented in Chapter 6 to evaluate the proposed uncertainty-aware 3D implicit mapping framework.

4.1.4 ShapeNet Dataset

In the subsequent 3D generation experiments presented in Chapter 8, we use the widely adopted 3D shape dataset, ShapeNet (Chang et al. 2015), as the training set. ShapeNet provides a large collection of 3D assets represented as continuous, watertight mesh data, as illustrated in Figure 4.6. The dataset includes over 50,000 unique 3D models across 55 object categories, such as chairs, cars, airplanes, and tables. Each model is aligned to a canonical pose, centered at the origin, and normalized to fit within a unit cube (approximately 1 meter in scale). This facilitates training and evaluation in generative modeling, shape completion, classification, and 3D reconstruction tasks.

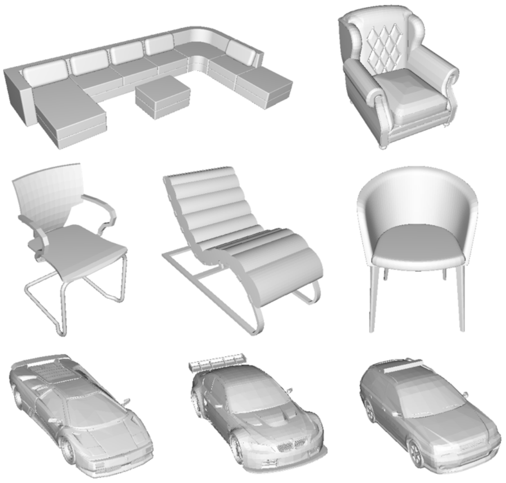


FIGURE 4.6. 3D Dataset: ShapeNet (Chang et al. 2015)

4.2 Data Preparation

4.2.1 Preprocessing of LUCOOP Riegl Data for Mapping

The real-world Riegl point clouds are evaluated in Chapter 5 and 6 for mapping tasks. In Chapter 5, an uncertainty-aware building model is proposed, where only building-related points are utilized. To obtain the "building" points from the full point cloud, the point clouds are semantically segmented by a deep learning-based method (Peters and Brenner 2019). In this approach, labelled scan strips are segmented, and then the 2D prediction is projected to 3D point clouds. Since this leads to various types of label noise due to occlusions, the point cloud must then be corrected, which was done in this case using a supervised neural network.

The point cloud labelled as "building" is segmented into individual building instances. In this step, ground plans from cadastral maps are applied as constraints to separate them and extract the instance-level building-specific data. However, the ground plans cannot accurately separate these points directly, as they are not the true footprints of buildings. For example, the protrusions of different buildings have various distances from the main walls, which are mostly generalized in the cadastral map. In addition, there are other objects in urban scenes, e.g., vegetation and cars, close to buildings. It would be improper to extract all building points by a buffer of ground plan polygons. This is why the learning-based classification is performed first.

The footprints of buildings are represented as polygons, in WGS 1984 UTM coordinates. The point clouds are measured in the same coordinate system. The classified "building" points are assigned to the nearest building instance in the cadastral map, involving a spatial join. This way, the points are segmented into individual building instances, as shown in Figure 4.7. To deal with little remaining label noise in the deep learning classification, points with a distance of more than two meters are ignored.

For the experiments of Chapter 6, it is not necessary to have the above instance-level segmentation, as the approach is proposed to general geometrical structures.

4.2.2 Preprocessing of LUCOOP Trajectory Data

In the LUCOOP dataset, post-processed vehicle trajectories are provided and serve as ground-truth pose information. For the localization experiments, we simulate the pose inaccuracies by artificially introducing perturbations to the data. Specifically, random

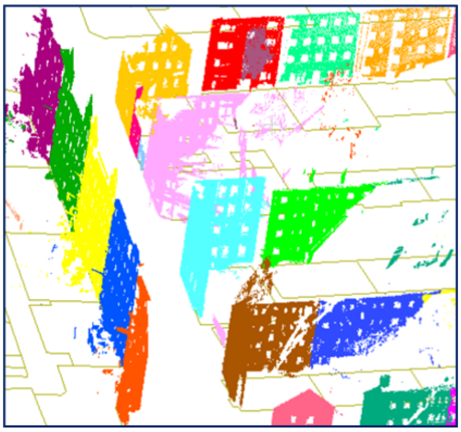


FIGURE 4.7. Building segmentation

biases are added to the ground-truth trajectories as the initial pose errors. The introduced translational bias ranges from 0.3 to 0.8 meters along the x, y, and z axes, and the rotational error is randomly sampled within a range of 10 to 20 degrees. These inputs are then refined through the point cloud registration, yielding the estimated poses used as localization results.

4.2.3 Pre-processing of ShapeNet Data

In the experiments of 3D generation, the goal is to train a model to generate an implicit 3D Gaussian representation for the object surface. Since the synthetic ShapeNet dataset provides watertight mesh models, a preprocessing step is necessary to convert the mesh data into point clouds, which will then be used to construct the proposed 3D/4D GMM representation described in Chapter 6.

The 3D assets in ShapeNet are provided as meshes, which include both externally visible surfaces and non-visible interior structures (e.g., seats inside a car). While it is straightforward to sample points directly from the mesh surface, this approach inevitably includes interior geometry that would not be observed by external sensors in real-world scenarios. Since real LiDAR sensors only capture surfaces visible from the outside, these internal points are undesirable. To generate point clouds that reflect realistic sensor observations, we instead render depth images of each mesh from multiple viewpoints and reconstruct 3D point clouds from the visible surfaces. The approach is illustrated in Figure 4.8.

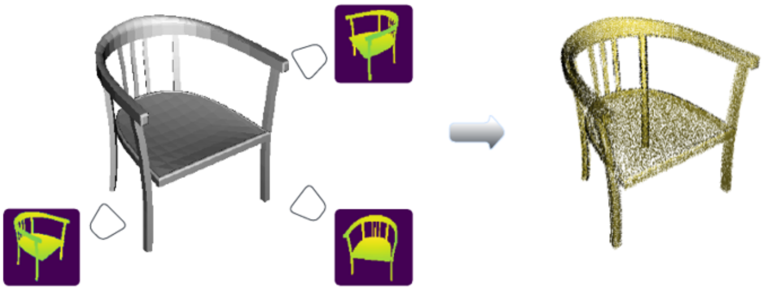


FIGURE 4.8. Reconstruct 3D point clouds from the mesh

4.2.4 Ground Truth Generation

Ground-truth of Occupancy Map

In Chapter 5, the evaluation of the proposed mapping method is conducted using probabilistic occupancy maps. To ensure a fair and valid comparison, the evaluation setup involves a separation between the data used for map generation and the data used for ground-truth construction. Specifically, the evaluated maps are generated using only a subset of the full dataset, and in contrast, the ground-truth probabilistic occupancy maps are constructed using the entire dataset with the probabilistic update introduced in Section 2.1.3. This approach ensures that the ground-truth map captures the true occupancy states with higher confidence and coverage. The same data partitioning and ground-truth generation strategy is applied consistently for both the LUCOOP and LARD datasets.

Ground-truth of SDF

In the experiment of Chapter 6, the result is evaluated with the implicit map representation - SDF. In these experiments, the LUCOOP Riegl point clouds and the synthetic CARLA data are used.

For the real-world LUCOOP dataset, the ground-truth SDF is established by calculating the distances from a query point to its nearest neighbors within the dense point clouds. Given that these reference points constitute discrete samples rather than continuous surfaces, it is crucial to ensure high density to reduce errors in ground truth data. This requirement is satisfied by the LUCOOP dataset, which aggregates point clouds from multiple measurement campaigns. The resulting high-density coverage supports accurate and reliable ground-truth SDF generation..

For the CARLA simulation dataset, the ground-truth SDF is computed from the provided high-fidelity mesh models. These mesh surfaces are directly extracted from the Unreal Engine assets, which serve as the geometric ground truth. The SDF is then calculated based on the shortest distance from each query point in the scene to the nearest point on the mesh surface. To ensure the accuracy of the ground-truth SDF computation, each mesh was subjected to a manual inspection and correction process. This step was crucial as some regions exhibited redundant repetitive meshes of the same surfaces that could introduce errors into the SDF calculations. By rectifying these inaccuracies in the mesh geometry, we enhanced the reliability of the simulated data for subsequent analysis and validation of our model.

Chapter 5

Urban Building Models using GP and GMM in a 2.5D Local Frame

5.1 Overview

Outdoor urban environments are highly unstructured and heterogeneous, comprising a wide variety of objects such as buildings, vehicles, trees, and other infrastructure. Among these, buildings are particularly important due to their relatively simple and stable geometric structures, which make them easier to detect and model. Leveraging building models as a mapping reference is thus highly beneficial for outdoor localization and navigation. However, existing models—such as 3D CityGML—typically do not account for model quality or incorporate uncertainty quantification. In the uncertainty-aware localization applications, often, the uncertainty of building models is not specified at all, or if it is, it is given in terms of a single, global uncertainty measure only (Ehambram et al. 2022). Additionally, the 3D CityGML LoD2 models also only simplify the building façade with one plane and ignore the extrusions and protrusions. This is not sufficient for many tasks requiring higher accuracy.

The objective of this chapter is to develop building models tailored for autonomous robots operating in outdoor settings, with an emphasis on incorporating quantitative uncertainty estimates for each surface point, as illustrated in Figure 5.1. In the figure, the regions with sparse or no measurements are denoted by high uncertainty. Specifically, the mapping task focuses on estimating the façade surface by determining both the relative surface depth and its associated uncertainty at arbitrary query points.

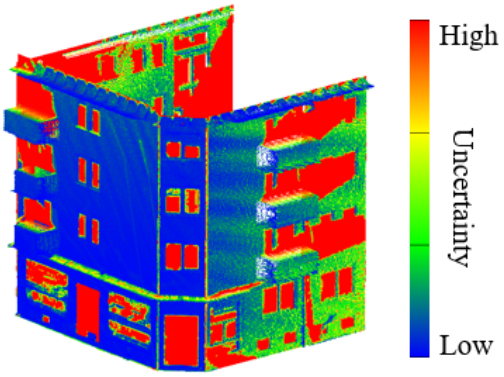


FIGURE 5.1. Depiction of a building model where for each location an uncertainty can be queried (indicated by color).

In this work, we use Gaussian Processes (GPs) combined with 1D Gaussian Mixture Model (GMM) priors to develop an uncertain surface model of structured buildings. Buildings are one of the most important man-made structures for localization in urban

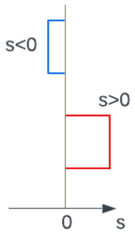


FIGURE 5.2. Illustration of the surface map

exploration, which have relatively simple geometry and are easy to extract. To enhance the scalability of the GP approach for large datasets and model sharp transitions on structured building façades, we segment the entire 3D structure of a building into multiple local planar patches. Each patch is then modelled within a 2.5D framework, allowing for local representation and analysis of each plane individually. Initially, GMMs identify the planar surface layers within each local plane using a few components; subsequently, GPs refine these models by detailing the irregular regions. In the GP inference process, sets of local GPs are assigned to specific planar structures to better capture discontinuities and reduce redundant computations. To further enhance computational efficiency, sparse kernel approximations and data block partitioning—similar to the techniques used in prior work (S. Kim and J. Kim 2015; J. Wang and Englot 2016) —are also employed.. This approach harnesses the flexibility of non-parametric methods for modelling detailed irregular structures and the scalability of parametric methods suitable for large urban environments.

The mathematical representation of the abstract surface map in a local frame can be expressed as:

$$s = f(\boldsymbol{x}) \begin{cases} = 0 & : \text{ on the mean surface,} \\ < 0 & : \text{ on an extrusion,} \\ > 0 & : \text{ on a protrusion.} \end{cases} \tag{5.1}$$

As an example, Figure 5.2 illustrates a simplified surface map. Red rectangle denotes the protrusion (e.g., a balcony), while the blue rectangle indicates the extrusion (e.g., a recessed window).

The surface is modelled as a function with depth values assigned to each point. Given a point $\boldsymbol{x} \in \mathbb{R}^2$ on the local projected surface plane, s denotes the distance to the prior mean surface in this local coordinate. It is important to distinguish this representation from the SDF. While SDF represents the shortest distance between the surface and the query point, s represents the surface location (depth) at the query point relative to a prior surface expectation.

Figure 5.3 illustrates the overall workflow of the mapping process and the evaluation. In the following sections, these processes will be explained in detail. Firstly, we detail the segmentation process used to divide 3D buildings into multiple planes, yielding local 2.5D frames. Secondly, each frame is modelled in two stages: (1) regular planar segments are delineated using a GMM; (2) deviations from these planar segments are captured using GPs, allowing for the modelling of arbitrary details. Finally, we generate an abstract surface map from these models, which is then transformed into a probabilistic occupancy map. This occupancy map is subsequently evaluated against other state-of-the-art methods to assess its accuracy and uncertainty.

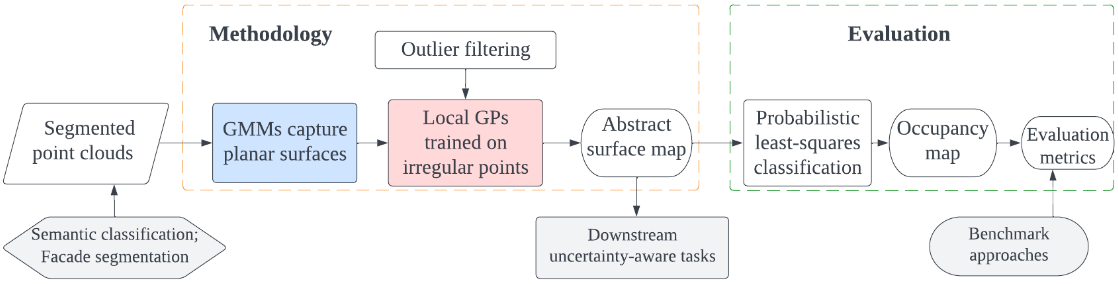

Methodology
Evaluation
Outlier filtering
Segmented point clouds
GMMs capture planar surfaces
Local GPs trained on irregular points
Abstract surface map
Probabilistic least-squares classification
Occupancy map
Evaluation metrics
Semantic classification; Facade segmentation
Downstream uncertainty-aware tasks
Benchmark approaches

FIGURE 5.3. Overall workflow.

5.2 Façade Segmentation

As described, façade segmentation serves as the initial step in dividing a 3D building into local 2.5D façades. In the following, we employ and compare two segmentation techniques: Random Sample Consensus (RANSAC) and GMM segmentation. We provide a detailed introduction to both methods, highlighting their mechanisms and functionalities.

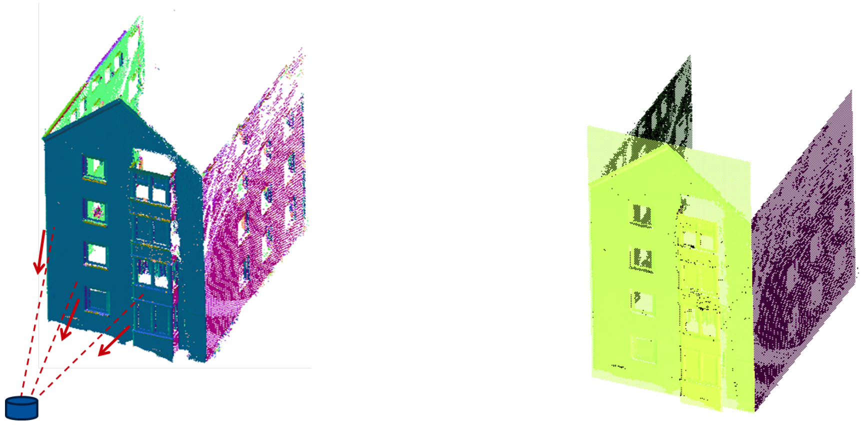
5.2.1 RANSAC Segmentation

Efficient RANSAC for point-cloud shape detection (Schnabel et al. 2007) is employed for façade segmentation, which is an advanced variant of the classic RANSAC algorithm, tailored specifically for efficiently identifying geometric shapes within point clouds. It is particularly effective in environments where the data contains significant noise or outliers. RANSAC fits multiple geometric models to the data—such as planes, spheres, cylinders, and cones—and validates these fits based on the number of inliers. In this work, it is applied to fit planes. The algorithm iterates, adjusting parameters and refining models to improve the fit and accommodate the characteristics of complex, real-world data. This approach implements a hierarchical segmentation strategy that progressively refines the search for shapes to manage the large point cloud datasets.

In façade segmentation, it is assumed that building structures can be decomposed into multiple planar primitives. Each façade plane of an individual building is identified by fitting planes to the data points, adhering to criteria such as a minimum number of support points per primitive and a maximum allowable distance to the fitted primitives. Figure 5.4 demonstrates the RANSAC segmentation of a 3D building, depicting the division into three distinct planes. Figure 5.4a displays the original building points, where colors represent the normal vectors associated with each point. Figure 5.4b shows the resulting segmented façades, clearly illustrating the outcome of the segmentation process.

Note that the planes derived from RANSAC segmentation are characterized by associated normal vectors representing critical orientation information for each plane, which are essential in establishing the local frames. However, these normal vectors do not inherently indicate an outward direction. Therefore, to ensure correct orientation, additional information regarding the sensor's location is utilized to adjust the normal vectors appropriately.

As shown in Figure 5.4a, the blue cylinder represents the LiDAR sensor, and the red dashed lines indicate the emitted rays that strike the objects. The red vectors represent the directions from the measured points back to the sensor. These vectors are averaged to obtain v and then compared with the current normal vector $\boldsymbol{n}$. More specifically, we calculate the dot product of the normal and the averaged vector from the measurement



(a) Original points of a building (b) RANSAC planes

FIGURE 5.4. Example: RANSAC segmentation of a building

points pointing to the sensor: $\boldsymbol{n} \cdot \boldsymbol{v}$. If the dot product is negative, the sign of the normal vector will be flipped: $\boldsymbol{n} = -\boldsymbol{n}$.

5.2.2 GMM Segmentation

3D GMM offers a flexible alternative to RANSAC for segmenting 3D buildings into local planes. This method represents each local plane as an individual Gaussian component. Using the training points, a predetermined number of GMM components is utilized to train a 3D GMM model. Given the 3D point clouds $\mathbf{X} \in \mathbb{R}^3$, a K-component GMM estimates the spatial distribution of the points with the mathematical expression:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k), \tag{5.2}$$

where $(\pi_k, \boldsymbol{\mu}_k, \Sigma_k)$ are the GMM parameters for the k-th planar primitive, π_k is the mixing weight, scaling the component with its importance. $\boldsymbol{\mu}_k$ and Σ_k are the mean and covariance matrix of the Gaussian distribution.

The model with the above parameters will be trained by the EM algorithm as introduced in Section 2.2.3. During segmentation, points are assigned to the component for which they exhibit the highest weighted likelihood $(\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k))$, effectively grouping points into their corresponding planar segments based on this criterion.

Figure 5.5 provides an illustration of a GMM of a building and its resulting segmentation. Figure 5.5a shows the ellipsoids representing the 3D Gaussians in the GMM, with each Gaussian colored differently to denote distinct components; there are three components in this example. Figure 5.5b depicts the segmented façades, where each point is colored according to the Gaussian component to which it has been assigned.

The normal vectors of planar primitives are derived through the eigen decomposition of the covariance matrices associated with each 3D Gaussian component in the model. This process involves calculating the eigenvalues and eigenvectors of the covariance matrices that describe the distribution of points within each Gaussian component. The eigenvector corresponding to the smallest eigenvalue typically represents the normal vector to the plane, as it points in the direction of least variance among the data points, effectively indicating the plane's orientation. The orientation of the normal vector is further adjusted based on the sensor's location, similar to the method introduced in the RANSAC segmentation.

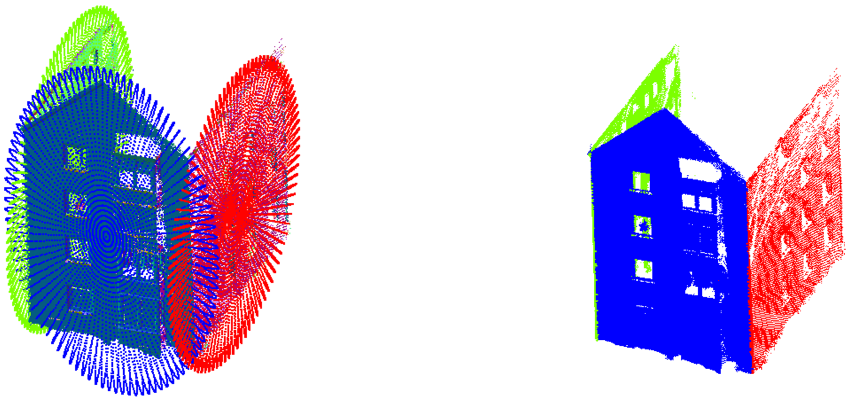

(a) GMM ellipsoids (b) GMM segmented planes

FIGURE 5.5. Example: GMM segmentation of a building

The model selection problem might arise for 3D GMM, i.e., it is often not easy to determine the optimal number of GMM components. However, for the specific application of segmenting large-scale planar primitives of a single building, achieving the exact optimal number of GMM components is less critical. Since the primary goal is to identify major planar structures rather than intricate details, a suboptimal choice in the number of components may still yield satisfactory segmentation results. For example, it can be set to a large fixed value based on empirical experience (e.g., 10 components per $10 \times 10 \times 10m^3$), or determined adaptively using the hierarchical splitting strategy described in Chapter 6.

In practice, the predetermined number of Gaussian components often exceeds the optimal count, leading to oversplit segments. To address this, a merging process is implemented based on specific geometric criteria: if the normals of the Gaussian components are aligned within an angular deviation of fewer than 10 degrees, and the distance between the centers of these Gaussians along the normal direction is less than 0.1 meters, the components are merged into a single Gaussian. This method effectively reduces over-segmentation by consolidating closely aligned and proximate Gaussian components, enhancing the model's accuracy and simplicity.

5.3 GMM for Planar Surfaces

After completing façade segmentation, we proceed to construct our surface model within a local 2.5D frame. As previously discussed, the normal of each segmented plane is established and subsequently designated as the z-axis, representing the axis of surface depth with the positive direction oriented outward. The corresponding x- and y-axes are then defined orthogonally based on the orientation of the z-axis, as shown in Figure 5.6, ensuring a coherent local coordinate system for modelling and analysis.

In this local frame, a 1D GMM is applied to cluster the points with different depth values to depth layers. Each depth layer is described by one Gaussian component in the GMM:

$$p(d) = \sum_{k=1}^{K} \pi_k \mathcal{N}(d|\mu_k, \sigma_k^2), \tag{5.3}$$

where d is the surface depth of a point along the local z-axis, K being the number of Gaussian components; (π_k, μ_k, σ_k) are the GMM parameters for the k-th depth layer, where π_k depends on the number of points belonging to this component. The parameters $\{(\pi_k, \mu_k, \sigma_k)\}_{k=1}^{K}$ are optimized by maximizing the log-likelihood using the EM algorithm.

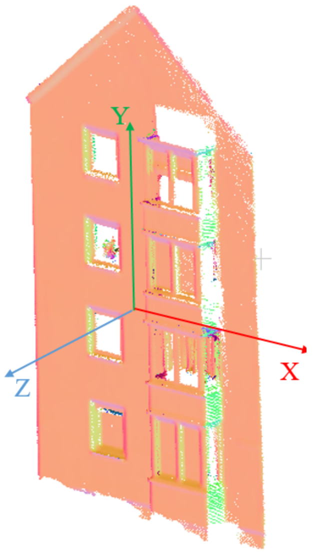


FIGURE 5.6. Local frame

In the simplified 1D GMM, the problem of model selection is reduced, and the number of components, K, is determined through signal analysis. Specifically, K corresponds to the number of peaks identified in the histogram of depth values. A peak in the histogram corresponds to a local maximum, representing a depth interval where a significant number of points are concentrated. These peaks are interpreted as indicating the presence of depth layers. A detailed experimental analysis will be presented later, where the impact of histogram interval selection on GMM results is investigated. This analysis will demonstrate that the GMM's performance is robust, showing minimal sensitivity to variations in histogram interval choices.

In this context, the 1D GMM differs fundamentally from the 3D GMM used in segmentation (Section 5.2.2). Specifically, the 1D GMM is designed to classify varying surface depths, functioning to differentiate between different levels or layers within a structure. Conversely, the 3D GMM is employed to segment distinct façade planes, effectively partitioning the 3D space into local planar sections.

Figure 5.7 shows an example of the GMM results. The left part shows a façade profile, i.e., all scanned points of a façade in a view from the side. Different colors indicate distinct depth layers, e.g., protrusions and extrusions. As observed, points belonging to the same layer are concentrated within a narrow depth range and are well-separated from other layers. The right figure displays the corresponding Gaussian components, with matching colors representing the associated depth layers. The one with the highest peak (green) is the main façade plane.

The limits of GMM are: 1) Non-planar surfaces cannot be modelled. 2) If the number of points on different layers varies greatly, the method may prioritize components associated with more points—those contributing more to the overall GMM distribution—while failing to model the small protrusions and extrusions. These detailed structures only have a few points and can be easily overlooked during the training of GMM using EM. 3) Non-vertical planes might be simply modelled as a vertical plane. The non-parametric approach GP, on the other hand, is capable of modelling all of the above structures in a data-driven fashion. It is, therefore, introduced here to solve these problems.

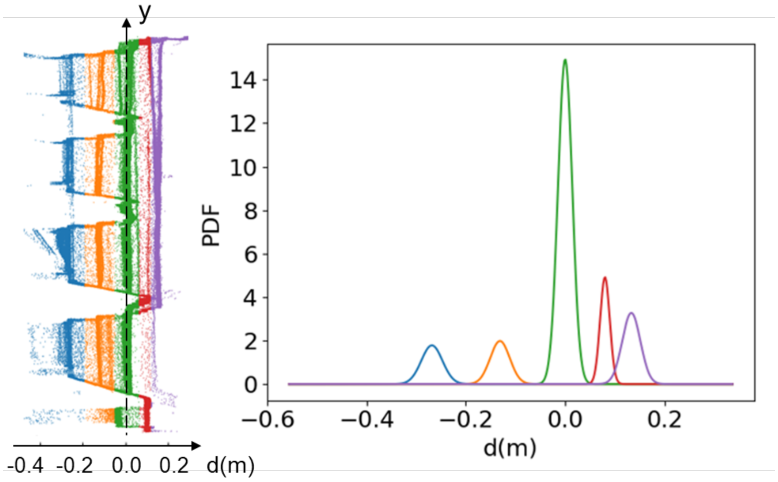


FIGURE 5.7. GMM for plane modelling ($K = 5$)

5.4 GP Using GMM as Prior

5.4.1 Application of GP

A GP $f(\boldsymbol{x}) \sim GP(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ is a popular Bayesian regression method in machine learning. As introduced in Chapter 2, it can be seen as a multivariate normal distribution with infinite variables in a continuous domain. Its posterior mean function $\mu(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ are employed to estimate the posterior surface depth expectation and the uncertainty of this estimation. There are many possible choices for the kernel function and in this application, the ARD kernel is used as the prior GP kernel function. Recalling Equation (2.54), and considering that the input dimension is 2 in this case, the equation can be rewritten as:

$$k_0(\boldsymbol{x}, \boldsymbol{x}') = \sigma_p^2 \exp\left(-\frac{1}{2}\sum_{m=1}^{2}\frac{(x_m - x_m')^2}{l_m^2}\right), \tag{5.4}$$

where $\boldsymbol{\theta} = (\sigma_p, l_1, l_2)$ are the hyper-parameters for the kernel function, as previously introduced in Chapter 2. They are optimized by maximum likelihood, given a training dataset.

In the context of modelling surface depth, provided the projected local points $\boldsymbol{D} = (\mathbf{X}, \mathbf{y})$, where $\mathbf{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ are N input vectors and $\boldsymbol{y} = \{y_i\}_{i=1}^N$ are the corresponding surface depths measured with additive noise $\eta_i \sim \mathcal{N}(0, \sigma_\eta^2)$, we have the following likelihood:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}|\mathbf{K}_N + \sigma_\eta^2\mathbf{I}), \tag{5.5}$$

where $\mathbf{K}_N = k_0(\mathbf{X}, \mathbf{X})$ is the covariance matrix of the input points with $N \times N$ entries calculated by the prior kernel function.

The prediction space $\mathbf{X}_*$ can be inferred conditioned on the training observations $\boldsymbol{D}$. The posterior GP mean and kernel are calculated by (Rasmussen and C. K. I. Williams 2006):

$$\mu(\boldsymbol{x}_*) = \mu_0(\boldsymbol{x}_*) + \mathbf{k}_*^T(\mathbf{K}_N + \sigma_\eta^2\mathbf{I})^{-1}(\boldsymbol{y} - \mu_0(\mathbf{X})), \tag{5.6}$$

$$k(\boldsymbol{x}_*, \boldsymbol{x}_*') = k_0(\boldsymbol{x}_*, \boldsymbol{x}_*') - \mathbf{k}_*^T(\mathbf{K}_N + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{k}_*' + \sigma_\eta^2, \tag{5.7}$$

where the prior mean $\mu_0(x)$ is set to zero, $[\mathbf{k}_*]_N = k_0(\mathbf{X}, x_*)$ and $[\mathbf{k}'_*]_N = k_0(\mathbf{X}, x'_*)$ are the prior covariance between the N input training points and the predicting points.

Figure 5.8 shows an example of the surface depth estimation using the standard GP inference. The right figure presents a 2.5D view of the estimated surface, where color encodes the inferred surface depth. A full GP model is employed, utilizing all available measurements as training points. The region contains several fine-scale surface structures, particularly within the central blue area. As illustrated, the GP approach enables the modelling of even minor, detailed surface variations. In comparison to GMM, the GP method offers greater flexibility, accommodating complex surface geometries with fewer constraints.

The left figure displays a 1D projection for visualization purposes, where x-coordinate remains constant, i.e., $x = 400$. Here, the red dots represent the observations used for GP training. The blue curve represents the GP mean function $f(x)$, which corresponds to the expected surface depth, while the shaded blue region reflects the associated uncertainty (95% confidence interval) derived from the GP posterior variance. The mean function closely follows the training observations and effectively captures local surface variations. However, two noticeable jumps appear around $x = 30$ and $x = 190$, corresponding to abrupt changes in surface geometry. These discontinuities are undesirable in the context of continuous surface modeling. Additionally, the associated uncertainty estimates in these regions are quite low, suggesting a potential underestimation of uncertainty by the GP model in the presence of sharp transitions.

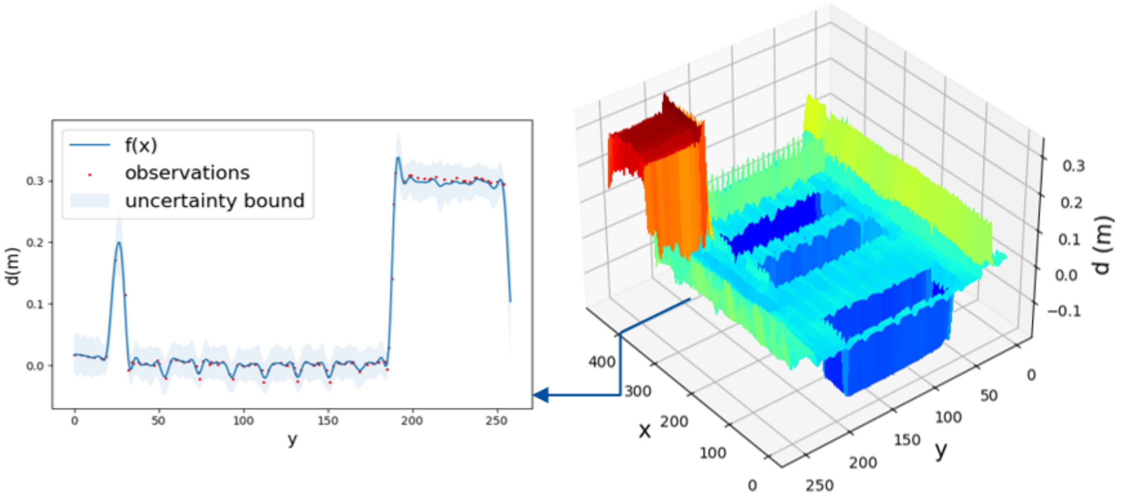


FIGURE 5.8. GP for surface modelling

To utilize a GP for generating an uncertain building model from dense point clouds, there are mainly two problems. First, mapping methods using GPs have to consider the discontinuity of the environment, while the GP often encodes the continuity and smoothness assumptions on targets. It is hard to model functions that exhibit arbitrarily large derivatives with a GP. Unfortunately, large derivatives often appear in environment mapping. This issue is also addressed in the previous work (O'Callaghan and F. T. Ramos 2012; J. Wang and Englot 2016), where they tried special kernels to adapt the discontinuities, e.g. Matérn covariance functions. However, the improvement is limited. Second, with large training datasets, the GP regression faces a computational problem due to the inversion matrix calculation having time complexity $O(N^3)$. To capture the discontinuity and reduce the computational cost of GPs, the two strategies utilized in the following work are: 1) using the GMM results as prior, and 2) partitioning data into blocks. These strategies are explained in detail in the following sections.

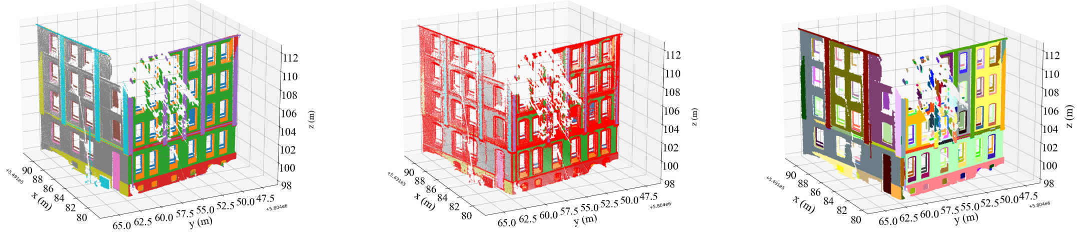
5.4.2 GMM as Prior

Utilizing the GMM as a prior offers several benefits for modelling building surfaces. Primarily, the GMM is employed to delineate the planar regions of buildings, allowing the GP to focus solely on training non-planar and irregular surfaces. These include the points that deviate from the GMM distribution or do not conform to the geometric assumptions underlying segmentation. As a result, the prior information provided by the GMM effectively reduces the number of training points required for GPs, by filtering out regular, planar points and leaving only the irregular points for modelling.

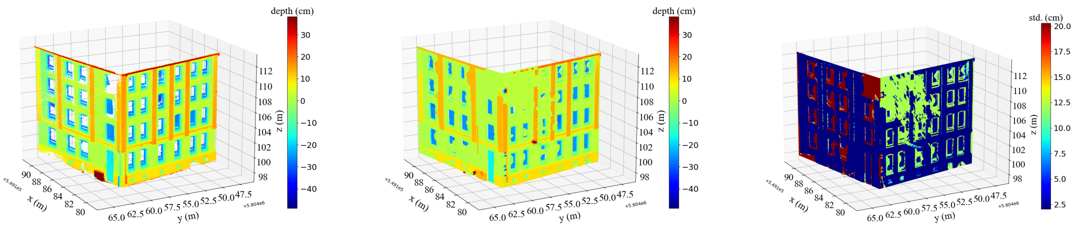
Furthermore, incorporating GMM as a prior enhances GP modelling by providing adaptive parameters for prior distributions. Specifically, the mean values (μ) estimated from the GMM serve as prior means in GP modelling, denoting the prior surface depth expectation of each planar layer. According to GMM clusters, we assign local GPs to different surface regions that belong to a specific GMM layer. Each local GP model has its own hyper-parameters and estimates the surfaces only for the corresponding bounded area. At any location ($\boldsymbol{x}$) on a certain local facet corresponding to the k-th GMM component, the surface depth $d(\boldsymbol{x})$ is modelled by a local GP that incorporates this GMM prior mean. The local GP surface representation is still estimated by the posterior GP inference using Equation 5.6. However, unlike in a standard GP where the prior mean $\mu_0(\boldsymbol{x})$ might typically be zero, here it equals the GMM-derived parameter μ_k, reflecting the expected plane depth from the GMM results. Similarly, the variance of the k-th GMM component parameterizes the prior covariance functions $k_0(\boldsymbol{x}, \boldsymbol{x}')$ in Equation 5.7 and depicts the local data variability.

All local GPs compose a GP list for inference. Since different mean and covariance functions adapt GPs to the local facets, the discontinuities on the façade surface are captured, which is caused by the sharp change between two depth layers.

Figure 5.9 visualizes this overall GP mapping process with GMM as prior. Figure 5.9a demonstrates how surface depth layers are estimated using GMMs. For each layer denoted by distinct colors, points falling outside the GMM distribution are selected as training points for the GP, marked as red in Figure 5.9b.



(a) Surface depth layers estimated by GMMs (b) Training points (red) in GPs (c) Partitioning into local GPs



(d) Ground truth (e) Estimated surface (f) Uncertainty estimation

FIGURE 5.9. Visualization of different stages of the mapping process.

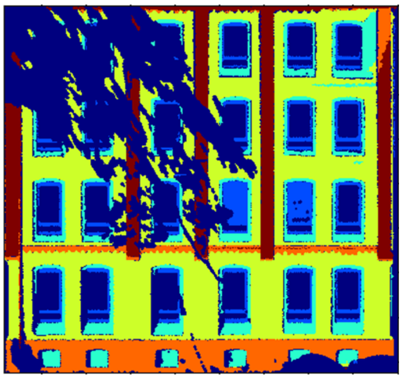


FIGURE 5.10. Surface depth layers

The criteria for selecting training points are introduced as following; we classify the points that cannot be effectively modelled by the GMM into three categories: (1) non-planar points, (2) non-vertical slopes whose normal vectors deviates considerably from the normal of the main plane, and (3) points that are significantly outside their Gaussian distributions, i.e. not within the 95% confidence interval ($1.96\sigma_k$) of a certain component. Figure 5.9b illustrates an example, where the red points are selected as training points for GP training, while the background colors on the building represent distinct depth layers, consistent with Figure 5.9a. These points are selected according to the rules regarding the depth value d and the normal vector $\mathbf{n} = \{n_x, n_y, n_z\}$:

$$|d - \mu_k| > 1.96\sigma_k, \tag{5.8}$$

$$n_z(\boldsymbol{x}) < \cos(\alpha_z), \tag{5.9}$$

where μ_k and σ_k are the parameters of the k-th Gaussian layer, and $n_z(\boldsymbol{x})$ is the component of the normal $\mathbf{n}$, which is perpendicular to the main façade plane. α_z is the angle threshold for the normal $\mathbf{n}$ deviating from the normal of the assigned GMM plane. In this thesis, $\alpha_z = 25°$ is applied.

Figure 5.10 visualizes a façade with segmented surface layers in 2D, where colors indicate different depth layers. Within each depth layer, points belonging to the same local facet are generally interconnected, while different facets are considered discrete from one another. Considering each layer as a cluster, within each depth cluster, we further cluster the points based on their spatial distribution relative to their respective facets. This is achieved by leveraging image processing techniques to project the points onto a 2D plane, enabling efficient and rapid clustering. The training points can be selected before this step and assigned to local facets based on their spatial locations in a 2D plane.

Therefore, each façade plane is further divided into smaller local facets, as depicted in Figure 5.9c, where different colors represent distinct local facets. Adaptive local GPs are employed in corresponding local facets to streamline the complexity of GP training.

Figure 5.9e displays the estimated surface depths and Figure 5.9f denotes the uncertainties. In uncertainty estimation, regions that are sparse or occluded are inferred with relatively high prior uncertainty. This prior uncertainty is derived from the overall surface depth variation of each façade, which explains the differences in color between the two façades. Based on the estimation results, the façade on the left exhibits larger depth variation, leading to higher associated uncertainty. The ground truth of this exemplary building is provided for comparison, as shown in Figure 5.9d.

The estimate for σ_p in the kernel of a local GP is set as the standard deviation of the

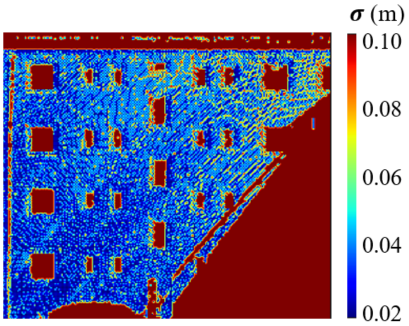


FIGURE 5.11. Example of the prior variance

corresponding planar patch σ_k. Optimization of other hyper-parameters is done on a small subset of the training data, involving a reasonable choice of the initial estimates:

- The initial $\{l_1, l_2\}$ are set to the same value, i.e. isotropic. We set $l_1 = l_2 = 0.05\,m$, observed from the investigation of the impact of the length-scale on kernels and the empirical value from the existing study (O'Callaghan and F. T. Ramos 2012);
- σ_η reflects the precision of the sensors, point cloud alignment noise and random environmental noise. In this case, $\sigma_\eta = 0.02\,m$.

There might be cases where there are regions without any training data and not covered by any local GPs. Then, a global GP prior is included in our GP list. As shown in Figure 5.11, regions without observations are assigned a prior uncertainty (red). The prior variance is set as the variance of the entire façade plane (i.e., compute the variance from all points on this façade), instead of any individual layer, representing the overall variability across all layers of the façade surface. The mean value of this GP is determined by the mean of the Gaussian component with the largest weight in the GMM.

Initially, we considered weighted averaging of the inferred results from adjacent local GP segments in empty areas, using either BCM (H. Liu et al. 2018) or Product of Experts (PoE) (Hinton 2002). However, in our case, the fusion did not significantly improve the results, and the resulting decrease in uncertainty was not desirable. Furthermore, the use of mixtures of GPs would have impacted efficiency. Therefore, in this work, mixtures of local GPs are not utilized.

5.4.3 GP-block Techniques

Although we reduced the training points by the strategy introduced in the previous section, the computational efficiency can be further improved by the data partitioning and GP-block techniques, which are inspired by (S. Kim and J. Kim 2015). In practice, the spatial correlation between two points decreases with their distance. This property is naturally captured by the length-scale hyper-parameter $\boldsymbol{l} = (l_1, l_2)$ in the covariance functions of the GPs. As points far from each other have negligible influence that can be ignored in the covariance matrix, only close points have significant covariance and need to be considered. An idea to exploit this property is to apply a sparse covariance function (Melkumyan and F. Ramos 2009), by setting the covariance to zero when the distance is larger than a threshold. It also indicates the possible data partition of the training data. The entire training dataset can be divided into local blocks within a certain range, as proposed in (S. Kim and J. Kim 2015; J. Wang and Englot 2016). In those methods, the block size is set as the predefined scale value of the sparse kernel. In contrast, our approach determines the block size based on the spatial distance at which the covariance value falls below the threshold c_{min}, as defined by Equation (5.4). This is computed under the condition that $\sigma_p = 1$ and $l_1 = l_2 = l_p$:

$$\exp\left(-\frac{1}{2}\sum_{m=1}^{2}\frac{(x_m - x'_m)^2}{{l_p}^2}\right) \geq c_{min}, \tag{5.10}$$

where $c_{min} = 0.0001$ in this case. The block size is equal to $|\boldsymbol{x} - \boldsymbol{x}'|$ when it makes both sides equal. The extended block of a given block is defined as the central block with its neighboring blocks. In GP regression, the points from the extended block are the training data for the central block being under consideration.

5.4.4 Outlier Filtering

To generate a GP inference robust to outliers, a model is required for outlier processing. There might be outliers in the training samples due to false classifications (e.g., points of road signs in the vicinity of a building). A statistical test on a GP with a Gaussian likelihood is applied to check the probabilistic significance and identify the outliers, e.g., using a Chi-squared test. In this work, the Chi-squared test with one degree of freedom is applied to remove outliers.

The procedure of outlier filtering is as follows.

1. The GP is trained normally with all training data and the mean and variance of each training point are inferred.
2. The Chi-squared test-statistic is calculated using $v_i^2 = (d_i - \mu_i)^2/\sigma_i^2$.
3. The test-statistics are compared to the χ^2 value given a certain α-quantile level and the points with test-statistic values larger than χ^2 value are removed as outliers.
4. The GP is trained again with the filtered training data, and the test points are predicted with mean and variance.

The above process may as well be repeated to iteratively trim the outliers, which, however, will lead to an increased computation time. Therefore, in this case, we only perform the test once in our experiment. The α-percentile level in the test serves as a threshold to filter outliers, indicating the strictness of outlier detection. High values of α will see many points as outliers, while a small α gives more tolerance to the points deviating from the mean and might miss the outliers.

5.5 Experiments

5.5.1 Experimental Setting

The evaluation of the proposed uncertain building model is conducted using real-world LiDAR point clouds from the LUCOOP datasets (Axmann et al. 2023) and the LARD dataset (Senanayake and F. Ramos 2017), as introduced in Chapter 4. For the experiments, only building-related points were utilized. The data preprocessing, data partitioning and the ground truth generation are detailed in Section 4.2.

5.5.2 Ablation Study

In section 5.3, the number of the GMM components K is chosen from the number of peaks in the depth histogram. As described in Section 5.3, these peaks correspond to local maxima, indicating depth intervals with high point density. To assess the sensitivity of the choice of the bin width used when constructing a histogram of depth values, we conduct an ablation study for different bin widths.

In practical experiments, the choice of the histogram bin width is affected by the quality and density of the data, which inherently contain noise that limits the distance resolution at which surface layers can be reliably distinguished. This resolution can be quantified as the minimum distinguishable distance, denoted d_d. For example, if the standard deviation of the measurement noise is $\sigma = 2\text{cm}$, then $d_d = \sigma \times 1.65 \approx 3.3\text{cm}$, where the factor 1.65 corresponds to a 90% confidence level. This implies that, due to sensor uncertainty, the minimum distinguishable distance between two depth layers is approximately 3.3cm. The histogram bin width directly affects the method's ability to resolve distinct surface layers. To preserve the mapping resolution, the bin width should not exceed this minimum distinguishable distance; otherwise, it would artificially merge nearby layers, reducing segmentation accuracy. Therefore, d_d serves as an upper bound for choosing the bin width.

However, choosing a bin width that is too small (e.g., $<$ 1mm) can lead to overly jagged histograms due to insufficient point counts per bin, making peak detection unreliable. To avoid this, we assume a practical lower bound of 1mm for the bin width.

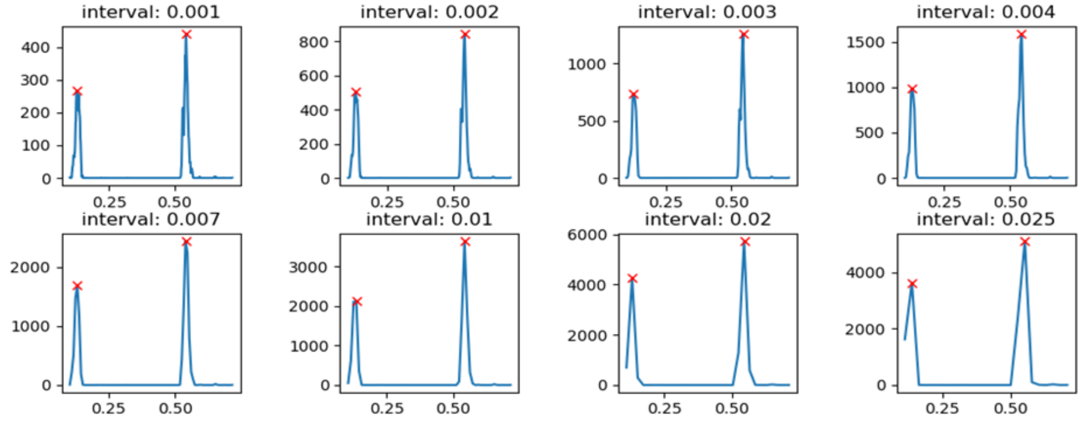
Figure 5.12 shows depth histograms for three façade examples, with depth on the $x-$axis and frequency on the $y-$axis. The depth range (x) varies across examples due to differing surface depth variations. Red crosses mark the detected peaks. For each example, the bin width (interval) is varied from 0.001m to 0.025m.

Experimental results show that the choice of histogram bin width has minimal impact on the outcome. Note that we only consider the number of red crosses, which determines the number of Gaussian components. There are a couple of reasons behind this observation: Firstly, a depth layer typically contains a substantial number of points, making it relatively easy to capture the essential characteristics. Secondly, our primary objective is to identify a reasonable number (K) of peaks for the GMM training. Even if the exact number is not perfectly determined, the subsequent steps involving GP can help correct any inadequately modelled aspects. For instance, consider the limiting case where $K = 1$. Even in this scenario, GP regression remains capable of modeling the underlying function, as demonstrated in Figure 5.8.

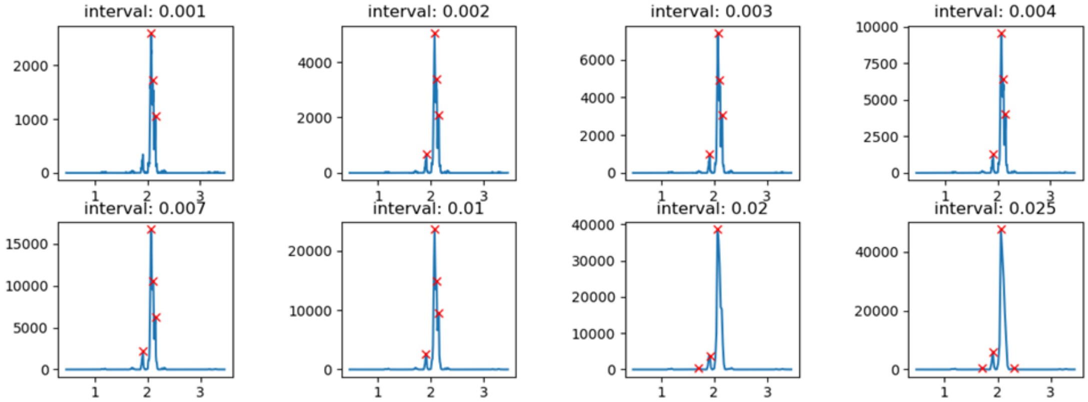
5.5.3 Evaluations

In the local façade frame, the planar points are used as i.i.d. samples to optimize the parameters of the GMM. Training points selected according to Equation (5.8) and (5.9) are used to derive the posterior GP-based models. For different building façades, the number of training points varies. Depending on the building structures, around 20% of the points are selected as training points in the experiment. The initial hyper-parameters are given based on the rules in Section 5.4.2. The initial σ_η is chosen as $2\,cm$ according to the data noise.

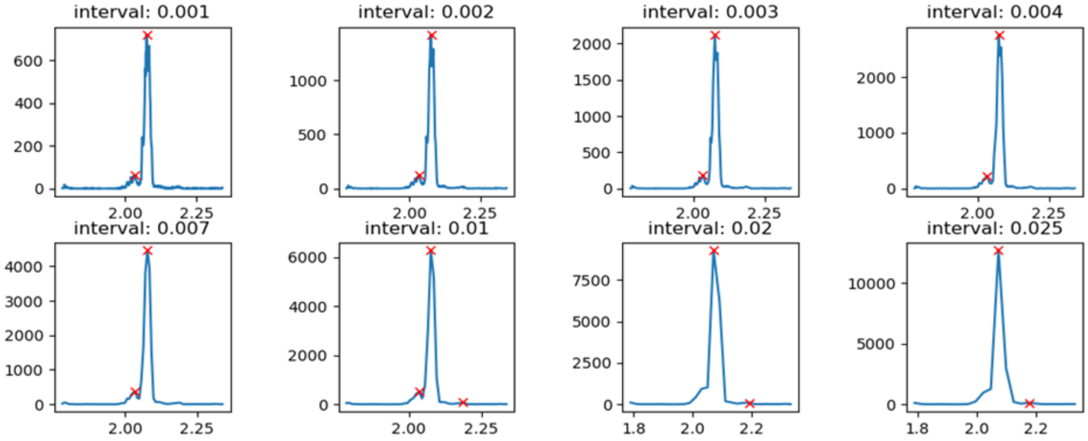
Figure 5.13 shows the qualitative result of modelled buildings with their associated uncertainties. Figure 5.13a displays the overall results, including areas with high uncertainty, while Figure 5.13b illustrates the regions with standard deviation smaller than $0.1\,m$. In the original measurements, certain areas are occluded, where the inferred surfaces exhibit high uncertainties in Figure 5.13a. The uncertainty in occluded regions varies



(a) Example 1



(b) Example 2

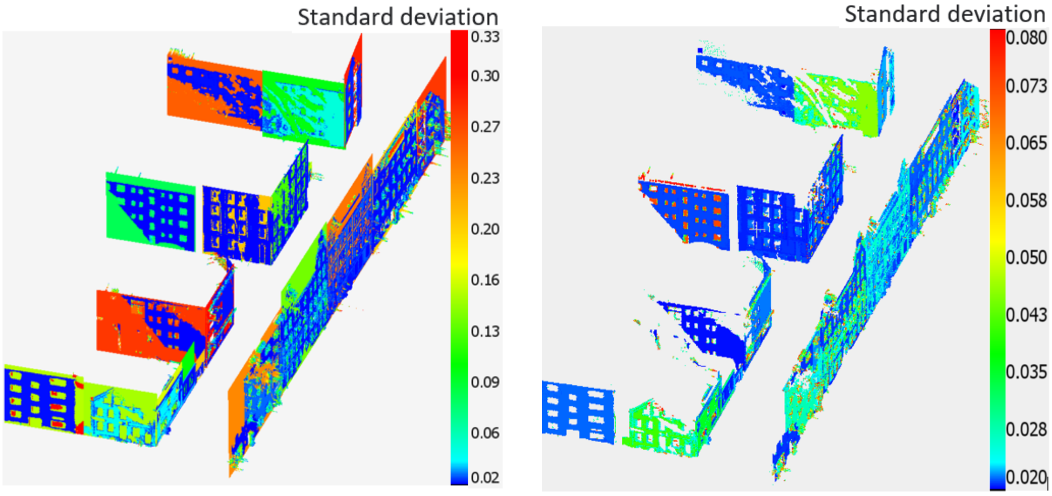


(c) Example 3

FIGURE 5.12. Examples of peak detection for different histogram intervals.

across buildings due to differences in their assigned prior variance, which serves as the uncertainty estimate in areas lacking observations. This prior variance is computed from the overall façade variance, reflecting the expected surface variability and, consequently, the potential reconstruction error. For example, occluded regions on simpler façades with lower depth variation are assigned smaller prior variances. In general, the buildings that have been observed with denser measurements are modelled with lower uncertainties.

In practical localization applications, users are able to select only the accurate part of maps in Figure 5.13b or use the entire map in Figure 5.13a with associated weights indicating the uncertainty. The benefits of only using maps with low uncertainties in occupancy maps are investigated in the following evaluation (see Section 5.5.3).



(a) Surface inference

(b) Accurate inference

FIGURE 5.13. Qualitative results of the building models: uncertainties are denoted by colors. Redder colors indicate higher uncertainties while blue colors denote low uncertainties.

Quantitative Evaluation

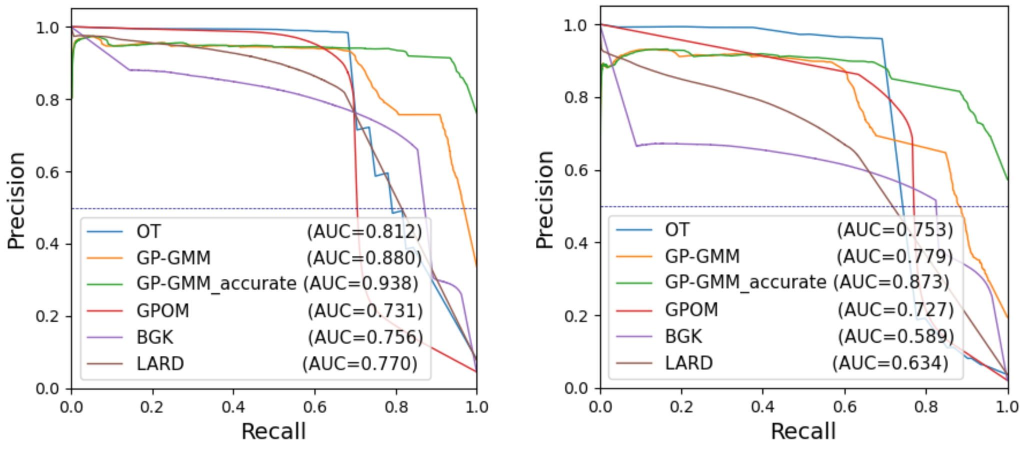
To evaluate the proposed method, several state-of-the-art mapping approaches are adopted as benchmarks, including OctoMap (Hornung et al. 2013), GPOM (J. Wang and Englot 2016), BGKOctoMap (Doherty et al. 2017), LARD Hilbert map (Senanayake and F. Ramos 2017), and the GPIS map (B. Lee et al. 2019). For a fair comparison, the proposed surface map is converted into a 3D occupancy representation. This transformation is achieved using a probabilistic least-squares classification approach, where occupancy probabilities are inferred through a cumulative Gaussian density function, following the methodology presented in prior studies (O'Callaghan and F. T. Ramos 2012; S. Kim and J. Kim 2015). Full implementation details are provided in (S. Kim and J. Kim 2015).

Evaluation metrics include (1) Precision-Recall (PR) curves and (2) Area Under the Curve (AUC). While some earlier works have used Receiver Operating Characteristic (ROC) curves and corresponding AUC values, ROC-based evaluation is known to be suboptimal for highly imbalanced datasets. In particular, when the number of negative (free space) samples vastly exceeds the number of positive (occupied space) samples, changes in false positives have a diminished effect on the false positive rate, which limits the utility of ROC curves (Davis and Goadrich 2006). This issue is especially relevant in outdoor mapping scenarios, where the class imbalance is severe. As noted in previous studies (Miao et al. 2021; Davis and Goadrich 2006), PR curves provide a more informative evaluation in such cases, particularly when the primary concern is the accurate detection of occupied regions. PR curves emphasize the trade-off between precision (false positives vs. true positives) and recall, with the optimal performance corresponding to the upper-right corner of the plot.

Figure 5.14 presents a comparative analysis of AUC scores for all evaluated methods at two spatial resolutions(0.1 m and 0.05 m grid sizes). Overall, GP-based mapping methods demonstrate superior performance, with the proposed method (depicted by green and orange curves) achieving the highest AUC values and exhibiting robustness to finer resolutions. The green curve represents the scenario in which only regions with high-confidence predictions (low uncertainty) are retained, using a variance threshold of 0.01 (Figure 5.13b). Regions with predictive variance exceeding this threshold are treated as unknown. The observed performance improvement in this setting underscores the value of incorporating uncertainty into map representations. In particular, the enhanced performance of high-confidence mapping (green curves) demonstrates the benefit of uncertainty-aware modeling and highlights its potential utility in downstream tasks such as localization, path planning, and navigation under uncertainty.

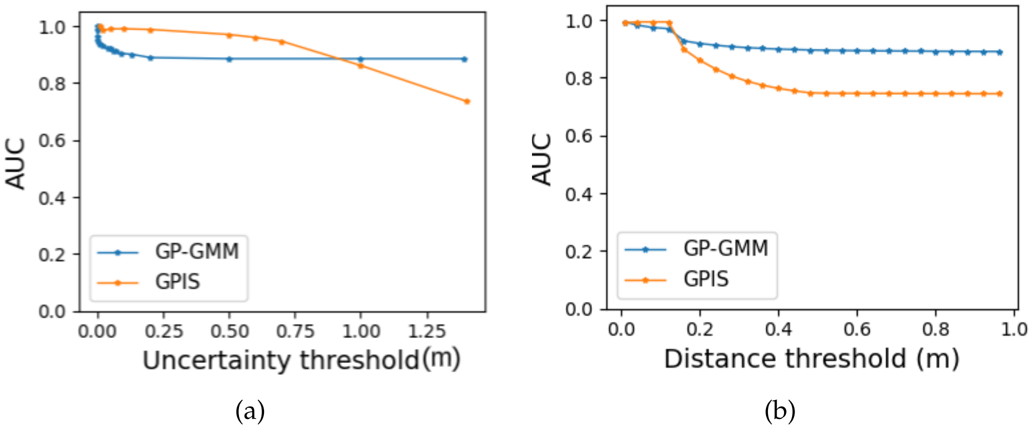
The evaluation of uncertainty is performed by analyzing the variation in AUC values under different uncertainty thresholds and distances between query points and their nearest observations. The proposed method is compared against the GPIS map baseline, with the results illustrated in Figure 5.15. The results reveal a clear monotonic relationship between the AUC and the uncertainty threshold: as the threshold increases, more uncertain regions are included in the evaluation, leading to a gradual decrease in performance. Notably, the proposed method demonstrates larger robustness in regions farther from the observed data, where predictive uncertainty is inherently higher. In these regions, it maintains consistently high AUC values, indicating reliable estimation beyond directly observed areas.

In contrast, the GPIS map achieves higher AUC scores only when predictions are restricted to a small number of query points that lie close to the training data, as shown in Figure 5.15b. As the distance increases, its performance drops significantly. This indicates a reduced ability to provide accurate inference in less observed or unobserved regions. Conversely, the proposed method retains strong performance even when a larger set of predictions is evaluated, highlighting its capability to provide reliable and uncertainty-aware estimations across larger spatial extents.



(a) Resolution: 0.1m (b) Resolution: 0.05m

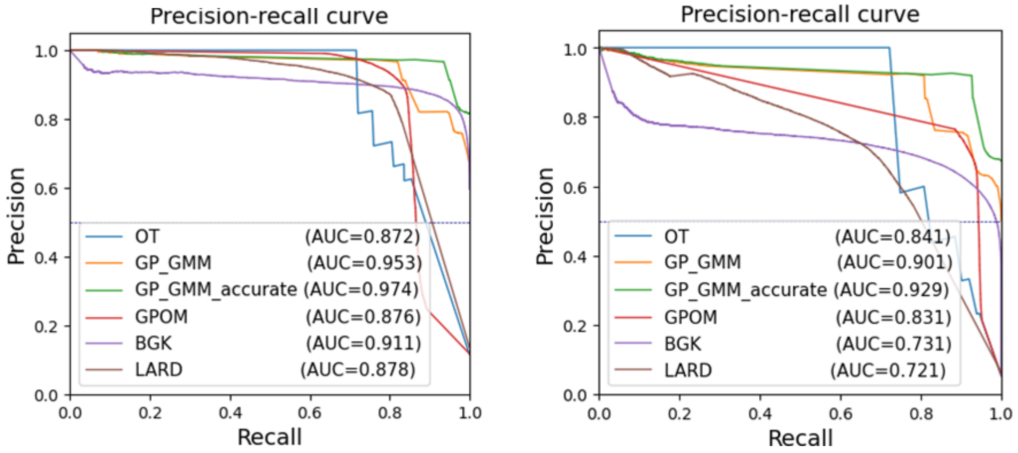
FIGURE 5.14. PR curves and AUC comparison: the AUC value of each curve is given in the legend.



(a) (b)

FIGURE 5.15. Uncertainty evaluation: change of AUC with varying uncertainty thresholds and distances between the predicted points and original training points.

The comparative evaluation of these mapping methods on an additional dataset, originally utilized in the work of the LARD Hilbert map, also confirms our findings in the LUCOOP dataset (Axmann et al. 2023). In both cases, our proposed method consistently outperformed competing approaches. This consistency in performance superiority is visually depicted in Figure 5.16. This auxiliary dataset is characterized by its limited size and relative simplicity. Such attributes typically mitigate modeling challenges, and consequently, contribute to the generally improved results observed for all approaches compared to those obtained with more complex urban datasets.

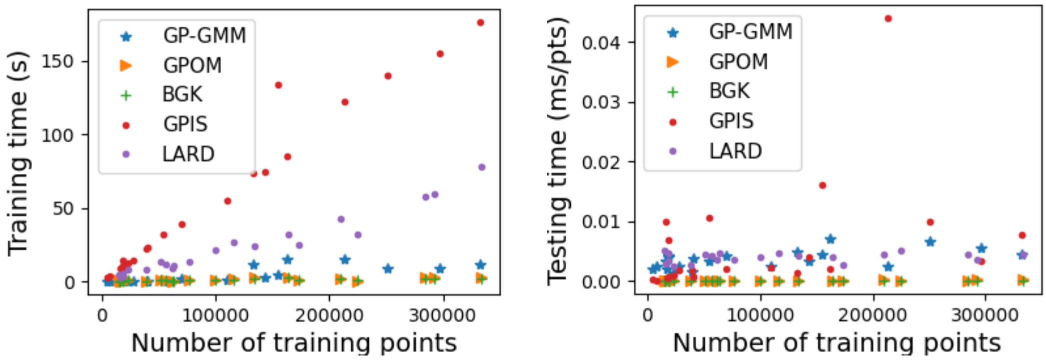


(a) Resolution: 0.1m (b) Resolution: 0.05m

FIGURE 5.16. PR curves and AUC comparison of the LARD Dataset.

5.5.4 Computational Time Comparison

Regarding computational time, BGK and GPOM reach the fastest speed for mapping, as shown in Figure 5.17. Our proposed method demonstrates comparable training times to benchmarks, thanks to the assistance of the GMM prior. During testing, it may require more time than BGK, GPOM, and LARD but remains faster than GPIS. While we acknowledge the importance of efficient online mapping, our goal in this chapter is to deliver a mapping framework that prioritizes accuracy and uncertainty awareness, enhancing map-based localization tasks.



(a) Training time (b) Testing time

FIGURE 5.17. Computational time.

5.5.5 Discussion

The GPOM and BGK models rely on fixed hyperparameters for prior variance, observation noise, and kernel length scale, which may limit their ability to capture spatial correlations and represent local variability in the data. The use of fixed prior variance and kernel parameters makes it challenging for these models to adapt to abrupt changes in voxel occupancy, potentially leading to suboptimal posterior occupancy estimates. Similarly, the GPIS benchmark employs pre-defined values for the prior mean and variance, which can introduce inaccuracies in surface estimation—particularly in regions distant from observations. The fixed variance assumptions can also result in either overestimation or underestimation of predictive uncertainty, making the model's performance highly sensitive to the choice of prior settings.

Our experimental results indicate that LARD exhibits similar performance to the BGK map; both methods demonstrate vulnerability to low resolutions. As a kernel-based approach, the LARD Hilbert map is also strongly influenced by the choice of hyperparameters, such as the kernel length scale, the number of clusters and the number of neighbors. There is limited prior information available to guide the selection process of ideal hyperparameters. Furthermore, the clusters obtained through k-means clustering may not adequately capture the underlying distribution of planar points, further affecting the performance of LARD.

In contrast, the proposed method addresses these limitations by using a GMM to derive data-adaptive priors for both the mean and variance, and by applying localized GP inference to better model spatial correlations and uncertainty. This combination enables the method to more accurately estimate façade surfaces, especially in complex or sparsely observed regions, while providing well-calibrated uncertainty estimates that reflect the confidence of the inference.

5.6 Summary

In summary, this chapter presents a method for generating accurate and uncertainty-aware building surface models from dense LiDAR point clouds. The proposed framework integrates 1D GMMs and local GPs to jointly model both planar and non-planar components of building façades. The GMM captures the dominant planar structures, while GPs are employed to model local surface variations and geometric details. To improve computational efficiency and support surface discontinuities, we introduce local GPs combined with spatial data partitioning strategies.

In the experimental evaluation, the resulting surface models are converted into probabilistic occupancy maps and benchmarked against state-of-the-art methods. The results demonstrate that our approach-using local GPs with GMM-based priors, achieves the highest AUC scores, highlighting its effectiveness in modeling building surface structures and uncertainty estimation.

Although this method is specifically designed for structured, man-made environments, such as building façades, the next chapter extends the framework to more general and complex geometries. This is achieved through higher-dimensional GMM priors and a fully 3D representation using implicit signed distance fields, allowing for a broader range of surface modeling applications.

*The results presented in this chapter are adapted from (Zou and Sester 2021) and (Zou and Sester 2022) published in the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives), and (Zou, Brenner, et al. 2023) published in IEEE Robotics and Automation Letters (RA-L).

Chapter 6

Uncertain Implicit Surface Mapping Using GP and HGMM

6.1 Overview

This chapter focuses on 3D implicit surface modelling with uncertainty quantification, wherein a SDF is inferred as the implicit map representation. A probabilistic mapping approach is proposed, utilizing GP inference with derivative observations, which is effectively integrated with four-dimensional (4D) hierarchical GMR.

The motivation for this work stems from the need for accurate, scalable, and uncertainty-aware 3D mapping in complex urban environments, as well as the limitations of the previous approach that rely on simplified surface assumptions for buildings. While effective in structured urban environments, the previous methods struggle with more complex geometries commonly encountered in general 3D scenes. Modeling rich, continuous surfaces with associated uncertainty in full 3D space is essential for applications such as localization, robotic navigation, and autonomous decision-making in unstructured environments. Therefore, we aim to develop a more general, expressive, and efficient mapping framework that extends beyond the previous planar assumptions.

In Chapter 5, we introduced a GP-based mapping approach tailored to structured buildings, utilizing a 1D GMM as a prior for uncertain surface depth estimation, where the surface was primarily assumed to have large planar geometries. The surfaces were segmented into planes, and each local plane was modelled in a local 2.5D frame rather than a full 3D representation.

In this chapter, we extend the previous methods to accommodate more complex structures by employing 4D GMMs to define the model prior. Note that in this context, "4D" refers to the three spatial dimensions augmented by an additional dimension representing the signed distance field. The use of higher-dimensional GMMs, along with direct regression of the SDF, allows for the transition from local 2.5D surface representations to comprehensive 3D models. This approach reduces geometric constraints, enabling its application to a wider range of surfaces and scenarios, including cases where local 2.5D representation is insufficient. Furthermore, the mapping process is streamlined by eliminating the need for local surface segmentation, thereby reducing training time and minimizing potential segmentation errors. Thus, the seamless connection between 4D GMMs and GP framework contributes to enhanced accuracy, efficiency and robustness compared with the mapping using GP and GMM in a 2.5D local frame.

When GMMs are extended to higher dimensions, the issue of model selection (choice of the component number) arises. Therefore, a hierarchical GMM (Hierarchical Gaussian Mixture Model (HGMM)) technique is employed to determine the number of components and speed up GMM training. GP inference is followed to refine the HGMM priors where they do not comply with the original training points. Incorporating the observations of derivatives, it also provides the capability to estimate the normal of distance fields.

Figure 6.1 depicts the overall mapping workflow, illustrating the integration of the HGMM within the GP framework. Since the GMR technique is used to directly regress SDF from 3D spatial coordinates, a 4D GMM—comprising three spatial dimensions and one additional dimension for the distance field—is required. This necessitates data augmentation. After augmentation, the GMM is adaptively split and trained hierarchically. The resulting HGMM parameters define the prior mean and kernel functions for GP training. Training points for GPs are selected from the original input point cloud data. Specifically, SDFs are regressed for all surface measurements, and points exhibiting significant deviations from the HGMM predictions are chosen as training points. The GP inference in this chapter differs from that in Chapter 5 in two key aspects: it utilizes non-stationary prior functions and incorporates derivative observations into the inference process. Further details on each component of the workflow are provided in the subsequent sections.

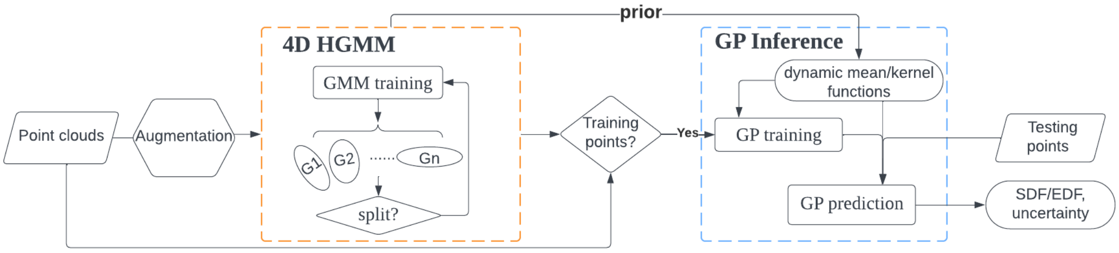


FIGURE 6.1. Overall workflow.

6.2 4D Hierarchical Gaussian Mixture Regression

6.2.1 Hierarchical Structure

First, we revisit a standard 3D GMM composed of K Gaussian components, described by a set of parameters $\Theta = \{(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^{K}$, as given in:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{6.1}$$

where π_k is the prior weight, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are the Gaussian parameters.

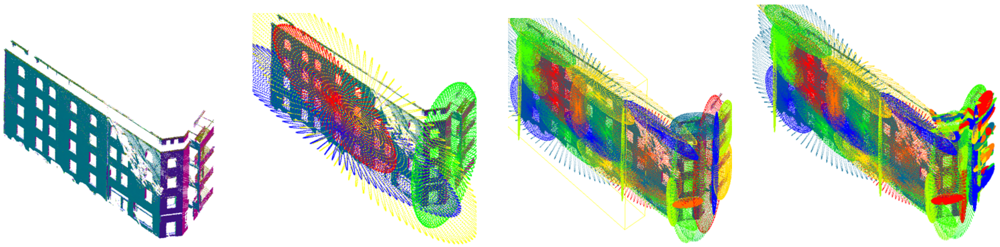
Given a perfect K, the components will properly approximate the object surface. Assuming the surface can be decomposed with many planar patches, each component estimates one planar patch, and the 3D manifold degenerates into a nearly 2D structure. This implies that performing an eigenvalue decomposition on the covariance matrices of the respective components results in their smallest eigenvalues approaching zero.

However, a 3D GMM cannot completely degenerate into 2D. Instead, what can be achieved is the optimization of the model by targeting very small values for the principal curvatures or the eigenvalues associated with the smallest dimension. It is non-trivial to select the proper number of GMM components to achieve this goal. The idea of HGMM is to create a hierarchical tree structure for GMM models. This structure initiates modelling with a small number of Gaussian nodes to encapsulate the data. Subsequently, the complexity of each node is evaluated to select the nodes that should be split. Here, their complexity is assessed based on the deviation from a near-2D (or 'flat') structure. Nodes exhibiting a complexity beyond a predefined threshold are then subdivided into more nuanced sub-nodes at the next level of the hierarchy; i.e., new Gaussian components and GMM training. The complexity of the Gaussian node is quantified by the principal curvature, given by:

$$c = \frac{\lambda_n}{\sum_{i=1}^{n} \lambda_i}, \tag{6.2}$$

where λ_i is the i-th eigenvalue of the principal components, λ_n is the smallest eigenvalue. When the smallest eigenvalue approaches zero, it indicates that the variance along one principal axis is minimal. The N-dimensional manifolds are nearly degenerated into (N-1)-dimensional manifolds. However, absolute eigenvalues are sensitive to the scale of the object (e.g., surface size). The principal curvature is introduced as it is dimensionless and scale-invariant, making them more robust across datasets with different spatial resolutions. If the absolute value of the principal curvature is close to zero, the surface can be approximated as planar. In practical experiments, the threshold of the principal curvature is set to the same empirical value as the previous work (Eckart, K. Kim, Troccoli, et al. 2016).

Figure 6.2 visualizes an example where GMM structures gradually split at hierarchical levels. In the lower right regions of the building, where there are some protruding balconies, Gaussian components split from one green Gaussian (level 1) into smaller Gaussians (level 3) to fit the local, smaller surface structures.



(a) Data (b) Level 1 (c) Level 2 (d) Level 3

FIGURE 6.2. Visualization of the error ellipsoids of Gaussian components in HGMM. (a) is the data. (b) shows four initial Gaussians, and they split into the next level in (c). The new components are checked by the principal curvature to see if they should be further split. (d) presents the final models.

Despite the extension of the hierarchical model from 3D to 4D and the consequent changes in the interpretations of complexity and split criteria, these criteria remain applicable and suitable in the 4D context. If the smallest eigenvalue approaches zero, the same dimension degeneration will occur. This condition typically leads to a tighter correlation between the variables involved. Consequently, in our case, the regression of the fourth dimension (surface distances) based on point coordinates will likely exhibit less noise and can be expected to be more accurate. In addition, it often forces the first 3D variables to have a more flat structure as well.

6.2.2 Surface Distance Regression

To infer the surface distance with HGMM, the original 3D HGMM is extended to the 4D space, and the GMR technique is employed. Given LiDAR measurements $\mathbf{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^{N}$, the 3D variables (coordinates) are appended by a fourth dimension representing surface distances, denoted as $\boldsymbol{Y} = \{y_i\}_{i=1}^{N}$, forming the complete dataset for training: $\boldsymbol{D} = (\mathbf{X}, \mathbf{Y})$. Following the EM training, the parameters $\Theta = \{(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}_{k=1}^{K}$ are derived and each k-th component is a joint Gaussian, given by:

$$\boldsymbol{\mu}_k = \begin{bmatrix} \boldsymbol{\mu}_X \\ \mu_Y \end{bmatrix}_k, \boldsymbol{\Sigma}_k = \begin{bmatrix} \boldsymbol{\Sigma}_{XX} & \boldsymbol{\Sigma}_{XY} \\ \boldsymbol{\Sigma}_{YX} & \boldsymbol{\Sigma}_{YY} \end{bmatrix}_k, \tag{6.3}$$

$$\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1} = \begin{bmatrix} \boldsymbol{\Lambda}_{XX} & \boldsymbol{\Lambda}_{XY} \\ \boldsymbol{\Lambda}_{YX} & \boldsymbol{\Lambda}_{YY} \end{bmatrix}_k. \tag{6.4}$$

where $\boldsymbol{\Lambda}_k$ is the precision matrix.

In GMM regression, the rule of conditionals of multivariate Gaussian distributions is applied to estimate the queried surface distance conditioned on the 3D coordinates. For each component, the posterior conditional distribution reads as:

$$p(y|\boldsymbol{x}) = \mathcal{N}(y|\mu_{Y|\boldsymbol{X}}, \Sigma_{Y|\boldsymbol{X}}), \tag{6.5}$$

$$\mu_{Y|\boldsymbol{X}} = \mu_Y - \boldsymbol{\Lambda}_{\boldsymbol{YY}}^{\boldsymbol{-1}} \boldsymbol{\Lambda}_{YX}(\boldsymbol{x} - \boldsymbol{\mu}_X), \tag{6.6}$$

$$\Sigma_{Y|\boldsymbol{X}} = \boldsymbol{\Lambda}_{\boldsymbol{YY}}^{\boldsymbol{-1}}, \tag{6.7}$$

where $\mu_{Y|\boldsymbol{X}}$ can be denoted as $\mu_k(\boldsymbol{x}_*)$, representing the expected surface distance for the queried point $\boldsymbol{x}_*$, and $\sigma_k^2(\boldsymbol{x}_*) = \Sigma_{Y|\boldsymbol{X}}$ is the associated uncertainty.

The overall surface regression is the mixture of different Gaussian components. In fact, not all components in the GMM have effective impacts on the estimation of a queried area; rather, only some neighboring Gaussians are important, and we call these components the active components. Thus, J components will be selected as active components according to the marginal probability density $p_j(\boldsymbol{x}_*|\boldsymbol{\mu}_{jX}, \boldsymbol{\Sigma}_{jXX})$.

Mixing all active Gaussian components in GMM, the regression is a mean function of all active $\mu_j(\boldsymbol{x}_*)$ averaged with the mixing weights $w_j(\boldsymbol{x}_*)$, or responsibilities, given by (Sung 2004):

$$m(\boldsymbol{x}_*) = \sum_{j=1}^{J} w_j(\boldsymbol{x}_*)\mu_j(\boldsymbol{x}_*), \tag{6.8}$$

$$w_j(\boldsymbol{x}_*) = \frac{\pi_j p_j(\boldsymbol{x}_*)}{\sum_{i=1}^{J} \pi_i p_i(\boldsymbol{x}_*)}, \tag{6.9}$$

and the uncertainty of the expected surface distances is specified as the variance or standard deviation:

$$\sigma^2(\boldsymbol{x}_*) = \sum_{j=1}^{J} w_j(\boldsymbol{x}_*)\{\sigma_j(\boldsymbol{x}_*)^2 + \mu_j(\boldsymbol{x}_*)^2\} - m(\boldsymbol{x}_*)^2. \tag{6.10}$$

Although the hierarchical strategy in HGMM helps to adaptively find the number of Gaussian components and quickly optimize them, the performance is still sensitive to the thresholds of the complexity and the size of the modelling objects. Using a big threshold of the principal curvature sometimes ignores the details of large objects, while it might oversplit the nodes with a small threshold. Also, one single threshold may affect the ability to fit the scene with both large and small objects, and the threshold can become too small with a higher hierarchy of smaller nodes. Therefore, HGMM is an approximation of the ideal GMM with the best fidelity, which can be further improved. Rather than selecting the perfect number of components, GP-based mapping is followed to refine the results from HGMM. More specifically, parameters optimized in HGMM are used as priors for GP training, which will be introduced in section 6.3.

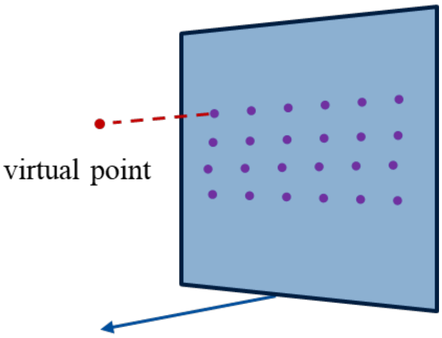


FIGURE 6.3. Virtual point generation

6.2.3 Data Augmentation

As illustrated above, the signed distance value y of each data point is required in the GMM training process. However, the measurements are all collected when the scanning ray hits the surface, i.e., $y = 0$ for all measurements. To obtain valid covariances between spatial coordinates and surface distances, the original dataset is augmented with virtual points along the normal direction of the hit points and at a small distance from the surface. As illustrated in Figure 6.3, the red dot represents the virtual points, and the purple points are the actual measurement points. The dashed line denotes the virtual distance along the direction of the surface normal. An ablation study is presented in Section 6.6.3 to analyze the impact of this virtual distance. The extension of GMM dimension for map regression was originally proposed in the prior work (Srivastava and Michael 2018). The key differences are: 1) the proposed method applies a "top-down" strategy with faster computation speed; 2) SDF is regressed instead of occupancy.

In the practical implementation of HGMMs, we augment the original observations by introducing virtual points adjacent to either side of the surface interfaces. To enhance the efficiency of the training process, we employ a strategy where the point clouds are sub-sampled, and virtual points are subsequently generated based on these reduced data sets. While it is intuitively assumed that denser datasets generally yield superior performance due to more comprehensive spatial information, our experimental results indicate only marginal variations in performance across the different sub-sampling resolutions tested. (We have conducted this ablation study and displayed the results in Section 6.6.3.) This suggests that the HGMM methodology exhibits considerable robustness to data sparsity, maintaining effectiveness even with significantly thinned datasets. Such robustness is advantageous for applications where acquiring dense point cloud data is impractical or resource-intensive.

In the process of data augmentation, another critical factor is the spacing (virtual distance) between virtual points and actual hit points on the surface, which should be set based on multiple considerations, including the model's accuracy and its ability to infer distances along the normal to the surface. The sub-sampling rate primarily addresses the resolution on the surface, reflecting surface detail retention in the processed dataset. In contrast to the sub-sampling resolutions, the spacing influences the extent of the model's predictive capability in the direction normal to the surface. As the distance between a virtual point and the actual surface increases, the confidence in the distance field of the virtual point diminishes, suggesting that excessively large spacing is undesirable. Conversely, overly small spacing can limit the model's capacity to estimate the distance field over larger spatial extents. Empirical observations have shown that reducing the spacing between points can improve the model's precision near the surface but may compromise accuracy at further distances. Thus, selecting the optimal spacing involves balancing these considerations to achieve a desirable trade-off between near-surface accuracy and the ability to model distance gradients effectively over larger areas.

An alternative option involves incorporating virtual points at multiple spacings, ranging from small to relatively large distances from the actual hit points. This method enhances the training dataset by providing a broader spectrum of spatial relationships, potentially improving both the model's accuracy and ability to further predictive distances. However, the expansion in dataset size consequent to this strategy may impact the model's computational efficiency. Consequently, while this approach can enrich the training data and potentially refine the model's performance, it also requires careful management of the computational resources to maintain a balance between accuracy and efficiency in model training.

6.3 GP Inference with HGMM as Priors

6.3.1 GP Regression with Non-stationary Priors from HGMM

Supposing a training dataset with location inputs $\mathbf{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^N$ and the corresponding target distance values $\boldsymbol{y} = \{y_i\}_{i=1}^N$ measured with additive noise $\eta_i \sim \mathcal{N}(0, \sigma_\eta^2)$. The inference of the testing space $\mathbf{X}_*$ can be estimated from the posterior GP mean and kernel (Rasmussen and C. K. I. Williams 2006):

$$\mu(\boldsymbol{x}_*) = \mu_0(\boldsymbol{x}_*) + \mathbf{k}_*^T(\mathbf{K}_N + \sigma_\eta^2\mathbf{I})^{-1}(\boldsymbol{y} - \mu_0(\mathbf{X})), \tag{6.11}$$

$$k(\boldsymbol{x}_*, \boldsymbol{x}'_*) = k_0(\boldsymbol{x}_*, \boldsymbol{x}'_*) - \mathbf{k}_*^T(\mathbf{K}_N + \sigma_\eta^2\mathbf{I})^{-1}\mathbf{k}'_* + \sigma_\eta^2, \tag{6.12}$$

where $\mu_0(\boldsymbol{x})$ and $k_0(\boldsymbol{x}, \boldsymbol{x}')$ are denoted as the prior mean and kernel function. $\mathbf{K}_N = k_0(\mathbf{X}, \mathbf{X})$ is the covariance matrix of the N input points calculated by the prior kernel function. $[\mathbf{k}_*]_N = k_0(\mathbf{X}, \boldsymbol{x}_*)$ and $[\mathbf{k}'_*]_N = k_0(\mathbf{X}, \boldsymbol{x}'_*)$ are the prior covariance between the N input training points and the predicting points.

In a standard stationary GP, the prior mean function is often set to a constant number, e.g., zero, and the kernel function depends only on the distance between points in the input space and not on the specific locations of those points, e.g., the squared exponential kernel or Matérn kernel. For instance, a 3/2 Matérn kernel is used as the prior GP kernel function.

As a stationary kernel, the hyper-parameters σ_p and l in the above equation should remain invariant at different input locations. In this thesis, however, the prior mean and prior variance come from the outcomes of HGMM, which vary across different inputs. It indicates that the stationary GP assumption does not hold anymore and the GP regression transforms into a non-stationary case.

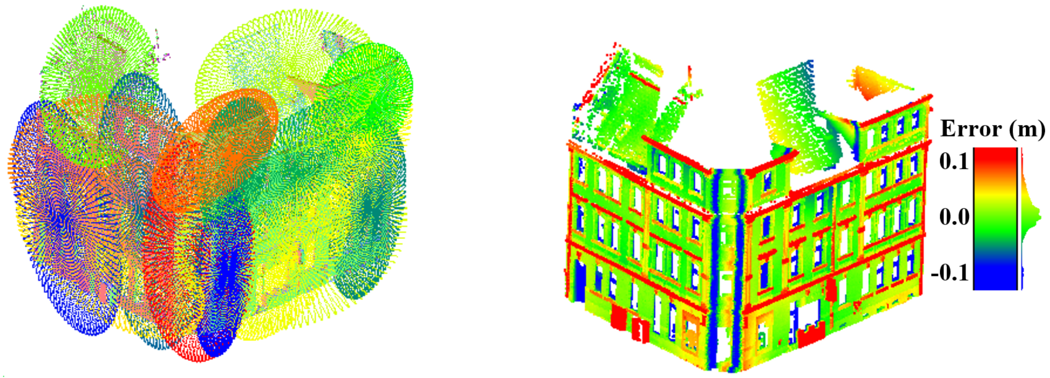
To integrate both methodologies seamlessly, we utilize surface distance estimation derived from HGMM as the prior mean for GP. The GMM regression function (Equation (6.8)) is employed as the prior mean function during the training of the GP: $\mu_0(\boldsymbol{x}) = m(\boldsymbol{x})$.

Accordingly, the estimated standard deviation computed in Equation (6.10) serves as σ_p in the kernel function. Since the hyper-parameter σ_p varies with different inputs, Equation (5.4) also changes into:

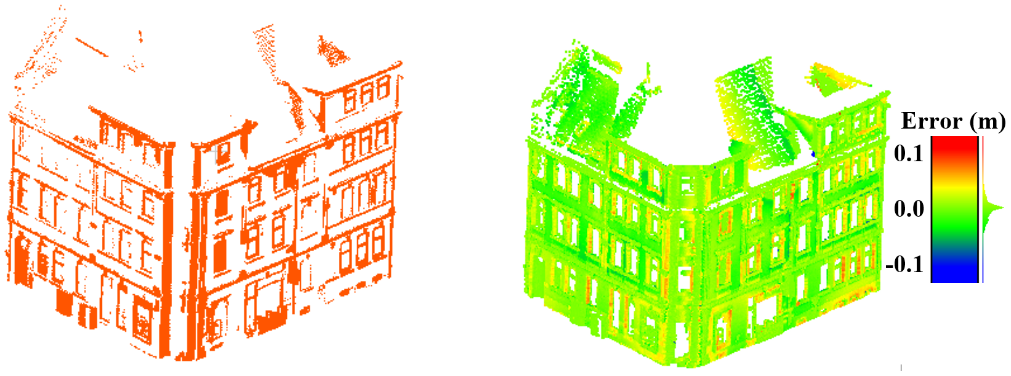
$$k_0(\boldsymbol{x}, \boldsymbol{x}') = \sigma_p \sigma_p' \left(1 + \frac{\sqrt{3}\,|\boldsymbol{x} - \boldsymbol{x}'|}{\ell}\right) \exp\left(-\frac{\sqrt{3}\,|\boldsymbol{x} - \boldsymbol{x}'|}{\ell}\right), \tag{6.13}$$

where σ_p is the prior standard deviation and l is the length scale parameter.

To fully exploit the HGMM results obtained from HGMMs and reduce the model size of GP training, we select only the points that are not well-modelled on the surface to be



(a) GMM components (b) GMM surface



(c) Training points (d) GP surface

FIGURE 6.4. Selected training points for GP: (a) shows 16 Gaussian components of GMM. (b) presents the differences between estimated signed distances and measured surface points. The green color indicates small errors as shown in the color bar. The points with large errors are selected for GP training in (c). (d) shows the GP inference results.

the training points, as shown in Figure 6.4. The selection criteria are given by:

$$|m(\boldsymbol{x})| > d_m, \tag{6.14}$$

where d_m is the allowed maximal discrepancy between the real observation and the HGMM estimation of surface distance. As measurements hit the surface, the real observation is always zero. This criterion indicates the HGMM estimation should be smaller than a given threshold. Otherwise, the point will be selected for GP training. This value is dependent on the required accuracy for the surface.

6.3.2 GP with Derivative Observations

Leveraging derivatives of the targets (distance fields) presents a potentially valuable approach for enforcing known constraints (Rasmussen and C. K. I. Williams 2006); i.e., a more robust inference of the signed distance fields can be made based on the gradients of the distances. In this case, the normal vectors of the points represent the derivative observations at the local surface. The distances increase along the normal direction.

Therefore, we augment the original 4D training dataset with additional normal information about each point, yielding a new dataset $\mathcal{D} = \{\boldsymbol{x_i}, y_i, \nabla \boldsymbol{y_i}\}_{i=1}^{N}$. The inference can be made based on the joint Gaussian distribution of the signed distances and the gradients of distances. Accordingly, the joint covariance matrix $\tilde{k}(\boldsymbol{x}, \boldsymbol{x}')$ involves $N(3+1)$ entries (L. Wu, K. M. B. Lee, Gentil, et al. 2023; Rasmussen and C. K. I. Williams 2006):

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}') & k(\boldsymbol{x}, \boldsymbol{x}')\nabla_{\mathbf{x}'}^{\top} \\ \nabla_{\mathbf{x}} k(\boldsymbol{x}, \boldsymbol{x}') & \nabla_{\mathbf{x}} k(\boldsymbol{x}, \boldsymbol{x}')\nabla_{\mathbf{x}'}^{\top} \end{bmatrix}, \tag{6.15}$$

where $\nabla_{\mathbf{x}} k(\boldsymbol{x}, \boldsymbol{x}')$ is the gradient of the covariance function with respect to $\boldsymbol{x}$, i.e., the covariance between target values and the partial derivatives. $\nabla_{\mathbf{x}} k(\boldsymbol{x}, \boldsymbol{x}')\nabla_{\mathbf{x}'}^{\top}$ is the covariance between partial derivatives.

The gradients or the normal vectors are estimated by PCA for each point. Given the training data $\mathcal{D}$ with derivative information, we can change the standard inference, substituting $k_0(\cdot, \cdot)$ in Equation (6.11) and (6.12) by Equation (6.15):

$$\mu(\boldsymbol{x}_*) = \begin{bmatrix} m(\boldsymbol{x}_*) \\ \nabla m(\boldsymbol{x}_*) \end{bmatrix} + \tilde{\mathbf{k}}_*^T(\tilde{\mathbf{K}}_N + \sigma_\eta^2 \mathbf{I})^{-1} \begin{bmatrix} \boldsymbol{y} - m(\mathbf{X}) \\ \nabla \boldsymbol{y} - \nabla m(\mathbf{X}) \end{bmatrix}, \tag{6.16}$$

$$k(\boldsymbol{x}_*, \boldsymbol{x}_*') = \tilde{k}(\boldsymbol{x}_*, \boldsymbol{x}_*') - \tilde{\mathbf{k}}_*^T(\tilde{\mathbf{K}}_N + \sigma_\eta^2 \mathbf{I})^{-1}\tilde{\mathbf{k}}_*' + \sigma_\eta^2, \tag{6.17}$$

The technique of GP inference with derivative observations has been employed in the existing work (B. Lee et al. 2019; L. Wu, K. M. B. Lee, Gentil, et al. 2023). They applied an additional 2D observation GP and occupancy tests to estimate the normals of measurement points. Unlike that, normal vectors are estimated by PCA directly here.

6.3.3 Integration with LogGP

The integration framework that combines GMM with GP extends beyond conventional GP models to encompass various GP variants, such as the approach involving logarithmic transformations of GP inference - LogGP (L. Wu, K. M. B. Lee, Gentil, et al. 2023). In the specific case of LogGP, the model dynamics involve transforming the inferred values from a standard GP. Here, the heat model denoted as v is directly inferred using the posterior of a GP. Subsequently, the Euclidean distance field d is derived from v through a logarithmic transformation, mathematically represented as::

$$d = -\sqrt{t}\log(v) \tag{6.18}$$

where t is a smoothing hyper-parameter.

Based on the gradient of the distance field d with respect to the heat model v, the variance of the distance field can be formulated as follows:

$$\mathbb{V}[d] = \frac{t}{v^2}\mathbb{V}[v], \tag{6.19}$$

where $\mathbb{V}[\cdot]$ denotes the variance.

To effectively incorporate a GMM as a prior within the LogGP framework, it is necessary to transform the distance fields and their uncertainties generated by the GMM into a format as the heat model by an exponential function, reversing the logarithmic transformation used in LogGP:

$$v_0 = \exp\frac{-|d_0|}{\sqrt{t}}, \tag{6.20}$$

$$\mathbb{V}[v_0] = \frac{v^2}{t}\sigma_0^2 \tag{6.21}$$

where v_0 serves as the prior mean function in the GP inference of the heat model $v \sim \mathcal{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$, which is calculated based on the prior distance d_0 and its prior standard deviation σ_0 obtained from GMMs.

After this transformation, the prior heat model can be integrated into the GP inference with the same mechanism as the normal GP in Section 6.3.2, substituting $m(\cdot)$ and $\nabla m(\cdot)$ in Equation (6.16) and (6.17) by $v(\cdot)$ and $\nabla v(\cdot)$. Finally, the inferred heat model is converted back to the distance model using Equation (6.18) and (6.19).

6.3.4 Octree-based GP Block

The use of GPs for inference, especially when incorporating derivatives, presents significant scalability challenges with large datasets due to the computational complexity, which scales cubically with both the number of data points N and the dimensionality D, represented as $\mathcal{O}(N^3D^3)$. While the adoption of GMM priors can effectively reduce the requisite number of training points, enhancing scalability, further optimizations remain necessary. Recognizing that distant points exert minimal influence on local GP inference outcomes, a practical approach in fully 3D scenarios involves segmenting the data into blocks using an Octree structure (Hornung et al. 2013). This partitioning facilitates more localized and efficient processing of data clusters, significantly mitigating the computational demands by circumventing the extensive cubic complexity associated with processing all data points simultaneously.

6.4 Uncertainty in SDF Estimation

There are two inherently different sources of uncertainty, which are often referred to as aleatoric and epistemic uncertainty. This section aims to clarify the distinctions between aleatoric and epistemic uncertainty within the contexts of SDF estimation using HGMMs and GPs, emphasizing their relevance and operational mechanisms.

Epistemic uncertainty reflects the similarity between the query point and the training data, i.e., their distance in the input space or feature space. In the context of GMMs, the similarity between training and query data can be quantified by the overall probability density or the distances from the query point to the means of the GMM components, which encapsulate the generalized information of the training data. For GPs, epistemic uncertainty is an inherent property of the posterior distribution based on noise-free observations, determined by the distances between inputs and training data and the corresponding kernel function. In contrast, the aleatoric uncertainty of GPs is captured by adding a noise term to the model's likelihood.

In this research, GMR is first utilized to estimate surface distances before GP inference, where the inherent variance of the regression, due to the distributional nature of GMMs, is naturally computed in Equation (6.10). However, this variance does not represent epistemic uncertainty within the GMM framework. The regressed distance is the expectation of the distance distribution conditioned on the 3D spatial coordinates and does not account for whether the 3D point is in/out of the GMM distribution. Instead, it represents aleatoric uncertainty as it does not vary with the similarity or distance between the query point and the training samples.

The actual epistemic uncertainty measure of GMR can be indicated by the overall density of a GMM, considering all components with the corresponding weights. A notable increase in epistemic uncertainty occurs in out-of-distribution (OOD) scenarios, where the query point deviates significantly from any training data, highlighting substantial uncertainty due to a lack of knowledge. In this work, the Mahalanobis distance is computed to test if the query point ($\boldsymbol{x} \in \mathbb{R}^3$) is OOD. If so, we are not confident in the whole regression, where the regression uncertainty (Eq. (6.10)) does not make sense. However, the epistemic uncertainty here only indicates how confident we are about the regression, but it cannot provide a direct statistical quantitative uncertainty measure for the surface distance, e.g., the variance of the regressed distance. That is the reason why a default large uncertainty of the regression is set for OOD.

Unlike that, the epistemic uncertainty in GP can directly represent the quantitative uncertainty for the regressed surface distance. However, a prior variance is also required for the OOD of GP. When using GMM as prior, the OOD of GP is still in the distribution of GMM and thus the variance from GMM works until it goes further and reaches the space that is OOD of GMM.

6.5 Incremental Update

As there may be new measurements forthcoming, it is essential to establish an effective strategy for incrementally updating the existing 3D map. Given that the map is constructed using probabilistic methods, it is appropriate to update the map in a Bayesian manner, reflecting its probabilistic nature.

Within the framework of the proposed GMMGP surface mapping, the process of updating the map draws inspiration from prior studies (Engel and Heinen 2010; Srivastava and Michael 2018). The updates are conceptualized under two distinct scenarios, with each treating the existing map as a probabilistic distribution:

- **Update case A - updating Previously Unknown Areas**: When new measurements introduce completely novel information, typically in previously unexplored areas, it may be necessary to integrate new Gaussian components. These new components are designed to represent and model these newly observed regions. This process aligns with methodologies discussed in Section 6.2, where the introduction of new components accommodates the expansion of the spatial model to incorporate fresh data inputs.

- **Update case B - refining Existing Distribution Parameters**: In situations where new data are collected in areas that have already been mapped, the existing distribution parameters require updates to reflect the new information. In this scenario, the parameters of the old distribution are adjusted to better fit the combined data set, enhancing the accuracy and detail of the map in these known regions. This update process must handle the potential ambiguities arising from integrating the old and new datasets, doing so in a probabilistic manner to maintain the integrity and reliability of the model.

  To address the above issue while updating the existing GMM, we adopt an incremental EM optimization strategy. Incremental GMM parameter optimization follows an adapted version of the traditional EM algorithm, enabling real-time updates of mixture model parameters as new data is introduced.

  Similar to the standard EM procedure, the update begins with an Expectation (E) step, where the posterior probabilities of the mixture components for a given data point, also referred to as responsibilities, $p(z \mid x)$, are computed. These are obtained using Bayes' theorem, leveraging the current conditional probability $p(x \mid z)$ along with the prior probabilities of the components. In the subsequent Maximization (M) step, the parameters of the Gaussian mixture model—including the updated means

($\boldsymbol{\mu}'$), covariance matrices ($\boldsymbol{\Sigma}'$), and mixing coefficients ($\boldsymbol{\pi}'$)—are refined to better represent the updated data distribution. The updated parameters can be expressed as:

$$N'_k = N_k + \sum_{i=1}^{N_r} \gamma(z_{ik}), \tag{6.22}$$

$$\boldsymbol{\mu}'_k = \frac{1}{N'_k}(N_k \boldsymbol{\mu}_k + \sum_{i=1}^{N_r} \gamma(z_{ik}) \boldsymbol{x}'_i), \tag{6.23}$$

$$\boldsymbol{\Sigma}'_k = \frac{1}{N'_k}(N_k \boldsymbol{\Sigma}_k + \sum_{i=1}^{N_r} \gamma(z_{ik})(\boldsymbol{x}'_i - \boldsymbol{\mu}'_k)(\boldsymbol{x}'_i - \boldsymbol{\mu}'_k)^T), \tag{6.24}$$

$$\pi'_k = \frac{N'_k}{N + N_r}, \tag{6.25}$$

where N and N_r represent the number of training points in the existing map and the newly acquired measurements, respectively. The term γ_{ik}denotes the responsibility assigned to component k for the i-th newly observed point, computed as per Equation 2.37 in Chapter 2. N_k is the effective number based on the current parameters of the existing model, given by $N_k = N\pi_k$. Similarly, N'_k denotes the updated effective number, incorporating the influence of new measurements.

Both scenarios necessitate a careful consideration of how new data influence the existing probabilistic models, ensuring that the updates not only expand the map's coverage but also refine its details and accuracy, thereby enhancing the overall utility and reliability of the 3D map.

Figure 6.5 illustrates the process of incremental map updating, comparing the existing map with the updated version incorporating new measurements. Figure 6.5a presents the initial reconstructed model, which exhibits significant missing regions, particularly on the right façade and the upper left corner of the central façade, where occlusions prevent complete data acquisition. The introduction of new measurements successfully covers these previously missing areas.

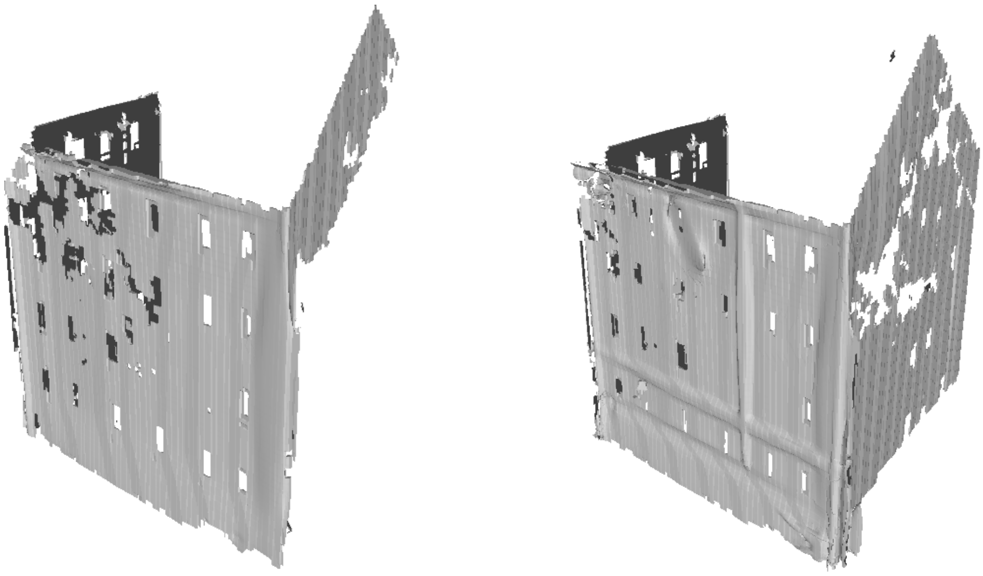
By integrating the newly acquired data, the existing map is incrementally updated, resulting in a more complete and refined reconstruction, as shown in Figure 6.5b. Beyond filling in missing sections (Update case A), the update also enhances the level of detail in previously captured regions, particularly on the central façade, where finer structural features become more discernible (Update case B).

Figures 6.5c and 6.5d depict the uncertainty estimates associated with the reconstructed surfaces, where the color encoding represents the magnitude of uncertainty. These visualizations provide insight into how the newly incorporated measurements influence both the completeness and the confidence of the reconstructed model.

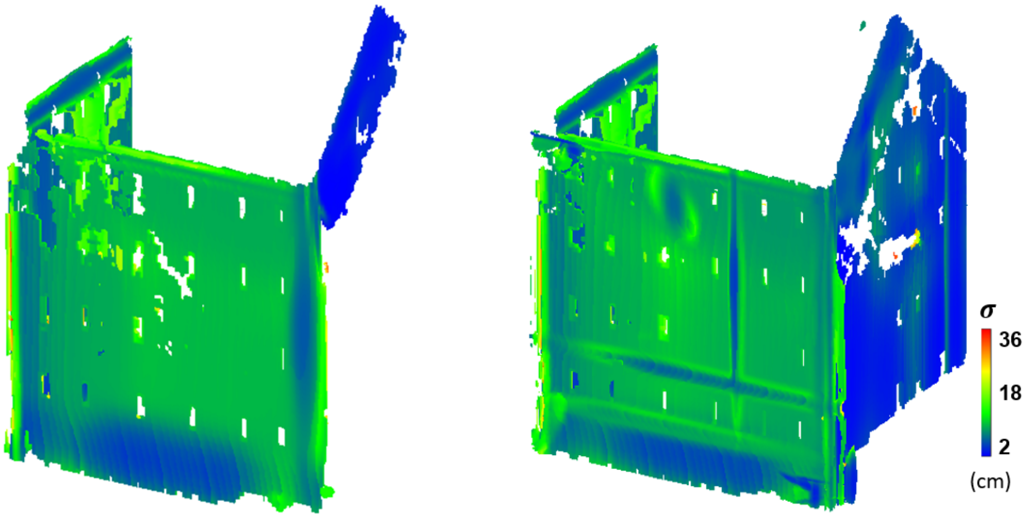
6.6 Experiments

6.6.1 Experimental Setting

To validate our method, we conducted evaluations on complex 3D urban environments using the real-world LUCOOP LiDAR dataset and the synthetic CARLA dataset (see Chapter 4). The real-world data offers a robust basis for assessing the model's performance in natural settings. The simulated dataset supplements this with idealized buildings and



(a) Surface reconstructed from scan 1 (b) Existing surface updated with scan 2



(c) Uncertainty of the surface (d) Uncertainty of the updated surface

FIGURE 6.5. Update of the map surface

a manually checked ground truth. The ground-truth SDF is established and explained in the Section 4.2.4

6.6.2 Evaluation Metrics

For the evaluation of our model, two key metrics were employed to assess performance quantitatively: 1) Root Mean Squared Error (RMSE) for the accuracy of the distance field; 2) Predictive log-likelihood for the uncertainty evaluation.

RMSE is used to evaluate the accuracy of the estimated distance field. RMSE quantifies the average magnitude of the error between the predicted distances from our model and the ground truth distances across a set of testing points. The formula for RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{N_*}\sum_{i=1}^{N_*}(\mu_i - \hat{\mu}_i)^2} \tag{6.26}$$

where N_* is the number of the testing points $\mathbf{X}_*$, μ_i is the ground truth distance, $\hat{\mu}_i$ is the posterior estimation from Equation (6.16). A lower RMSE value indicates higher accuracy in distance estimation.

Predictive log-likelihood is a widely utilized metric for assessing the quality of uncertainty estimates in probabilistic models. It measures the extent to which the estimated probability distribution conforms to the true distribution observed in the data. This metric evaluates not only the discrepancies between the estimated values and the ground truth but also considers the model's estimated variance or covariance. Importantly, a model that either underestimates or overestimates the uncertainty associated with its predictions will incur penalties under this metric, as it fails to accurately represent the error distribution between the predicted and actual values. A higher predictive log-likelihood value indicates superior performance in uncertainty quantification, suggesting that the model's uncertainty estimates are both appropriate and informative. In the context of this study, the log-likelihood values are normalized by the number of testing samples to facilitate comparison across different datasets or model configurations. The mathematical formulation for this normalized predictive log-likelihood is expressed as:

$$LL = \frac{1}{N_*}\sum_{i=1}^{N_*}\log p(\mu_i|\hat{\mu}_i, \hat{\sigma}_i^2) \tag{6.27}$$

where $\hat{\sigma}^2$ is the posterior uncertainty obtained from Equation (6.17).

6.6.3 Ablation Study

As outlined in Section 6.2.3, the dataset undergoes augmentation to facilitate the development of a valid 4D GMM for SDF regression. This process involves sub-sampling the point clouds to strategically generate virtual points at predefined virtual spacings from the query points. In this section, we present a series of ablation studies designed to evaluate the impact of various sub-sampling resolutions and virtual distances on the performance of the SDF regression model. These experiments are crucial for determining the optimal parameters that balance computational efficiency with the accuracy of the regression model. The findings will help elucidate how changes in sub-sampling strategies and virtual spacing influence the model's effectiveness.

To systematically assess the impact of varying sub-sampling resolutions on model performance, we conducted experiments using sub-sampling resolutions ranging from 0.02 to 0.5 meters. This resolution represents the minimal distance between points, reflecting the point density after sub-sampling. The results of these experiments are depicted in Figure 6.6, where each sub-sampling resolution is represented by a distinct color. The experimental findings indicate minimal variation in performance across the tested sub-sampling resolutions. This consistency underscores the robustness of the HGMM approach, demonstrating its substantial resilience to variations in data density. Such robustness is particularly advantageous in applications where data may be sparse or unevenly distributed, suggesting that the HGMM methodology can reliably perform under diverse operational conditions without significant loss of accuracy.

Figure 6.7 displays the performance of the GMM-based SDF regression evaluated over a range of spacing distances from 0.02 to 0.5 meters. The figure employs a color-coded scheme to delineate different spacing intervals, with the purple curve specifically

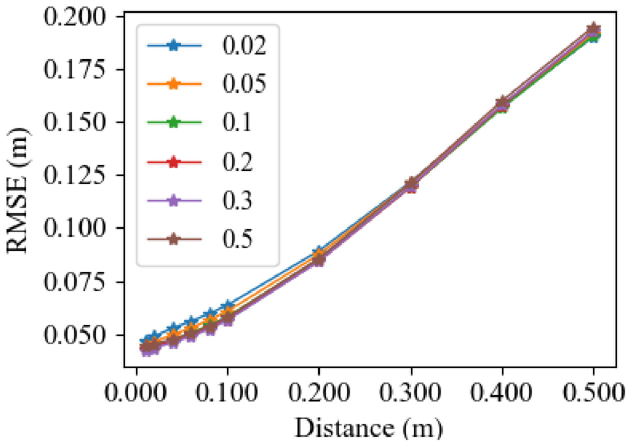


FIGURE 6.6. RMSE with different sub-sampling resolutions

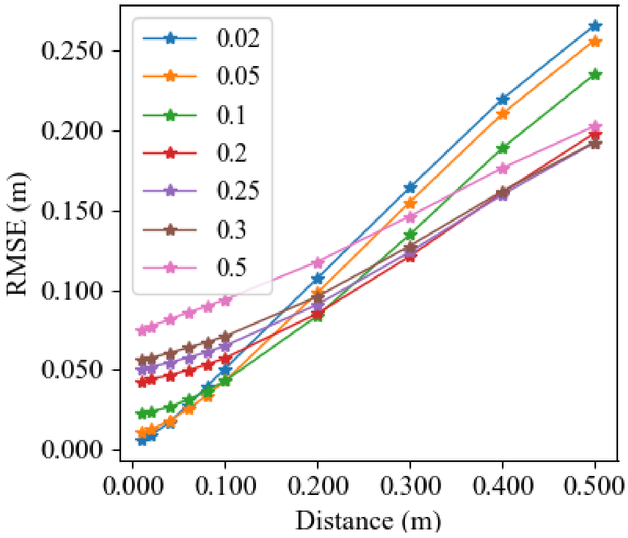


FIGURE 6.7. RMSE with different spacing distances

illustrating the performance at the spacing distance of 0.25 meters used in our experiments. The result demonstrates a distinct trade-off in model performance relative to spacing distance: smaller spacing intervals lead to higher accuracy at proximal distances but result in decreased performance at more considerable distances. Conversely, larger spacing intervals enhance the regression accuracy at larger distances but reduce accuracy at closer proximity.

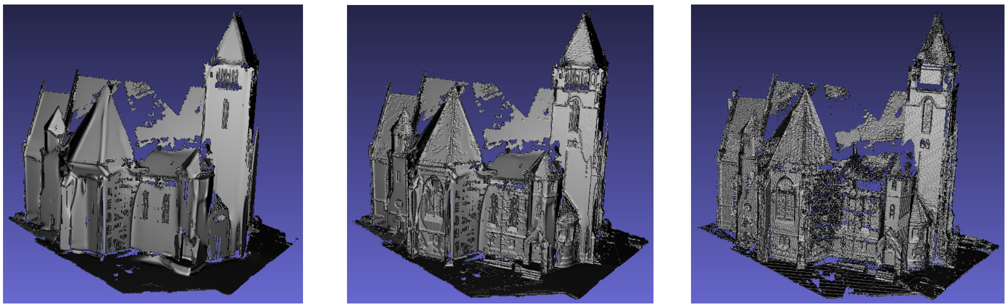
6.6.4 Surface Reconstruction

The implicit surface representation provided by the SDF is converted into an explicit surface mesh through the application of the Marching Cubes algorithm (Lorensen and Cline 1998). This algorithm facilitates the extraction of a surface mesh by identifying and rendering the zero crossings within the SDF, where the sign of the distance field changes, indicating the boundary of the object.

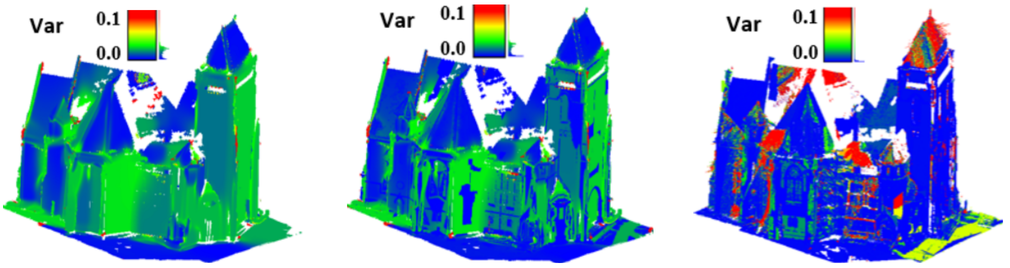
We conducted a comparative analysis of surface reconstructions derived from different modeling approaches: 1) employing HGMM regression alone; 2) utilizing a comprehensive 3D mapping framework that integrates HGMM and GP, denoted as GMMGP; and 3) applying a previous mapping method (Chapter 5) that combines GMM and GP within 2.5D local frames (GMMGP2.5D). Figure 6.8 showcases the qualitative results of this surface reconstruction, accompanied by the corresponding uncertainties associated with each surface estimation. The visualization is organized into two rows for clarity: the first row displays the estimated surfaces, and the second row represents the uncertainties related to these surfaces. The uncertainty values are encoded using a color gradient where warmer colors (redder) indicate higher uncertainty levels, and cooler colors (bluer) signify lower uncertainty levels, as shown in the color bar.

Notably, the surfaces derived from the integrated GMMGP approach exhibit significant advancements in surface detail and accuracy, as shown in Figure 6.8b, when compared to surfaces reconstructed using HGMM alone (Figure 6.8a). In the HGMM-only approach, surface regions lacking in detail correlate with higher uncertainty. Both GMMGP and GMMGP2.5D frameworks capture finer details than the HGMM-only approach, as they integrate GPs to model local, fine-grained surface structures. In particular, the GMMGP framework effectively improves the accuracy and reduces uncertainty in these poorly detailed areas. This improvement is evident in the broader areas of surfaces displayed in the GMMGP outputs, which exhibit lower uncertainty levels, signaling a boost in the model's overall reliability and predictive confidence.

Additionally, the previous GMMGP2.5D method introduced in Chapter 5 uses segmented local 2.5D frames, resulting in increased surface discontinuity in reconstruction. It performs better in primarily planar areas but occasionally presents discontinuous reconstruction errors. Moreover, it generally underestimates the uncertainty relative to HGMM and GMMGP.



(a) Surface of HGMM (b) Surface of GMMGP (c) surface of GMMGP2.5D



(d) Variance of the HGMM surface (e) Variance of the GMMGP surface (f) Variance of the GMMGP2.5D surface

FIGURE 6.8. Qualitative results extracted by the marching cubes algorithm

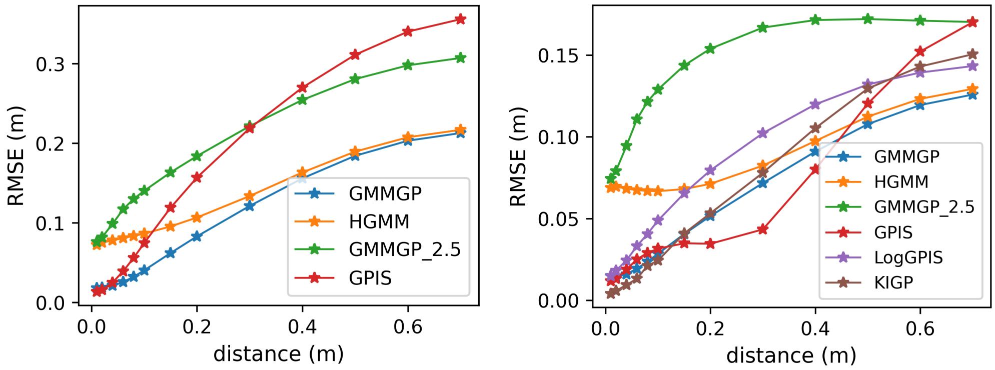
Overall, the GMMGP method not only improves surface detail and accuracy but also enhances the consistency and reliability of the uncertainty estimates, offering a more robust and trustworthy model compared to both the HGMM-alone and the previous GMMGP2.5D methods.

6.6.5 Quantitative Evaluation

Real-world point clouds

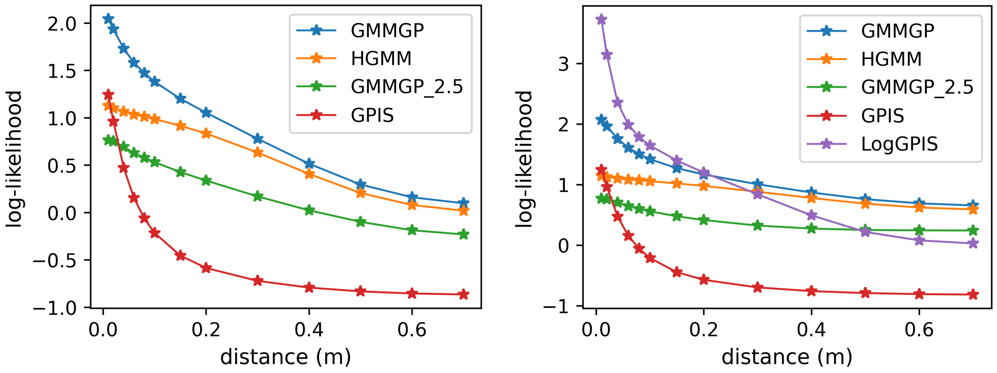
To evaluate both the mapping accuracy and the reliability of the uncertainty measures, the introduced GMMGP method is benchmarked with existing state-of-the-art mapping approaches: GMMGP2.5D (Zou, Brenner, et al. 2023), GPIS map (B. Lee et al. 2019), Log-GPIS (L. Wu, K. M. B. Lee, Gentil, et al. 2023) and Kernel-Inverse GP (KIGP) (Gentil et al. 2024), which all provide uncertainty measures along with the mapping results. The uncertainty measure of KIGP is a proxy and does not align with our quantitative evaluation, hence it is excluded from comparison. We compared the SDF of the testing points between GMMGP2.5D, GPIS and the proposed HGMM and GMMGP. Given that Log-GPIS and KIGP only provide unsigned distance fields, EDF was compared.

Figure 6.9 and 6.10 present a comprehensive comparison of RMSEs and log-likelihoods for estimated surface distances, encompassing both established benchmark methodologies and our proposed GMMGP as well as standalone HGMM mapping. Both metrics are plotted against varying distances between the query points and the original training points. With a growing distance from the observations, a decline in performance is noted across all methods, characterized by increasing RMSEs and diminishing log-likelihoods.



(a) SDF (b) EDF

FIGURE 6.9. Accuracy evaluation with RMSEs.



(a) SDF (b) EDF

FIGURE 6.10. Uncertainty evaluation with Log-likelihood.

In the analysis of the SDF accuracy (Figure 6.9a), GMMGP exhibits the best performance with the smallest RMSEs across the entire range of distances. The discrepancy between GMMGP and HGMM is particularly pronounced in regions proximal to the surface. Here, GMMGP demonstrates an enhanced ability to accurately infer the surface and its adjacent distance fields - a distinction in the qualitative results is compared in Figure 6.8, where Figure 6.8b shows more details on the reconstructed surface than 6.8a. Since the GMMGP2.5D is proposed for structured building models, more arbitrary and complicated shapes can cause improper segmentation for local planes. It shows worse robustness and accuracy than the proposed GMMGP in full 3D. Although GPIS shows accurate inference for query points near the surface, its RMSEs notably increase as the distance from the surface increases. Mirroring this pattern, GMMGP tends to align with the GMM's outcomes for query points located at larger distances, indicating a convergence towards the prior model in areas far from the observational data.

Within the context of EDF outcomes, GMMGP generally achieves higher accuracy. For distances less than 0.1m, the outcomes from the KIGP are marginally superior. At proximities close to 0.2m, GPIS shows the lowest RMSEs. This result arises because GPIS tends to converge towards the prior beyond a certain threshold, with the experimental setup defining this prior distance as 0.2 meters. Given that signs are disregarded in EDF calculations, this prior setting intrinsically aids GPIS in obtaining accurate estimations for query points around the 0.2-meter mark. However, as distance increases, the error associated with GPIS exceeds that of the proposed GMMGP method, highlighting the limitations of relying on the fixed prior in distance estimation tasks.

In the assessment of uncertainty, the proposed GMMGP demonstrates promising performance. When evaluating SDF, both the GMMGP and HGMM outperform GPIS and the previous work GMMGP2.5D, exhibiting higher likelihoods. For EDF, the uncertainty quantification provided by Log-GPIS proves to be more dependable at proximal distances, with log-likelihoods diminishing rapidly as distance increases. Overall, the GMMGP method exhibits the most consistent and robust performance relative to the alternative approaches considered.

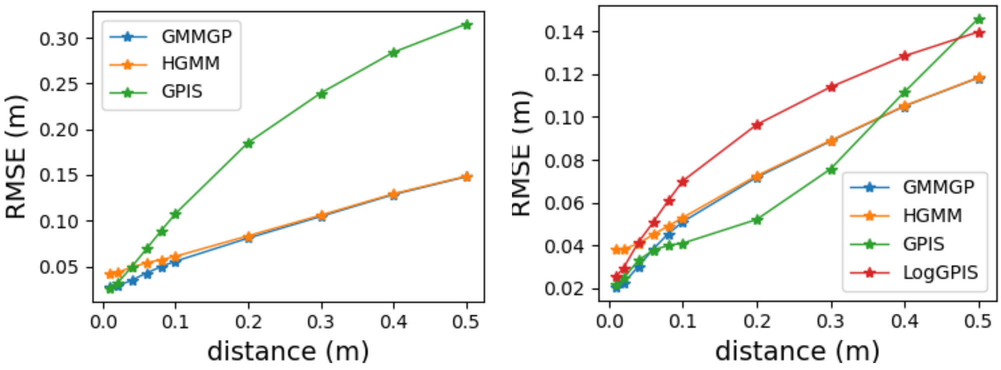
The predetermined priors in GPIS lead to diminished accuracy at larger distances from observed data. To solve this problem, log-GPIS introduced a heat model in the standard GPIS, which modifies the regression target as the exponential transformation of the original surface distances. This converts the target value at the surface from zero into one, and transitions the prior mean from a fixed value to infinity $\sim -\log(0)$. However, Log-GPIS encounters constraints related to the choice of the length scale parameter; e.g., a small length scale parameter leads to more accurate results at the expense of surface smoothness. Additionally, the small length scale parameter also limited its capability to infer further distances. Although KIGP mapping employs kernel reverting functions to improve Log-GPIS, it is not scalable to the large 3D dataset, as it applies full GP inference. Thus, in this case, the data must be partitioned into cells for large urban scenes, which, however, affects its accuracy.

In contrast, our proposed method addresses these challenges by integrating the HGMM to obtain a calibrated prior, and employing local GPISs with non-stationary mean and kernels. This enhances the accuracy of signed distance field estimations and achieves more dependable uncertainty measures.

Simulated Point Clouds

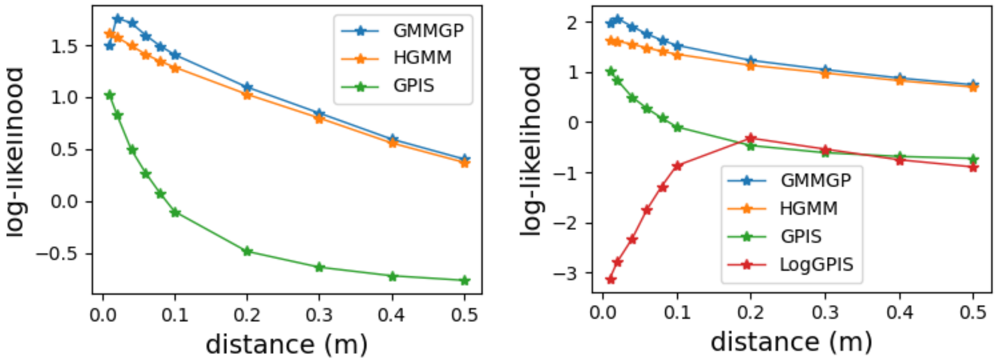
The evaluation of the simulated dataset demonstrates performance metrics that are comparable to those obtained from real-world dense point cloud data. Specifically, the proposed GMMGP method outperforms other techniques in terms of both accuracy and uncertainty quantification. These results are quantitatively illustrated in Figure 6.11, which details the RMSEs across different shortest distances from the testing points to the observation points, and Figure 6.12, which shows the log-likelihood measures of predictive uncertainty.

A notable observation from the analysis is the Log-likelihood performance degradation of the Log-GPIS method at shorter ranges. This degradation is attributed to an underestimation of uncertainty by the Log-GPIS method at closer distances. The tendency of the Log-GPIS method to underestimate uncertainty likely stems from its inherent model estimation procedures, where the GP inferred heat variance has to be further transformed to the actual SDF variance, i.e., multiplied by $\frac{t}{v^2}$, which is an approximation of the non-linear uncertainty propagation of a non-linear function. This indirect estimation might affect the robustness of the uncertainty quantification.



(a) SDF (b) EDF

FIGURE 6.11. Accuracy evaluation with RMSEs.



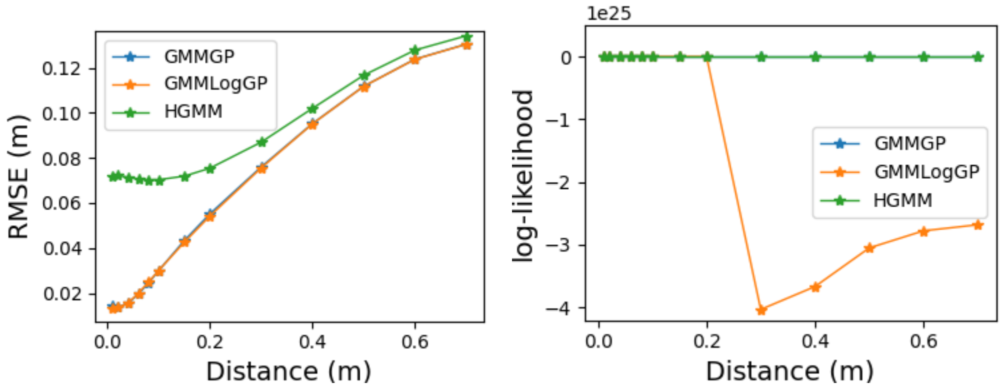
(a) SDF (b) EDF

FIGURE 6.12. Uncertainty evaluation with Log-likelihood.

Integration of LogGP with HGMM

As detailed in Section 6.3.3, the integrated framework involving HGMM and GP extends its applicability to various GP variants. In this section, we conduct an evaluative comparison of mapping results derived from the integration with normal GP and Logarithmic GP (GMMLogGP), as depicted in Figure 6.13. This figure presents the RMSE and Log-likelihood metrics resulting from different GP inferences, employing the same HGMM prior.

The RMSE curves for both GMMGP and GMMLogGP are similar, demonstrating the framework's capability to generalize across different GP variants in terms of EDF accuracy.



(a) EDF (b) Log-likelihood

FIGURE 6.13. Evaluation of the GMMGP framework with LogGP variant.

However, the analysis of log-likelihood reveals that the log-transformed GP inferences exhibit inferior performance. This reduction in performance is likely attributable to the robustness issues associated with the transformation process between heat variances and EDF variances.

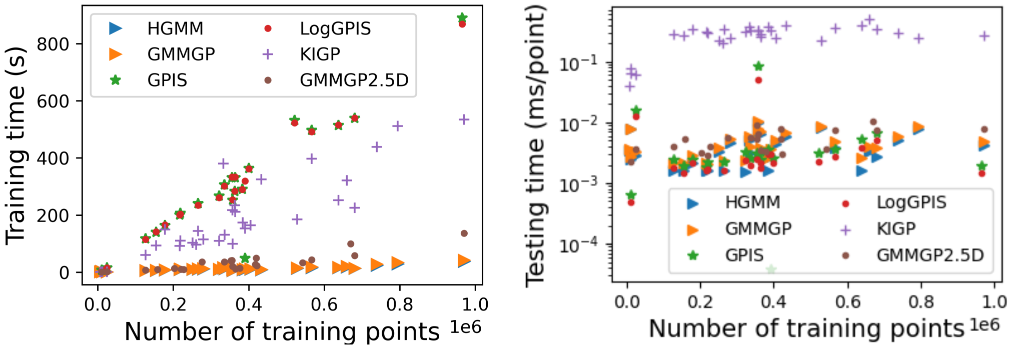
6.6.6 Computation Time Comparison

Regarding computation time, the proposed GMMGP and HGMM generally reach a faster performance, as shown in Figure 6.14. The introduction of the HGMM prior reduces the GP model size in GMMGP, facilitating rapid GP training nearly coinciding with HGMM. Similarly, GMMGP2.5D scales well with large datasets. However, the hierarchical structure and avoidance of segmentation in the proposed GMMGP enhance its efficiency compared to the earlier 2.5D version. During training, the time complexity of GPIS, Log-GPIS and KIGP increases significantly with the number of data points, while the proposed methods remain at a low level. This increase for GPIS and Log-GPIS is primarily attributed to their inherent normal estimation. During the prediction, KIGP incurs high computational costs, whereas performance differences among other methods are less pronounced. Nonetheless, both GPIS and LogGPIS occasionally exhibit significantly higher computational costs, while the proposed methodology maintains a consistently lower and more stable level of time consumption. The HGMM method is often faster, suggesting its viability as a standalone approach for more time-efficient modelling. It is also observed that the total time complexity of GMMGP is highly affected by the HGMM regression; a more optimized spatial structure (like efficient spatial partitioning or indexing) for GMR computation will further reduce the computational time.

6.6.7 Discussion

Based on the experimental results, the HGMM method demonstrates high efficiency due to its "top-down" hierarchical strategy and its robustness to different sub-sampling resolutions (see Section 6.6.3). This indicates it can potentially act as a standalone solution for time-efficient modeling, e.g., real-time localization.

However, its accuracy lags behind that of the integrated HGMM-GP framework, as the hierarchical approximation may overlook finer structural details. Although the hierarchical strategy in HGMM helps to adaptively find the number of Gaussian components and quickly optimize them, the performance remains sensitive to the threshold of the principal curvatures and the size of the modelling objects. For instance, setting a large threshold for



(a) Training time (b) Testing time

FIGURE 6.14. Computational time: note that blue triangles show the HGMM prior estimation while the orange triangles denote the total computation time as the sum of HGMM and GP.

the principal curvature may overlook details in large objects, whereas a small threshold can lead to excessive partitioning of the scene into overly fine nodes. Also, a single fixed threshold may affect the model's ability to fit the scene with both large and small objects. In deeper hierarchies, the effective threshold can become disproportionately small, further exacerbating over-segmentation.

Therefore, HGMM is an approximation of the ideal GMM with the best fidelity, which should be further improved. To enhance model accuracy and reliability, the GP refinement is needed. Instead of exhaustively searching for an optimal number of Gaussian components, the HGMM output is used to initialize priors for subsequent GP training. This hybrid strategy allows the expressiveness of GPs to complement the efficiency of HGMM.

6.7 Summary

In this chapter, we extend the work presented in Chapter 5 to a more general 3D mapping framework for urban environments, utilizing continuous SDF as the underlying representation. A 4D HGMM is integrated with GP inference, incorporating derivative observations within a Bayesian framework. This combination enables the construction of an accurate and computationally efficient implicit surface model with well-calibrated uncertainty estimates. Additionally, we explore the integration of HGMM with alternative GP variants (i.e., LogGP) and introduce a probabilistic update strategy based on the proposed mapping approach.

The framework is evaluated using both real-world LUCOOP LiDAR point clouds and synthetic data from the Carla simulator. Qualitative surface reconstruction results, along with the corresponding uncertainty estimates, are presented and compared across HGMM-only, the proposed 3D GMMGP method, and the previously introduced GMMGP2.5D (from Chapter 5). Furthermore, the proposed methods are benchmarked against state-of-the-art approaches in terms of accuracy, uncertainty quantification, and computational efficiency. The results demonstrate that our approach outperforms existing methods, achieving superior accuracy and uncertainty estimation with efficient runtime performance.

In the following chapters, we further explore the application of this 3D uncertainty-aware map representation in localization, as well as in 3D generation and completion tasks.

(*The results presented in this chapter are adapted from (Zou and Sester 2024), published in IEEE Robotics and Automation Letters (RA-L).)

Chapter 7

Uncertainty-aware Localization Using Uncertain Maps

7.1 Probabilistic Localization with GMMs

In the realm of LiDAR-based localization, a prevalent technique involves the alignment or registration of two point clouds: a reference map and the noisy LiDAR scans captured from a potentially erroneous position. Probabilistic localization utilizes statistical models to estimate the alignment based on the likelihood of data given certain positions or transformations. It is the task of estimating the positions within an environment while considering the associated uncertainty. This approach is based on probability theory and is essential for real-world scenarios, which are dominated by imperfect sensors and measurement noise. The key concepts are often connected to Bayesian Filters and the Markov assumption. The existing probabilistic localization techniques include Monte Carlo Localization (MCL), Kalman filter, Extended Kalman filter, etc.

The process of registering new sensor data against a map can also be probabilistic. NDT, for example, represents the map as a grid of cells, modeling the point distribution within each cell as a single Gaussian. Scan matching is then achieved by finding the transformation that maximizes the log-likelihood of a new scan's points fitting into this probabilistic map, which is a form of registration with an uncertain map. While NDT provides a probabilistic framework, its single-Gaussian-per-cell model can be limiting. In this section, we introduce an approach that builds on this concept, using GMM to model the uncertain map instead of a single Gaussian distribution for point cloud registration.

In traditional NDT approaches, the physical space is partitioned into fixed-size cells, each represented by an individual Gaussian distribution. This method, however, presents several challenges, including the selection of an appropriate cell resolution and the inherent issue of discontinuous distributions between adjacent cells, which might fail to accurately represent the continuity of real-world spaces.

To address these limitations, we proposed the utilization of 3D HGMMs as a more sophisticated alternative to single Gaussian representations. GMMs offer improved performance in environmental modelling by facilitating a more flexible and detailed representation of spatial distributions, which is particularly beneficial for complex and heterogeneous environments.

In large-scale environments, space discretization is still required for computational efficiency. The use of GMMs allows for much larger cell sizes, thereby mitigating boundary errors typically encountered between adjacent cells in traditional NDT setups. This flexibility of GMM structures significantly diminishes the sensitivity of the localization process to the specific dimensions of the cells used, thereby making the cell size a less critical factor in the overall performance of the algorithm.

7.1.1 GMM Transform

To effectively localize an agent within an environment, we first construct a probabilistic map using GMM based on the point clouds collected from the surroundings. This process involves training a 3D HGMM according to the methodologies outlined in Section 6.2. Note that as SDF is not regressed and the additional dimension for SDF is not needed, 3D GMM is trained instead of 4D. Once this probabilistic map is established, incoming point clouds from a vehicle are matched against this reference map to determine the vehicle's position by maximizing the likelihood of the data given the map, akin to point cloud registration processes.

In this case, the registration process involves optimizing the transformation of vehicle poses to ensure that the observed point clouds align with the probabilistic distribution modeled by the GMM, which essentially is a point-model association task.

NDT provides a strategy to optimize the transformation to align points to a single Gaussian distribution. In contrast, the introduction of a mixture model complicates the process due to the presence of multiple Gaussian components in the overall distribution. If the model is accurately trained, it is typical that one primary Gaussian component predominantly represents a local area, while the influence of other components remains marginal. This is the assumption of HGMM, where the environmental surface is primarily modeled by planar Gaussian components. Thus, it's feasible to approximate the model by focusing predominantly on this primary Gaussian component for likelihood maximization, employing a strategy similar to NDT optimization.

To identify the primary Gaussian component that a point $\boldsymbol{x}$ most likely belongs to, we evaluate the weighted probability of each component for the given point. The primary component is identified as follows:

$$k_* = \arg\max_k \ \pi_k p(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k), \tag{7.1}$$

Thus, the likelihood function within the GMM framework can be approximated by focusing on the primary component k_*:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \approx c_1 p(\boldsymbol{x}|z = k_*) + c_2 \tag{7.2}$$

where c_1 and c_2 are constants calibrated such that the total probability mass of $p(x)$ is normalized to one within a specified neighboring space.

Given a set of point clouds $\mathbf{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^N$, utilizing the techniques introduced in Section 2.3.3, the final approximation of the negative log-likelihood of all points with the transformation $(\mathbf{R}, \mathbf{t})$ reads:

$$\boldsymbol{x}_i' = \mathbf{R}\boldsymbol{x}_i + \mathbf{t} \tag{7.3}$$

$$L(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N} -d_1 \exp\left(-\frac{d_2}{2} (\boldsymbol{x}_i' - \boldsymbol{\mu}_{k_*})^\top \boldsymbol{\Sigma}_{k_*}^{-1} (\boldsymbol{x}_i' - \boldsymbol{\mu}_{k_*}) \right) \tag{7.4}$$

where $L(\cdot)$ is the score function. The constants d_1 and d_2 are specified in Equation (2.79).

The transformation parameters in the score function can be iteratively optimized with Newton's algorithm, as introduced in Section 2.3.3.

7.2 Localization with Uncertain Implicit Surfaces

Instead of the probabilistic distributions of points, the uncertain map used in localization can also be represented as an implicit surface map with uncertainty measures, e.g., SDFs.

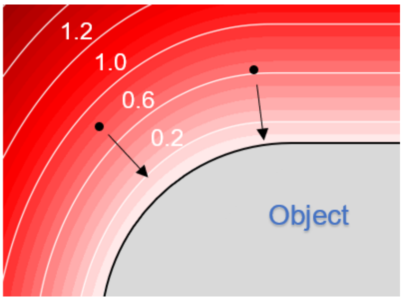


FIGURE 7.1. SDF-based alignment

In this section, we elaborate on how to apply the proposed uncertain implicit fields in the localization and the value of the uncertainty measures.

Compared to localization methodologies that employ probabilistic distributions for map representation, the use of SDFs shifts the focus from probabilistic likelihood maximization to the minimization of geometric distances. While traditional localization approaches, such as the ICP, require the identification of nearest point pairs between datasets and subsequent optimization of their distances to achieve alignment, SDF-based methods provide a distinct computational advantage. With SDF, the localization process can be streamlined significantly. Rather than searching for the nearest point counterparts, the SDF method allows for the direct computation of the shortest distances to the surface from a given query point through a straightforward regression based on the 3D coordinates:

$$d = f(\boldsymbol{x}) \tag{7.5}$$

As illustrated in Figure 7.1, the black dots represent the points that need to be aligned to the surface, while the red gradient indicates the distance field. Given the positions of these points, their distance fields can be estimated using the implicit representation derived from the map. The objective is to iteratively adjust the point positions to minimize the magnitude of their corresponding distance values, thereby aligning them with the zero-level set of the surface. This eliminates the need to identify the closest neighbors, thereby simplifying the localization process. Moreover, this approach circumvents the potential biases that often arise from nearest-neighbor assignments.

7.2.1 Pose Optimization

The objective is to align the erroneous LiDAR points to the surface, which means their SDFs should equal zero. Thus, the objective function is the norm of the distances, which should be minimized. Considering the transformation parameters $\mathbf{R}$ and $\mathbf{t}$, the minimization of the distance function can be expressed as:

$$\mathbf{R}, \mathbf{t} = \arg\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \| f(\mathbf{R}\boldsymbol{x}_i + \mathbf{t}) \|^2 \tag{7.6}$$

If we assume that the rotation angles are small, the rotation matrix $\mathbf{R} \in \mathcal{SO}$ can be approximated in terms of a skew-symmetric matrix associated with an infinitesimal rotation vector $\boldsymbol{\phi}$. It can be expressed in the form:

$$\mathbf{R}(d\boldsymbol{\phi}) \approx \mathbf{I} + [d\boldsymbol{\phi}]_{\times} \tag{7.7}$$

where the small rotation vector is a 3D vector: $\boldsymbol{d\phi} = (d\phi_x, d\phi_y, d\phi_z)$. Therefore, there are 6 transformation parameters to be optimized: $\boldsymbol{\theta} = [t_x, t_y, t_z, d\phi_x, d\phi_y, d\phi_z]^\top$.

The objective function can be rewritten as:

$$d\boldsymbol{\phi}, \mathbf{t} = \arg\min_{d\boldsymbol{\phi}, \mathbf{t}} \sum_{i=1}^{N} \| f(x_i + \mathbf{t} + [d\boldsymbol{\phi}]_{\times} x_i) \|^2 \tag{7.8}$$

For simplicity, we define the updated term in Equation (7.8) as $\delta_i = \mathbf{t} + [d\boldsymbol{\phi}]_{\times} x_i$. As we assume the transformation of the point is small, using the Taylor transformation, the distance function of the transformed point reads:

$$\begin{aligned} f(x_i + \delta_i) &\approx f(x_i) + \nabla f(\boldsymbol{x_i})^{\top} \delta_i && (7.9) \\ &= f(x_i) + \nabla f(\boldsymbol{x_i})^{\top} \mathbf{t} + \nabla f(\boldsymbol{x_i})^{\top} [d\boldsymbol{\phi}]_{\times} x_i && (7.10) \\ &= f(x_i) + \begin{bmatrix} \nabla f(\boldsymbol{x_i})^{\top} & (x_i \times \nabla f(\boldsymbol{x_i}))^{\top} \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ d\boldsymbol{\phi} \end{bmatrix} && (7.11) \\ &= f(x_i) - \mathbf{a}_i \boldsymbol{\theta} && (7.12) \end{aligned}$$

where we define the vector $\mathbf{a}_i = -\begin{bmatrix} \nabla f(\boldsymbol{x_i})^{\top} & (x_i \times \nabla f(\boldsymbol{x_i}))^{\top} \end{bmatrix}$ and $\nabla f(\boldsymbol{x_i})$ represents the gradient of the SDF function.

Equation (7.8) is further reformulated as a quadratic problem of the Least-Squares optimization:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} (\mathbf{f}(\mathbf{X}) - \mathbf{A}\boldsymbol{\theta})^{\top} (\mathbf{f}(\mathbf{X}) - \mathbf{A}\boldsymbol{\theta}) \tag{7.13}$$

where $\mathbf{f}(\mathbf{X})$ represents the vector $[f(\boldsymbol{x_1}), f(\boldsymbol{x_2}), ..., f(\boldsymbol{x_N})]^{\top}$, and $\mathbf{A}$ is the coefficient matrix: $[\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_N]^{\top}$.

Since we applied the proposed GMMGP as the surface map, the distance function and its gradient can be obtained according to Equation (6.16); i.e., the coefficient matrix $\mathbf{A}$ and $\mathbf{f}(\mathbf{X})$ can be derived from that. Assuming that $\mathbf{A}^T\mathbf{A}$ is non-singular, a solution to the above linear Least Squares problem is given by:

$$\boldsymbol{\theta} = (\mathbf{A}^{\top}\mathbf{A})^{-1}\mathbf{A}^{\top}\mathbf{f}(\mathbf{X}) \tag{7.14}$$

As we approximate the non-linear optimization problem in Equation (7.6) with a linear optimization setting, the target states should be updated iteratively until a convergence condition is satisfied.

In the iterative update, the poses are first initialized by a coarse vector, which can be obtained from a GNSS positioning solution or by the global localization method, e.g., MCL. At each iteration step, the transformation parameters $\boldsymbol{\theta}$ are updated by a small incremental term $\delta\boldsymbol{\theta}$, which is estimated in Equation (7.14) at every step. To keep the displacement update δ_i stay small during iterations, we update the points $\boldsymbol{x_{i,k}}$ based on the $\boldsymbol{\theta}_k$ at each k-th iteration:

$$x_{i,k} = \mathbf{R}_k x_{i,0} + \mathbf{t}_k \tag{7.15}$$

where $\mathbf{R}_k$ and $\mathbf{t}_k$ are the rotation matrix and translation vector computed from the k-th iteration. They can be interconverted to $\boldsymbol{\theta}_k$, each representing the equivalent transformation.

The transformation is updated by:

$$\mathbf{R}_k = \mathbf{U}_k \mathbf{R}_{k-1} \tag{7.16}$$

$$\mathbf{t}_k = \mathbf{U}_k \mathbf{t}_{k-1} + \delta \mathbf{t}_k \tag{7.17}$$

where $\mathbf{U}_k$ and $\delta\mathbf{t}_k$ represent the equivalent rotation matrix and translation of the transformation $\delta\boldsymbol{\theta}_k$. It can be denoted as:

$$\mathbf{U}_k, \mathbf{t}_k = T(\delta\boldsymbol{\theta}_k) \tag{7.18}$$

The iteration stops when the update of the transformation parameters $|\delta\boldsymbol{\theta}_k|$ is smaller than a pre-defined convergence threshold.

7.2.2 Uncertainty-aware Optimization

In the described pose estimation framework, all scan points from the SDF are treated with uniform significance, as there is no prior information. Given that each point's SDF is inferred with an associated uncertainty in the GMMGP framework, it is reasonable to integrate this uncertainty into the pose estimation process, assigning varying levels of importance to each SDF based on its reliability.

Therefore, we leverage the variance of the inferred SDFs during the optimization process by transitioning from a standard least squares formulation to a weighted least squares approach. Denote the set of all considered SDFs as $\mathbf{Y} = \mathbf{f}(\mathbf{X})$, with their associated uncertainty matrix $\boldsymbol{\Sigma}_Y$. This adjustment is mathematically represented by modifying the objective function to incorporate the variances as weights, as shown below:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} (\mathbf{f}(\mathbf{X}) - \mathbf{A}\boldsymbol{\theta})^\top \boldsymbol{\Sigma}_Y^{-1} (\mathbf{f}(\mathbf{X}) - \mathbf{A}\boldsymbol{\theta}) \tag{7.19}$$

Accordingly, the estimation of the transformation update is then computed using the generalized least squares solution:

$$\delta\boldsymbol{\theta} = (\mathbf{A}^\top \boldsymbol{\Sigma}_Y^{-1} \mathbf{A})^{-1} \mathbf{A}^\top \boldsymbol{\Sigma}_Y^{-1} \mathbf{f}(\mathbf{X}) \tag{7.20}$$

Here, since we are only interested in the variance of the SDF inference, each SDF output is assumed as independent, yielding a diagonal covariance matrix $\boldsymbol{\Sigma}_Y$.

This approach not only refines the accuracy of pose estimation by factoring in the varying uncertainties of the SDF measurements but also enhances the robustness of the estimation against less reliable data. Thus, SDFs with lower uncertainty exert a larger influence on the determination of $\boldsymbol{\theta}$, optimizing the pose estimation process to yield more reliable outcomes.

7.2.3 Uncertainty Propagated from Maps

In the final pose estimation, various error sources contribute to uncertainty, one of which is the uncertainty inherent in the mapping data. The mapping from SDFs to pose estimation can be expressed in Equation (7.20) for the last iteration. The Jacobian matrix of this transformation, defined as $\mathbf{J} = (\mathbf{A}^\top \boldsymbol{\Sigma}_y^{-1} \mathbf{A})^{-1} \mathbf{A}^\top \boldsymbol{\Sigma}_Y^{-1}$, facilitates the propagation of uncertainty from the SDFs to the pose estimates. Thus, the propagated uncertainty in the pose estimation $\boldsymbol{\theta}$ is quantified by:

$$\begin{aligned} \boldsymbol{\Sigma}_\theta &= \mathbf{J}\boldsymbol{\Sigma}_Y\mathbf{J}^\top \qquad (7.21)\\ &= (\mathbf{A}^\top \boldsymbol{\Sigma}_Y^{-1} \mathbf{A})^{-1} \qquad (7.22) \end{aligned}$$

Note that this is only calculated for the last iteration.

SDF Independence

In linear least squares estimation, when the measurements—here, the estimated SDFs—are considered informative (i.e., neither identical nor highly correlated), each additional measurement incrementally contributes to a more robust understanding of the system being modeled. This progressive accumulation of information systematically refines the parameter estimates. Consequently, the uncertainty associated with each parameter estimate typically diminishes as the number of samples grows, resulting in increasingly precise estimations. This reduction in uncertainty is most pronounced when the error terms in the model are independent and identically distributed (i.i.d.), which maximizes their informativeness.

This can also be explained by the key term in the formula for the covariance of the estimation: $(\mathbf{A}^\top \mathbf{\Sigma}_Y^{-1} \mathbf{A})^{-1}$. As more measurements are added, $\mathbf{A}^\top \mathbf{\Sigma}_Y^{-1} \mathbf{A}$ typically becomes larger in magnitude because each row of $\mathbf{A}$ weighted by the inverse of its respective variance contributes positively to the overall matrix. The determinant of $\mathbf{A}^\top \mathbf{\Sigma}_Y^{-1} \mathbf{A}$ increases, making the inverse matrix (which is the covariance matrix of the parameter estimates) smaller in magnitude.

In practice, we assume the map uncertainty Σ_Y to be a diagonal matrix, positing that errors are independent. This simplification leads to an underestimation of the uncertainty in parameter estimates due to the inherent dependencies within the data. In reality, a significant correlation exists between the SDFs, particularly among neighboring points. This correlation implies the presence of redundant or non-informative samples, which distorts the accuracy of our uncertainty estimates. This uncertainty underestimation due to the simplified independence assumption is illustrated in the experiment section.

In fact, in our uncertain SDF-based method, it is possible to approximate a full covariance matrix of the SDF. Typical techniques primarily provide variance estimates for outputs but fail to capture the correlations between two different outputs, $f(x_i)$ and $f(x_j)$. Our method extends these capabilities by treating GMR results as the prior of GP, enabling the computation of a full covariance matrix, inspired by the kernel-based approach. For instance, if the Matérn kernel is employed, the covariance matrix can be expressed as:

$$\Sigma_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_i \sigma_j \left(1 + \frac{\sqrt{3}\,|\boldsymbol{x}_i - \boldsymbol{x}_j|}{\ell}\right) \exp\left(-\frac{\sqrt{3}\,|\boldsymbol{x}_i - \boldsymbol{x}_j|}{\ell}\right). \tag{7.23}$$

Other kernels, such as squared exponential kernels, can also be employed.

However, if the SDF correlation is incorrect, the solver can degrade in performance — sometimes more than in the simpler diagonal case. Over-parameterizing the covariance with spurious or mis-estimated correlations can lead to numerical instability and poor conditioning. Additionally, using a full covariance will introduce additional complexity and computation. A full $N \times N$ covariance for N measurements can be expensive. For large point sets, this becomes $O(N^2)$ in storage and manipulation. The computation of the inverse of the large covariance matrix is $O(N^3)$.

Prior Fusion

If we already have an initial pose with the covariance Σ_0, it can be treated as a prior in a Bayesian sense. The result is akin to prior fusion:

$$p(\boldsymbol{\theta}|\mathbf{X}) \propto p(\mathbf{X}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{7.24}$$

$$\log p(\boldsymbol{\theta}|\mathbf{X}) = \log p(\mathbf{X}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) + \text{const} \tag{7.25}$$

Therefore, the objective function changes, and the prior gives an additional cost:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} [(\mathbf{f}(\mathbf{X}) - \mathbf{A}\boldsymbol{\theta})^\top \boldsymbol{\Sigma}_Y^{-1} (\mathbf{f}(\mathbf{X}) - \mathbf{A}\boldsymbol{\theta}) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^\top \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\theta}_0)] \tag{7.26}$$

The corresponding estimation of the transformation update is written as:

$$\delta\boldsymbol{\theta} = (\mathbf{A}^\top \boldsymbol{\Sigma}_Y^{-1} \mathbf{A} + \boldsymbol{\Sigma}_0^{-1})^{-1} (\mathbf{A}^\top \boldsymbol{\Sigma}^{-1} \mathbf{f}(\mathbf{X}) - \boldsymbol{\Sigma}_0^{-1} (\boldsymbol{\theta}_k - \boldsymbol{\theta}_0)) \tag{7.27}$$

with the resulting posterior covariance $\boldsymbol{\Sigma}_{\theta,posterior}^{-1}$calculated by:

$$\boldsymbol{\Sigma}_{\theta,posterior} = (\mathbf{A}^\top \boldsymbol{\Sigma}_Y^{-1} \mathbf{A} + \boldsymbol{\Sigma}_0^{-1})^{-1} \tag{7.28}$$

Hence, the resulting posterior covariance $\boldsymbol{\Sigma}_{\theta,posterior}$ typically does not exceed the covariance obtained if there is no prior.

Empirical Residual

The pose uncertainty can be underestimated or overestimated if the initial SDF covariance includes an overall mismatch in the global noise level. The empirical residual check can be helpful in this case. Often in classical least squares, after solving $\boldsymbol{\theta}$, the final empirical residuals are checked to estimate how large the measurement noise actually is. It can be incorporated to refine the assumption about the measurement noise level, i.e., the estimated SDF noise in this task, as the SDF uncertainty might have been too optimistic or too pessimistic. The calibration factor of the empirical residual is calculated by:

$$s = \frac{(\mathbf{f}(\mathbf{R}_\theta \mathbf{X} + \mathbf{t}_\theta))^\top \boldsymbol{\Sigma}_Y^{-1} (\mathbf{f}(\mathbf{R}_\theta \mathbf{X} + \mathbf{t}_\theta))}{(N-6)} \tag{7.29}$$

where s denotes the final calibration scale based on empirical residuals, and it is unitless.

If the initially estimated $\sigma_{sdf}(\boldsymbol{x})$ is significantly larger or smaller than the final residual, a scaling can be applied to the final covariance matrix:

$$\boldsymbol{\Sigma}_{\theta,scale} = s \cdot \boldsymbol{\Sigma}_\theta. \tag{7.30}$$

However, the residual is only an approximation of the potential uncertainty. The quality of the pose covariance estimation can still be affected by the local minima or non-Gaussian outliers. If there are correlated errors in the SDF or outliers, empirical residuals might not reflect the real noise variance. In that case, scaling the entire covariance could be misleading.

Non-linear Issues

Furthermore, the uncertainty propagation through a linearized model merely accounts for the direct transformation of input uncertainties (Σ_Y) via the model's linear approximation. This approach fails to capture additional uncertainties that arise from the model's nonlinear behavior and the possibility of encountering multiple local minima. These factors are crucial in understanding the comprehensive uncertainty landscape and require more sophisticated modeling to address adequately.

7.3 Experiments

7.3.1 Experimental Setting

In the experimental evaluation of our proposed localization methods, we utilize the real-world LUCOOP dataset (introduced in Chapter 4), which provides dense reference map point clouds and sparse LiDAR scans acquired from a Velodyne sensor. The experiment is conducted in the challenging urban canyon environment. Initially, the dense point cloud is employed to create a map incorporating uncertainties. Subsequently, points captured from the Velodyne scanner are aligned with this map to refine the pose estimation. We assessed the accuracy of our localization method by comparing the estimated poses against these ground truths and computed the Mean Absolute Error (MAE) and RMSE. These metrics were also used to evaluate the performance relative to established NDT localization techniques.

7.3.2 Accuracy Evaluation

In Table 7.1, we present a comparative analysis of localization accuracy among our proposed GMM Transform (GMMT), GMMGP-based localization, and the traditional NDT. We employ both RMSE and MAE as metrics. The metric RMSE is more sensitive to outliers compared to MAE. The results demonstrate that the GMMGP-based approach yields the lowest RMSE and MAE values, suggesting fewer outliers and higher overall accuracy, thereby indicating superior precision and robustness. Conversely, the larger RMSE values associated with NDT suggest a higher prevalence of outliers, potentially compromising its reliability under similar conditions.

TABLE 7.1. Localization Accuracy

RMSE (m)	X	Y	Z	3D
NDT	0.032	0.158	0.043	0.166
GMMT	0.061	0.062	0.094	0.128
GMMGP	0.008	0.042	0.033	**0.055**
MAE (m)				
NDT	0.014	0.070	0.019	0.078
GMMT	0.026	0.052	0.058	0.082
GMMGP	0.006	0.037	0.029	**0.051**

7.3.3 Uncertainty Investigation

The evaluation of pose uncertainty associated with the proposed SDF-based, uncertainty-aware localization method was conducted as described in Section 7.2.3. To enhance our understanding of the model's performance, the ground truth trajectory is compared with the estimated trajectory, which includes associated covariance ellipses representing pose uncertainty. Figure 7.2 displays the 3D errors in the trajectory, mostly at the centimeter level. For a more detailed analysis of the pose uncertainties and corresponding errors, attention is directed to specific points along the trajectory, as shown in Figure 7.3. The blue trajectory represents the estimated poses, while the green trajectory indicates the ground truth. The red ellipses depict the uncertainty as error ellipses at a 99% confidence level. If the uncertainty estimation is accurate and well-calibrated, the ground truth trajectory points are expected to fall within these ellipses. In this figure, the first row highlights

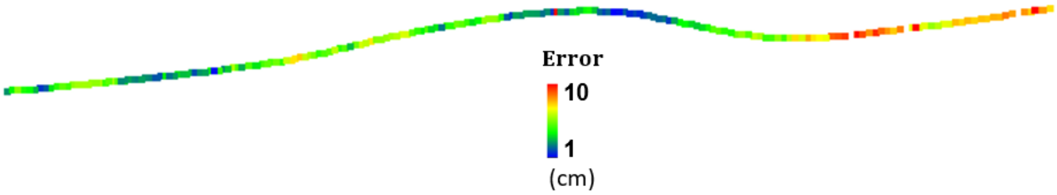


FIGURE 7.2. Errors in pose estimation

scenarios where the true poses are well-contained within the error ellipses (denoted by red ellipses), indicating accurate uncertainty estimation. Conversely, the second row illustrates instances where the pose uncertainty is underestimated, as evidenced by the true poses extending beyond the boundaries of the error ellipses.

In the second row, there are some results slightly exceeding the error ellipses. As depicted in Figures 7.3d and 7.3e, larger discrepancies predominantly occur along the trajectory direction. This pattern is typical in urban canyon environments, where the measurements of building façades on either side of the street are mostly aligned in parallel structures.

The third row of Figure 7.3 illustrates cases with large localization errors that significantly exceed the extent of the corresponding uncertainty ellipses. For visualization purposes, the scale of the coordinates differs from that in the first two rows, where the grid cell width is of $2\,cm$ instead of $1\,cm$. Again, these large deviations primarily occur along the trajectory direction. Although the uncertainty ellipses do not fully encompass the errors, their elongated shape accurately reflects the dominant uncertainty along the y-axis, aligning with the direction of the primary error.

To assess the accuracy of the pose uncertainty estimates quantitatively, the Chi-square statistic was calculated, comparing the pose errors to the predicted distribution of pose errors. Histograms of these statistics are presented in Figure 7.4. For 3D data, the critical values for the Chi-square statistic at 95% and 99% confidence levels are 7.9 and 11.3, respectively. As shown in Figure 7.4, the x-axis denotes the magnitude of the Chi-square statistics. The results indicate that a substantial proportion of the computed statistics still exceed these thresholds by reaching magnitudes in the hundreds. This suggests that the pose uncertainty estimated by the model is significantly underestimated. Specifically, because the model's estimated uncertainty is quite small—on the millimeter scale—relative to some centimeter-level errors as shown in the figure, the computed statistics frequently surpass the critical values.

The systematic underestimation of pose errors suggests a limitation in using a diagonal SDF covariance matrix to account for the final pose estimation. It may be necessary to refine and adjust the covariances to better capture these directional uncertainties and improve overall pose accuracy. To investigate the bias introduced by the simplified mean-field diagonal covariance, an experiment is conducted in which the full covariance matrix of the SDF is employed, as computed using Equation (7.23). Subsequently, the Chi-square statistic is recomputed based on the localization results obtained with the full covariance matrix. As shown in Figure 7.5, the results yield smaller Chi-test statistic, suggesting that using a full covariance can improve the realism of the pose covariance estimate. Although the uncertainty is slightly larger than the method only using the diagonal matrix, it is still underestimated relative to the pose errors, with the test statistics much larger than the limit of the 99% confidence - 11.3. This suggests that the underlying correlations may not be fully captured or adequately represented in the current model. Since the covariance matrix of the SDF is derived using kernel-based methods in GPs, the modeled correlation

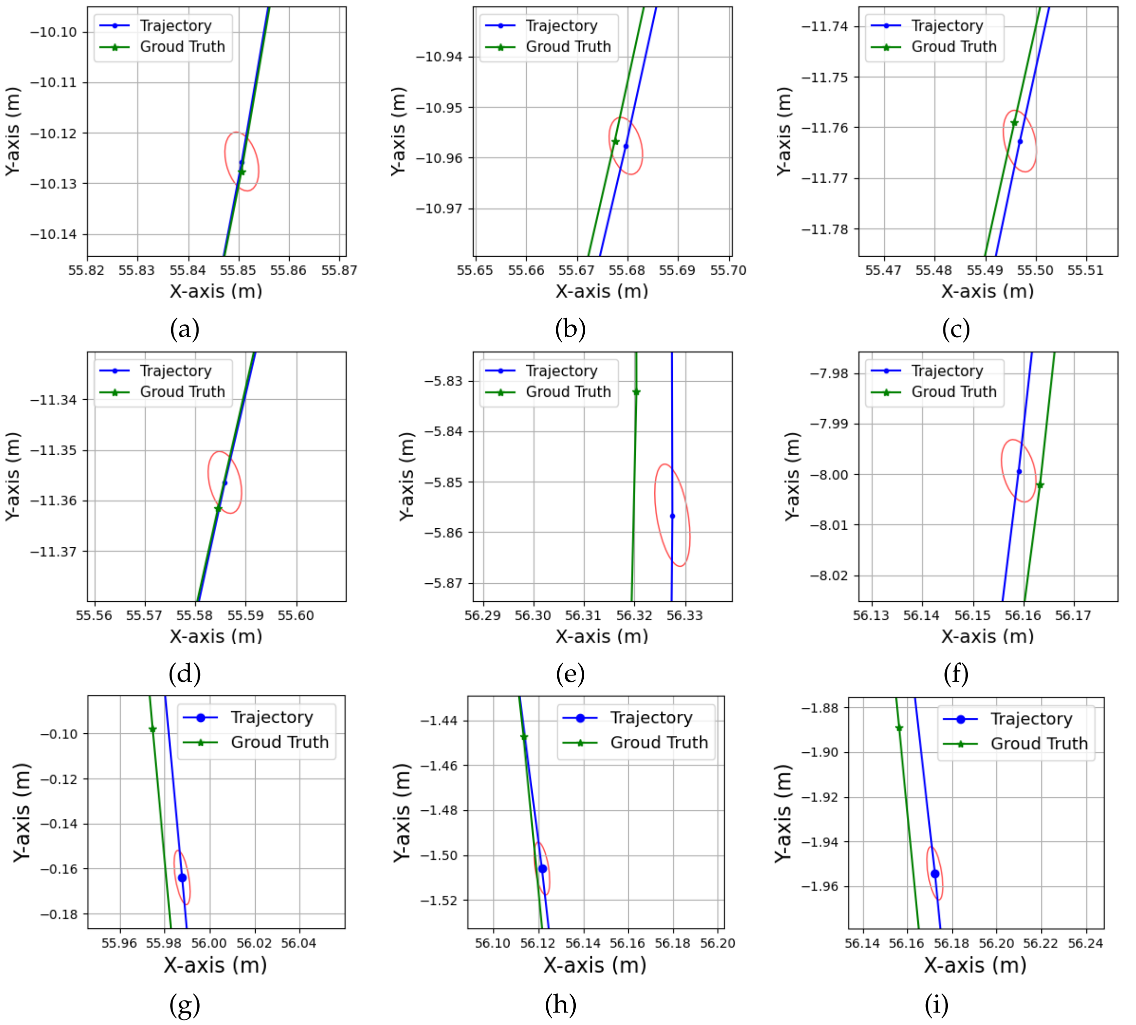


FIGURE 7.3. Trajectory and Covariance Ellipses

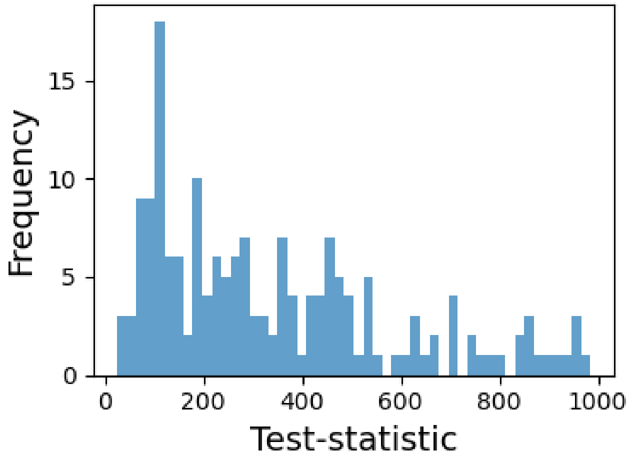


FIGURE 7.4. χ^2-test statistic of 3D pose errors

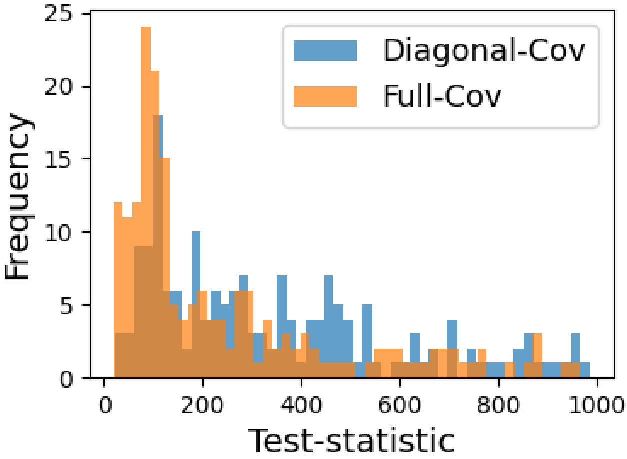


FIGURE 7.5. χ^2-test statistics with full covariance matrix

depends primarily on the spatial distance between point pairs. In Chapters 5 and 6, a small length-scale parameter was chosen to prevent overly smooth predictions, resulting in a rapid decay of correlation with distance, e.g., falling to near zero for points more than 0.5 meters apart (see Section 2.2.5). However, in the context of localization, this distance-based correlation may not adequately capture the true geometric dependencies. For instance, two distant points lying on the same planar surface may still exert similar, strongly correlated influences on the pose estimation due to the structural alignment, which is not reflected by the kernel function. Given a large number of those points, the estimated pose uncertainty is largely affected, and thus, it remains underestimated.

It also has to be mentioned that, with the improved accuracy and uncertainty quantification, there is a price of efficiency. The iterative computation with the full covariance matrix is quite expensive, 107.66 times slower than the one with the diagonal matrix.

In the experiment, empirical residuals were calculated to assess the appropriateness of the scale at which the SDF uncertainty was modeled. Figure 7.6 displays the scaling factor derived from SDF residuals of the transformed scan points, with all scaling factors of the empirical residuals being smaller than 1.0. This observation suggests that the SDF variances were likely overestimated. However, this is contradictory to the underestimation of the actual pose uncertainty, denoted as Σ_θ. Therefore, the underestimation of the pose uncertainty cannot be attributed to an underestimation of the SDF uncertainty Σ.

7.3.4 Discussion

While the proposed map framework has led to improvements in localization accuracy, the associated uncertainty estimates remain insufficient, resulting in an underestimation of the true pose uncertainty. To address this limitation in future work, several directions can be explored.

- First, the covariance structure can be enhanced by incorporating geometric relationships between points—for instance, accounting for planar or structural consistency when constructing the covariance matrix. As mentioned above, the points on the same plane and with similar effects on the pose estimation should have high correlation. This would allow better modeling of spatial correlations and reduce the problem of underestimation of the pose uncertainty.

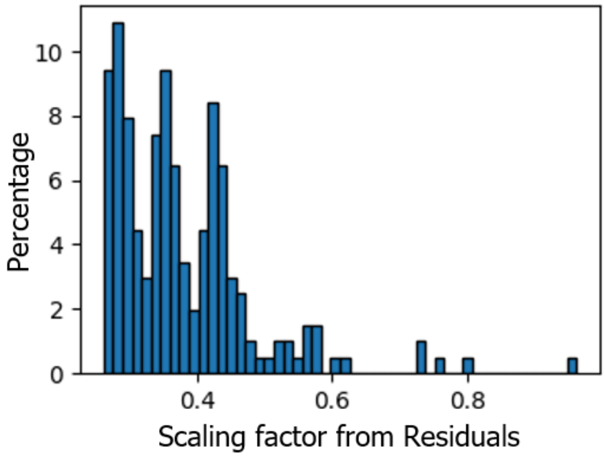


FIGURE 7.6. SDF residual calibration

- Second, although the use of a full covariance matrix for the SDF did not completely capture the errors in the current experiments, it nonetheless indicated the value of modeling inter-point correlations. Compared to simplified mean-field assumptions, the use of the current covariance matrix slightly improves the uncertainty estimation of pose estimation. This suggests that further refinement of the correlation structure could enhance the robustness of uncertainty estimates in localization.

- Lastly, ambiguities in point-to-surface associations—especially in challenging scenarios such as urban canyons with repetitive and parallel structures—should be further investigated. These ambiguities often lead to poorly conditioned optimization.

Chapter 8

Probabilistic Diffusion Models for 3D Gaussian Generation

8.1 3D Geometry Generation

Generative models for 3D modeling have emerged as a rapidly advancing research area with broad applicability across various domains, including autonomous driving (scene reconstruction for navigation and perception), robotics (training in simulated environments), and AR/VR gaming (enhancing immersive experiences). A critical aspect of applying generative models in these tasks is selecting the appropriate map representation for generation and reconstruction, tailored to the specific requirements of each application.

For instance, in some safety-critical applications such as autonomous systems, an uncertainty-aware map representation is essential to ensure reliability and robustness in decision-making under uncertain conditions. In large-scale outdoor environments, where computing resources and memory are often limited, a scalable mapping framework becomes crucial to efficiently manage and process extensive datasets. Additionally, in scenarios where GPU acceleration is unavailable, such as embedded systems or low-power robotic platforms, developing CPU-compatible mapping representations is equally valuable.

In the previous chapters, we introduced an uncertainty-aware 3D mapping framework based on probabilistic inference and demonstrated its effectiveness in outdoor localization tasks. This framework is specifically designed to model complex urban environments with high accuracy, robustness, and computational efficiency, while explicitly representing spatial uncertainties. However, given the inevitable incompleteness of real-world measurements, a key question arises: can shape completion using deep generative models help to fill in missing regions and produce realistic surface reconstructions—ideally, with quantifiable uncertainty?

In this chapter, we extend our uncertainty-aware map representation by integrating it with modern 3D deep generative models. This integration aims to leverage the probabilistic structure of the proposed map to improve both the fidelity of shape completion and the reliability of uncertainty estimation in generative 3D reconstruction. We begin by developing and investigating methods for indoor, instance-level objects, and subsequently discuss the potential for transferring these methods to outdoor scenes with more complex and large-scale geometries.

Recently, image rendering-based map representations leveraging implicit radiance fields have gained significant attention as an alternative to direct geometric modeling. Notable approaches such as NeRF (Mildenhall et al. 2021) and 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) prioritize radiance field modeling and rendering accuracy over explicit geometry reconstruction. These methods provide a novel representation that enables photorealistic rendering and can relax the need for direct, precise geometric reconstruction in certain downstream applications.

However, geometry reconstruction remains of high interest, particularly for applications requiring accurate 3D spatial understanding. To address this issue, recent studies (P. Wang et al. 2021; Z. Yu et al. 2024) have integrated SDFs and geometry modeling into NeRF and 3DGS frameworks to improve surface reconstruction capabilities. While these advancements enhance geometric fidelity, they still suffer from key limitations: (i) a lack of uncertainty quantification, making it difficult to assess confidence in reconstructed structures, and (ii) scalability challenges, as these methods remain computationally expensive and memory-intensive, particularly when applied to large-scale environments.

Addressing these limitations requires further research into developing more efficient, uncertainty-aware geometric modeling approaches that can integrate with implicit representations while maintaining scalability for real-world applications. In contrast to NeRF, which requires a large number of neural parameters, and 3DGS, which models millions of Gaussians even for small scenes or objects, the proposed mapping framework in this thesis offers a more scalable and memory-efficient solution while simultaneously providing uncertainty quantification for geometric estimation.

To further explore generative 3D modeling, we employ a deep generative diffusion model to generate 3D probabilistic geometry based on Gaussian representations (see Chapter 6). Similar to other implicit representations, the generated outputs require additional post-processing steps to extract explicit meshes for downstream applications. In this work, we adopt the Marching Cubes algorithm (Lorensen and Cline 1998) to convert the implicit representation into an explicit surface mesh, facilitating further analysis and visualization.

8.1.1 Diffusion Model for 3D Gaussian

In light of recent breakthroughs in denoising diffusion probabilistic models (DDPM) (Ho et al. 2020), we have developed a methodology for modeling the distribution of our 3D Gaussian maps by employing a diffusion-based generative model. A diffusion model for 3D Gaussian generation can generate a continuous point distribution for the 3D geometry, which provides the possibility to sample any desired resolution from the continuous distribution. In contrast, the original point cloud generation, e.g., Point-E (A. Nichol et al. 2022), only generates a fixed quantity of the points, and requires up-sampling for a more dense point cloud.

Prior to the training process for the 3D diffusion model, it is essential to preprocess the original training dataset. This preparation involves the generation of implicit 3D Gaussian maps from the mesh data or point clouds. This step transforms the spatial geometric data into a format that is compatible with the diffusion model's requirements, ensuring that the model can effectively learn the underlying distributional characteristics of the 3D objects represented in the dataset. Specifically, we utilize a modified version of the previously proposed 3D Gaussian framework (see Chapter 6) to generate implicit 3D Gaussian maps from the training dataset: we standardize the number of Gaussian components to a fixed constant (e.g., $K = 512$). This modification ensures compatibility with the diffusion model, which requires a consistent input and output structure in terms of the number of 3D Gaussians.

Additionally, it is essential to parameterize the Gaussian in a manner that aligns with the deep generative diffusion model. A single 3D Gaussian is characterized by its mean vector in 3D space, its 3×3 covariance matrix, and its associated mixing weight. By normalizing the mean $\boldsymbol{\mu} \in \mathbb{R}^3$ and the mixing weight $w \in \mathbb{R}$ to conform to the standard normal distribution $\mathcal{N}(0, \boldsymbol{I})$, the normalized mean and weight can be directly utilized as features within the diffusion model framework. In contrast, parameterization of the covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$ would be more complicated due to the requirement that

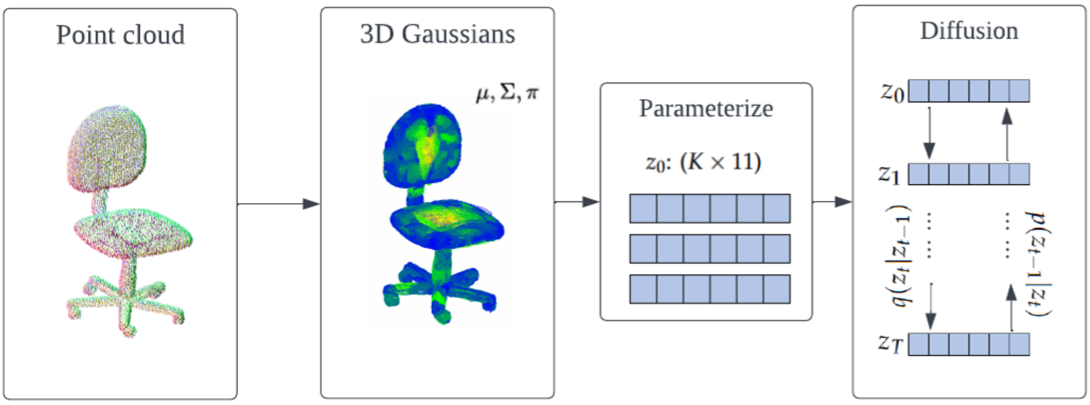


FIGURE 8.1. 3D Gaussian representation

covariance matrices remain symmetric and positive-definite. Naïve parameterization by individually setting its diagonal and off-diagonal elements does not inherently guarantee positive definiteness. This constraint depends on all elements of the matrix in a complex, non-linear way (through the eigenvalues). Ensuring that directly during optimization is difficult. To address this, we adopt a parameterization strategy based on the eigenvalues and eigenvectors of Σ:

$$\Sigma = Q \Lambda Q^{\top} \tag{8.1}$$

where $\Lambda = \text{diag}(\boldsymbol{\lambda})$ is a diagonal matrix, denoting the eigenvalues. It can be represented by the vector $\boldsymbol{s} = \pm\boldsymbol{\lambda}^{0.5}$. $Q \in \mathcal{SO}(3)$ is an orthogonal matrix with a positive determinant, composed of eigenvectors. Q can be seen as a rotation matrix, and represented by a quaternion $\boldsymbol{q} = (w, u_x, u_y, u_z)$. In summary, the covariance can then be parameterized by a 7-dimensional vector $(\boldsymbol{q}, \boldsymbol{s})$.

To generate 3D Gaussians with diffusion, we developed a 3D diffusion framework using a transformer-based network as the denoising backbone to encode the parameters for each Gaussian component. In particular, we represent a training dataset as a tensor of shape $K \times 11$, where K is the number of Gaussian components. Within this framework, the representation of a 3D Gaussian mixture is analogous to a point cloud, enriched with complex feature sets. The data contain 3D (x, y, z) coordinates, seen as the means of the Gaussians, as well as other features $\theta = (\boldsymbol{q}, \boldsymbol{s}, w)$ parameterizing the covariance matrices and the component mixing weights. Once the diffusion model is trained, we can generate these tensors directly with diffusion, starting from random noise of shape $K \times 11$, and progressively denoising it. Additionally, the Gaussian parameters can also be interpreted as latent features for the 3D geometry. From this perspective, these parameters are intricate latent features that manifest complex dependencies and spatial relationships, thus offering a nuanced approach to modeling and generating 3D structures.

To illustrate this process, Figure 8.1 presents a case where the 3D point cloud is initially modeled using a mixture of 3D Gaussian distributions. Subsequently, the parameters of these Gaussian mixtures are encapsulated into a tensor z_0 with dimensions $(K \times 11)$, which will be fed to the diffusion model as the training sample.

Figure 8.2 illustrates the diffusion process implemented in our model, which is the last block in Figure 8.1. The variable z_0 represents an actual sample from the underlying true data distribution $p(z)$, whereas z_T denotes the diffused noisy sample at timestep T. $q(z_t|z_{t-1})$ and $p_\theta(z_{t-1}|z_t)$ mathematically describe the forward and reverse transformations of the 3D Gaussian parameters through the diffusion process, respectively, with the

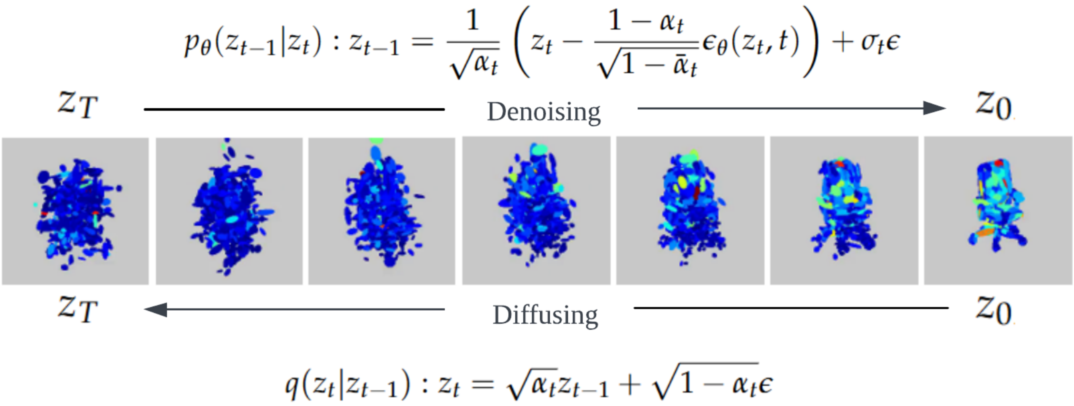

$p_\theta(z_{t-1}|z_t) : z_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(z_t, t)\right) + \sigma_t\epsilon$
z_T
Denoising
z_0
z_T
Diffusing
z_0
$q(z_t|z_{t-1}) : z_t = \sqrt{\alpha_t}z_{t-1} + \sqrt{1-\alpha_t}\epsilon$

FIGURE 8.2. An illustration of the 3D Gaussian Diffusion framework with the forward diffusing process and the reverse denoising process

arrows indicating the direction of these transformations.

During the training phase, noise is incrementally introduced to the 3D Gaussian parameters following the specified diffusion equation. By minimizing the discrepancy between the added noise and the noise predicted by the model using the DDPM loss function, the neural model effectively learns the noise characteristics at each timestep. The denoising phase starts from the noisy sample z_T and progressively reconstructs a new sample z_0 of the 3D Gaussian mixtures, according to the denoising equation depicted in the figure, employing the learned noise models at each step.

8.1.2 Diffusion Model for 4D Gaussian

In Section 8.1.1, we introduced a novel framework utilizing a diffusion model to generate 3D Gaussian mixture distributions, facilitating the continuous representation of point distributions in 3D geometrical space. Building upon this foundational work, this section delineates our progression to employing diffusion models for generating 4D Gaussian mixtures, encompassing three spatial dimensions and one SDF dimension, represented as (x, y, z, d). In this 4D setting, a direct inference of SDF based on the 4D Gaussian mixtures becomes feasible.

Transitioning from 3D to 4D necessitates a significant modification in our parameterization strategy. While the approaches for parameterizing the means and weights remain unchanged, the method for parameterizing the covariance matrices undergoes a critical adjustment. Originally utilizing eigen-decomposition for its beneficial properties in direct geometrical interpretation, we shift to employing Cholesky decomposition in the 4D context. This change addresses the increased complexity and computational demands associated with parameterizing the orthogonal group $Q \in \mathcal{SO}(4)$ in four dimensions. Although the Cholesky decomposition can be less intuitive, it offers a more computationally efficient and stable method for ensuring the positive definiteness required for the covariance matrices in higher-dimensional spaces, as outlined below:

$$\Sigma = LL^\top \tag{8.2}$$

where L is a 4×4 lower triangular matrix with positive diagonal entries. It can be defined by its diagonal elements $\boldsymbol{l_d} = [l_{00}, l_{11}, l_{22}, l_{33}]$ and its off-diagonal elements $\boldsymbol{l_{ij}} = [l_{10}, l_{20}, l_{21}, l_{30}, l_{31}, l_{32}]$.

Therefore, we process the training data of 4D GMMs as a tensor of $K \times 15$, with the inner dimension including 4-dimensional mean values, one-dimensional weight, and 10-dimensional covariance parameters. All features are normalized to the standard Gaussian distribution.

By minimizing the difference between the true added noise and the neural approximation, we trained the neural parameters in the denoising networks. In the generation step, a 4D GMM can be generated from the random noise of shape $K \times 15$. Based on the SDF inference, surface meshes can be obtained through the same method introduced in Chapter 6, using the marching-cube algorithm.

8.1.3 Loss Functions

The collection of parameter matrices across all training samples defines the target data distribution for our diffusion model. As described in Chapter 2, the denoising network, parameterized by θ, is trained to match this distribution by minimizing the KL divergence (or cross-entropy) between the true data distribution and the model's reverse-diffusion distribution, thereby maximizing the evidence lower bound. In practice, these training objectives can be unified and written as:

$$\mathcal{L}_\epsilon = \mathbb{E}_{t \sim \mathcal{U}(0,1), z_0 \sim p(z_0), \epsilon \sim \mathcal{N}(0,I)} \left[\|\epsilon - \epsilon_\theta(z_t, t)\|^2 \right], \tag{8.3}$$

In this formulation, the model is optimized to predict the injected noise: at each diffusion timestep t, the neural network learns to predict the noise ϵ_θ that has been added to the original data.

Alternatively, one can train the network to regress the denoised sample directly rather than the noise. In that case, the loss function takes the form:

$$\mathcal{L}_{z_0} = \mathbb{E}_{t \sim \mathcal{U}(0,1), z_0 \sim p(z_0), \epsilon \sim \mathcal{N}(0,I)} \left[\|z_0 - z_{0,\theta}(z_t, t)\|^2 \right], \tag{8.4}$$

8.1.4 Denoising Backbones

The 2D diffusion model for image processing utilizes a U-Net architecture, a type of convolutional neural network originally designed for biomedical image segmentation. The U-Net is distinguished by its symmetric structure, comprising a contracting path to capture context and a symmetric expanding path that enables precise localization. This architecture effectively facilitates the gradual denoising of images through the diffusion model's iterative refinement process.

To extend this denoising framework to 3D space and accommodate the complexity of unstructured data representations, such as the rich point clouds in this case, it is necessary to adapt the underlying neural network architecture. For this purpose, we can utilize advanced architectures - the transformer-based networks (Vaswani et al. 2017) and the established 3D point cloud learning frameworks, such as Point-Voxel CNN (PVCNN) (Z. Liu et al. 2019).

Transformer-based models for 3D Gaussian denoising in the diffusion introduce an innovative approach by employing self-attention mechanisms that directly capture the global dependencies between Gaussian parameters in the samples. Unlike traditional convolutional methods, transformers compute responses at a position based on weighted contributions from all positions, thus inherently adapting to the irregular and sparse nature of 3D Gaussian parameters. This offers superior flexibility and scalability for 3D Gaussian representation with rich information and complex dependencies between features.

PVCNN (Z. Liu et al. 2019) was originally proposed for 3D point clouds with color features, which leverages the efficiency of voxel-based processing while retaining the fine-grained detail provided by point-based processing, making it well-suited for handling the nuanced spatial relationships inherent in 3D data. The Gaussian mixture parameters in our case can be seen as the 3D point clouds with richer information, i.e., more features. By aggregating point features within each voxel, PVCNN preserves fine-grained details while employing convolutions over the voxel grid to capture broader contextual information.

Incorporating the advanced architectures into the diffusion model framework allows for robust handling of the high-dimensional, information-dense, and unstructured nature of 3D Gaussian representation. Such adaptations ensure that the diffusion process can accurately model and reconstruct complex spatial structures. In the experiments, we provide a comparison of using these two different backbones for diffusion models.

8.2 Reconstruction with Partial Measurements

8.2.1 Posterior Sampling

As discussed earlier, incomplete measurements of an object often arise due to occlusions or limited viewpoints. Deep generative models can leverage learned prior knowledge to reconstruct the full map representation from partial observations. Several strategies exist for addressing this problem: (1) training a conditional diffusion model where partial measurements serve as conditioning inputs during training, (2) performing posterior sampling using an unconditional diffusion model, and (3) employing an optimization-based approach to distill the knowledge of the trained model. In this section, we illustrate how the posterior sampling can be utilized in the completion task with our trained unconditional 3D Gaussian diffusion models, where point clouds are the partial measurements.

Completion tasks involving partial measurements can be formulated as an inverse problem, where the objective is to reconstruct the complete map parameters from incomplete observations. Within a Bayesian framework, this process corresponds to posterior inference, where the goal is to estimate the full representation given the observed partial measurements and prior knowledge about the underlying distribution:

$$p(z|x) \propto p(x|z) \cdot p(z) \tag{8.5}$$

where z denotes the model parameters and x represents the incomplete observations, e.g., point clouds.

In the context of reconstruction using generative probabilistic models, the prior distribution $p(z)$ is derived from the learned knowledge encoded within the diffusion model, while the likelihood $p(x|z)$ is determined by the available partial observations. The standard diffusion sampling process considers only the learned data distribution, where each step in the reverse denoising chain generates a new latent sample z_{t-1} conditioned on the previous step: $p(z_{t-1}|z_t)$. The mathematical formulation is given by Equation 2.89.

In posterior sampling, measurement likelihood $p(x|z_{t-1}, z_t)$ is incorporated into the diffusion denoising process, which extends the standard diffusion framework by conditioning each step of the sampling process not only on the previous latent state but also on the observed partial measurements x. However, the term $p(x|z_{t-1}, z_t)$ can not be solved in closed form due to its dependence on time. Approximating this term by $p(x|\hat{z}_0)$, Diffusion Posterior Sampling (DPS) (Chung et al. 2023) is introduced, leading to the following formulation:

$$p(z_{t-1}|z_t, x) \propto p(x|z_{t-1}, z_t) p(z_{t-1}|z_t) \simeq p(x|\hat{z}_{0,t}) p(z_{t-1}|z_t) \tag{8.6}$$

where $\hat{z}_{0,t} = \frac{1}{\sqrt{\bar{\alpha}_t}}(z_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(z_t, t))$ represents the estimated clean parameters at time 0.

Since the diffusion denoising process estimates samples using the score function at each step, the posterior sampling can similarly be expressed in terms of the score function:

$$\nabla_{z_t} \log p(z_t|x) = \nabla_{z_t} \log p(x|z_t) + \nabla_{z_t} \log p(z_t) \tag{8.7}$$

$$\simeq \nabla_{z_t} \log p(x|\hat{z}_{0,t}) + \nabla_{z_t} \log p(z_t) \tag{8.8}$$

where the score function $\nabla_{z_t} \log p(z_t) = -\epsilon_\theta(z_t, t)/\bar{\sigma}_t$ can be estimated by the trained neural net.

Thus, at each time step, the new sample z_{t-1} is derived by incorporating both the prior information from the learned diffusion model and the measurement likelihood:

$$z_{t-1} = \hat{z}_{t-1} + \lambda_p \frac{1-\alpha_t}{\sqrt{\alpha_t}} \nabla_{z_t} \log p(x|\hat{z}_{0,t})$$

where $\hat{z}_{t-1}$ denotes the updated part from the unconditional diffusion prior $p(z_{t-1}|z_t)$ and the latter term ensures that the generated samples are conditioned on the observed partial data.

In our setting, where partial measurements are represented as point clouds, the likelihood function can be modeled as a point distribution estimated using a Gaussian mixture: $\log p(x|z) = \log \sum_j^J \pi_j \mathcal{N}(x|\mu_k, \Sigma_k)$.

Although this approach aims to steer the sampling process toward regions of the latent space that are consistent with both the learned prior and the observed data, the practical effectiveness of this method can be influenced by the characteristics of the likelihood function and the number of inference steps allocated. In the experimental section, we present results from applying diffusion posterior sampling to 3D completion and reconstruction tasks. Specifically, we evaluated the performance of posterior sampling using Gaussian Mixtures as the likelihood function. The results indicate that this approach yielded unsatisfactory performance, suggesting that the direct GMM likelihood may not be well-suited for these applications. This highlights the need for exploring alternative likelihood formulations or refining the integration of likelihood into the diffusion process to improve reconstruction quality.

8.2.2 Optimization-based Completion

Optimization-based completion distills the knowledge embedded in pre-trained deep generative models and leverages the alignment between the generated 3D model parameters and the prior distribution learned through diffusion, using certain loss functions. Additionally, a geometric loss function serves as a constraint, ensuring that the reconstructed 3D model not only adheres to the learned prior but also remains consistent with the available partial measurements.

Among optimization-based approaches, the Diffusion Score Distillation Sampling (SDS) loss has emerged as one of the most influential techniques for distilling prior knowledge from pre-trained deep generative models for 3D generation. This approach was first introduced by DreamFusion (Poole et al. 2022), which extends the knowledge learned in large-scale 2D diffusion models, such as Stable Diffusion (Rombach et al. 2022), to optimize 3D parameters by rendering images from the 3D representation. The primary advantage of this approach is its ability to harness the rich knowledge contained in large-scale 2D models for 3D generation. Building upon this foundation, subsequent research has expanded beyond pure 3D generation to applications in 3D completion, where partial measurements are incorporated as an additional constraint in the objective function. In these methods, geometric loss or image-guidance loss is introduced to enforce consistency with the observed data. Common 3D representations used in such approaches include NeRF and 3DGS.

Inspired by this framework, we extend SDS-based optimization to our completion task using the proposed uncertainty-aware Gaussian representation, employing a pre-trained 4D Gaussian diffusion model introduced in the previous section. Unlike the original SDS formulation, which is primarily designed for 2D-to-3D optimization, our method directly operates in 3D, thereby mitigating the multi-view inconsistency issues often encountered in 2D-to-3D synthesis.

In our approach, the trained diffusion model serves as a frozen prior, where its neural parameters θ remain unchanged. The 3D parameters to be optimized are denoted as z_0, representing the clean data sample within the diffusion framework. During the optimization process, instead of updating the diffusion model itself, we iteratively refine the clean data sample z_0 at the timestep t_0 using a gradient-based optimization. The gradient of the loss function is formulated as:

$$\begin{aligned}\nabla_{z_0} L_{\text{SDS}}(z_0) &= \nabla_{z_0}\mathbb{E}_{t,\epsilon}\Big[w(t)\|\epsilon_\theta(z_t,t)-\epsilon\|^2\Big] && (8.9)\\ &= \mathbb{E}_{t,\epsilon}\Big[w(t)\cdot(\epsilon_\theta(z_t,t)-\epsilon)\ \frac{\partial \epsilon_\theta(z_t,t)}{\partial z_t}\frac{\partial z_t}{\partial z_0}\Big], && (8.10)\end{aligned}$$

where $z_t = \sqrt{\bar{\alpha}_t}z_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$. Note that the last term $\frac{\partial z_t}{\partial z_0}$ can be absorbed into the weighting function $w(t)$. The term $\frac{\partial \epsilon_\theta(z_t,t)}{\partial z_t}$ denotes the deep neural gradient of the Transformer or U-Net. According to the original SDS loss (Poole et al. 2022), the neural gradient item $\frac{\partial \epsilon_\theta(z_t,t)}{\partial z_t}$ can be ignored. Therefore, the gradient formulation can be simplified as:

$$\nabla_{z_0} L_{\text{SDS}}(z_0) = \mathbb{E}_{t,\epsilon}\Big[w(t)\cdot(\epsilon_\theta(z_t,t)-\epsilon)\Big], \tag{8.11}$$

The aforementioned SDS optimization guides the model parameters to align with the prior knowledge learned by 3D diffusion models. However, to effectively solve the inverse problem $p(z_0|x)$, additional constraints are necessary to enforce consistency between the optimized model parameters and the observed data. To achieve this, an additional loss term representing the observations must be incorporated into the optimization framework. Similar to posterior sampling, the constraints imposed by the partial measurements can be formulated as the negative log-likelihood of the observed points, expressed as $-\log p(x|z_0)$:

$$L_{\text{PDF}}(z_0) = \sum_x \Big[\log \sum_j^J \pi_j \mathcal{N}(x|\mu_k,\Sigma_k)\Big]. \tag{8.12}$$

The observation loss can also be calculated based on the SDF of the measured points under the assumption that the distance field satisfies $f(x)=0$ for the points measured on the surface. Mathematically, this can be expressed as:

$$L_{\text{SDF}}(z_0) = \sum_x \|f(x;z_0)\|^2. \tag{8.13}$$

where the SDF function $f(x)$ is defined by the Equation 6.8 in Chapter 6.

The overall loss function can then be written as:

$$L(z_0) = \lambda_{sds}L_{SDS}(z_0) + \lambda_{pdf}L_{PDF}(z_0) + \lambda_{sdf}L_{SDF}(z_0).$$

where λ_{sds}, λ_{pdf} and λ_{sdf} are hyper-parameters that balance the contributions of different loss components to the overall optimization objective.

In the experimental section, we evaluated the performance of the optimization-based method on the completion task, which employs our proposed map representation and utilizes a pre-trained Gaussian diffusion model as a prior. We conducted a comparative analysis of diffusion models trained with sample-based versus noise-based objectives. Our findings indicate that, in contrast to 2D-to-3D SDS optimization, the noise-based objective does not yield satisfactory results for shape generation, while the sample-based objective supports the generation process.

8.3 Experiments

8.3.1 Experimental Setting

In the experiments, we employed the widely used 3D shape dataset - ShapeNet (Chang et al. 2015) as our training set. Initially, the mesh is converted into 3D point clouds as introduced in Section 4.2.3. These point clouds are subsequently modeled using 4D GMMs.

Upon inputting the 3D point clouds derived from 3D assets, we employ the introduced 4D HGMMs to create an implicit representation of the surfaces. This approach enables the reconstruction of surfaces equipped with probabilistic uncertainty measures. Figure 8.3 showcases some examples of GMMs generated for these 3D objects, where the uncertainty and the reconstructed explicit surfaces are also provided. The first row displays the ellipsoids of the Gaussians corresponding to the 3D training assets, with colors representing the mixing weights of the Gaussian components. The second row highlights the uncertainty of the reconstructed surface, with a color gradient from blue (low uncertainty) to red (high uncertainty). Note that the ShapeNet dataset is normalized to a unit scale of approximately 1 meter. Consequently, the reported uncertainty (standard deviation) reflects this normalized scale and does not correspond to the actual physical size of the objects. **The uncertainty in all subsequent generation results is also expressed with respect to this normalized scale**. The final row illustrates the reconstructed surface rendered as an explicit mesh. High uncertainty often appears at the edges of the surface and in regions with large curvatures. The Gaussian parameters as training samples are normalized globally across the whole training dataset.

8.3.2 Diffusion Generation

This section will present the generation results from our trained 3D diffusion model. Specifically, the two backbones of PVCNN and Transformer will be compared qualitatively and quantitatively.

As a qualitative evaluation, Figure 8.4 and 8.5 present the generated 3D geometries from the diffusion model based on the Transformer-based backbone and PVCNN backbone, respectively. The first row displays the 3D Gaussian ellipsoids of the GMMs, with colors representing the mixing weights of the Gaussian components. The second row is the points sampled from the generated distribution. The third row highlights the uncertainty (standard deviation σ) of the reconstructed surface, with a color gradient from blue (low uncertainty) to red (high uncertainty). The final row illustrates the reconstructed surface as an explicit mesh.

The qualitative results include the visualization of 3D Gaussian ellipsoids, point clouds ($N = 10000$) sampled from GMM distribution, reconstructed mesh, and the associated uncertainties. In Figure 8.4, the Transformer-based model shows an effective generation

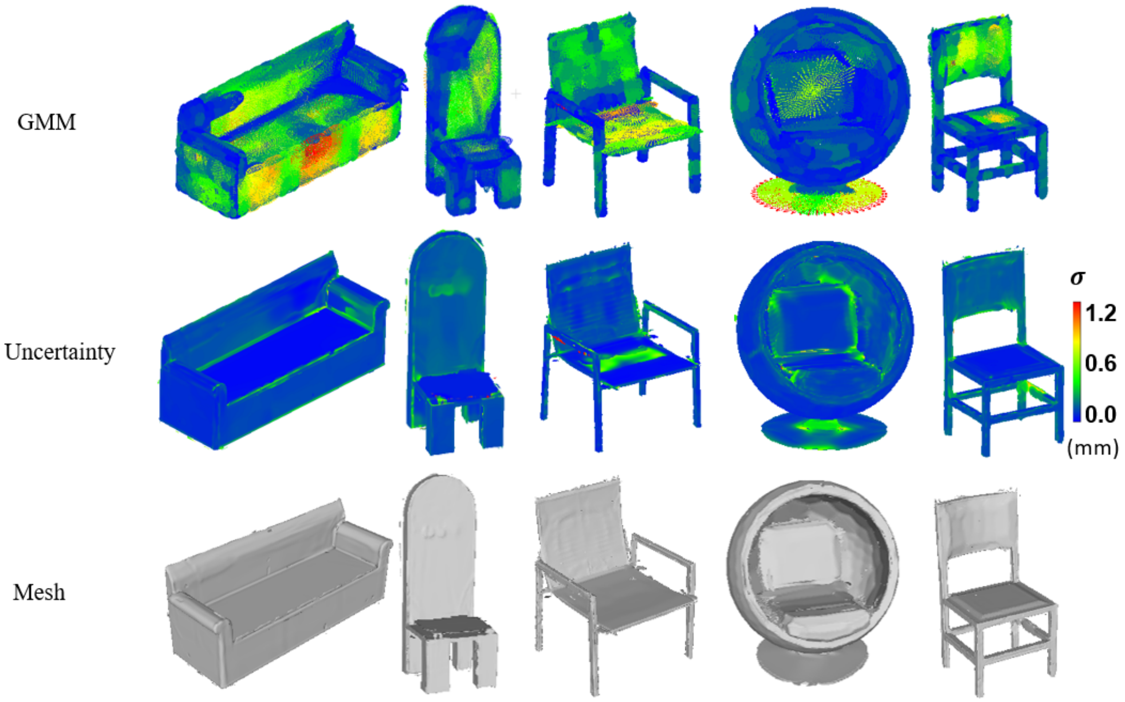


FIGURE 8.3. GMMs training samples

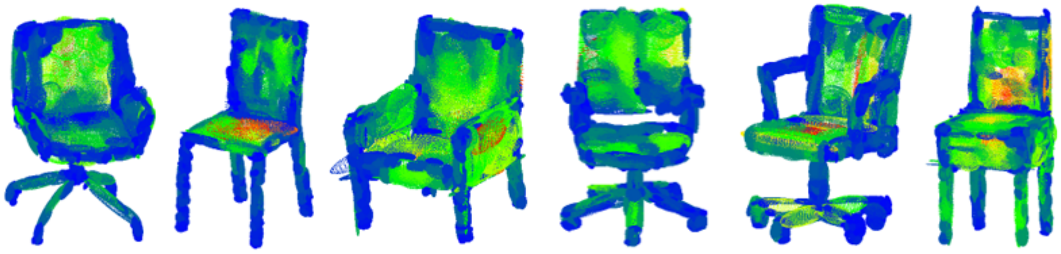
of 3D geometries from the diffusion model. The global features of the 3D geometry for the trained chair category are well captured and generated. Although the surfaces reconstructed from the GMMs look jaggy and not as smooth as the originally modeled surface in the training data, the jaggy parts are often assigned a higher uncertainty measure to indicate the generation quality. In contrast, the generated results from the PVCNN backbone in Figure 8.5 are less robust and with lower fidelity. The Transformer backbone shows superiority in processing 3D geometry with rich information, yielding more complete and smoother surfaces.

Although the Transformer-based model exhibits increased uncertainty in regions with poor surface quality (e.g., jagged areas), suggesting a reasonable correlation between uncertainty and reconstruction fidelity, both methods generally produce low overall uncertainty levels. This may indicate a tendency to underestimate uncertainty.

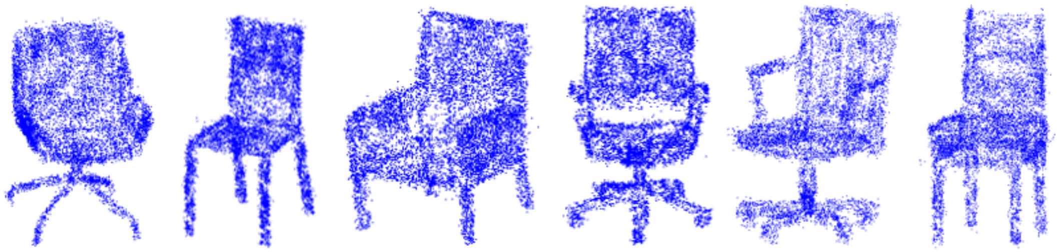
One possible reason is that the uncertainty encoded in the generated Gaussian representations primarily reflects the model's intrinsic uncertainty, while the quality and potential errors introduced by the diffusion process itself are not adequately captured. Factors such as an inappropriate model backbone (as illustrated in Figure 8.5) or incomplete training can significantly affect the model's ability to accurately learn the underlying data distribution and generate realistic 3D structures, yet these sources of uncertainty remain unaccounted for in the final output. Quantifying the uncertainty inherent to the generative model itself remains a non-trivial challenge. Therefore, selecting an appropriate backbone architecture and a well-designed training pipeline is essential to ensure that the generative model captures the data distribution effectively.

Nevertheless, even if the uncertainty estimates primarily reflect the model's intrinsic uncertainty rather than the full range of generation errors, they still provide valuable insight into the quality of the reconstructed geometry—offering a partial but informative measure rather than no uncertainty characterization at all.

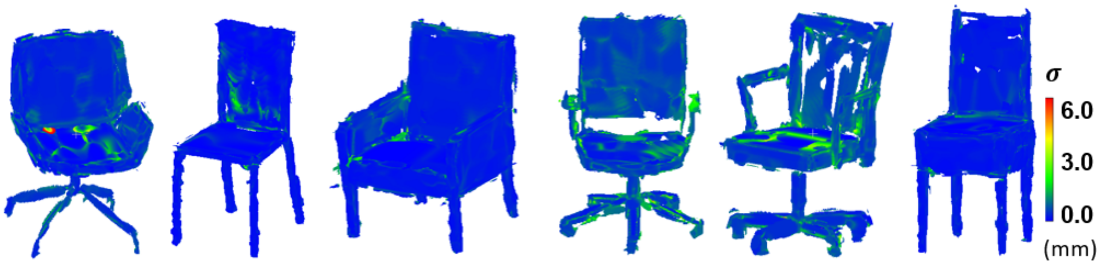
To quantitatively evaluate the performance of different backbones in generating 3D points, we employ three metrics: 1-Nearest Neighbor Accuracy (1-NNA), Coverage (COV),



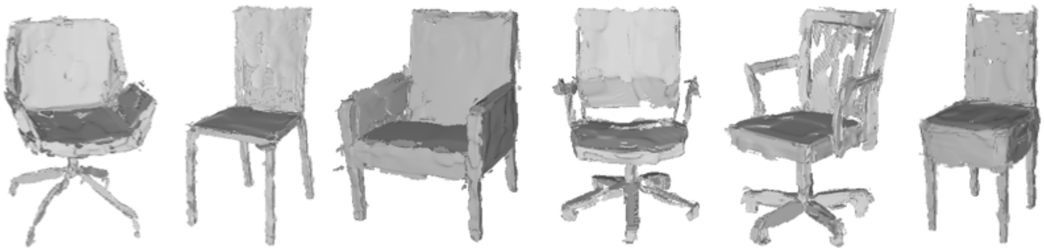
(a) Generated 3D Gaussians: colors denote the mixing weights



(b) Point clouds sampled from the generated model

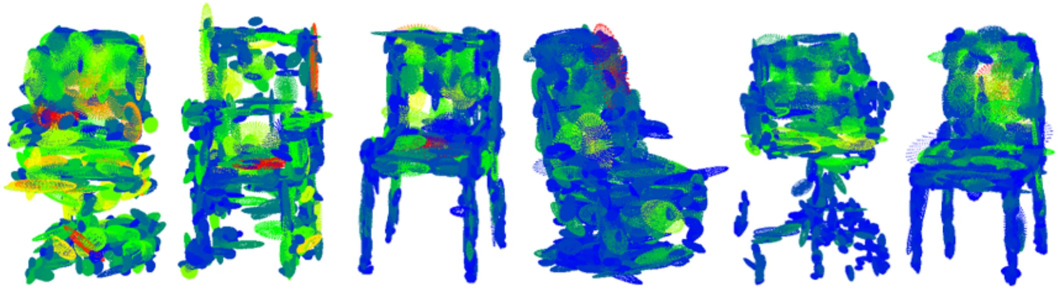


(c) Uncertainty of the reconstructed surface, visualized by colors

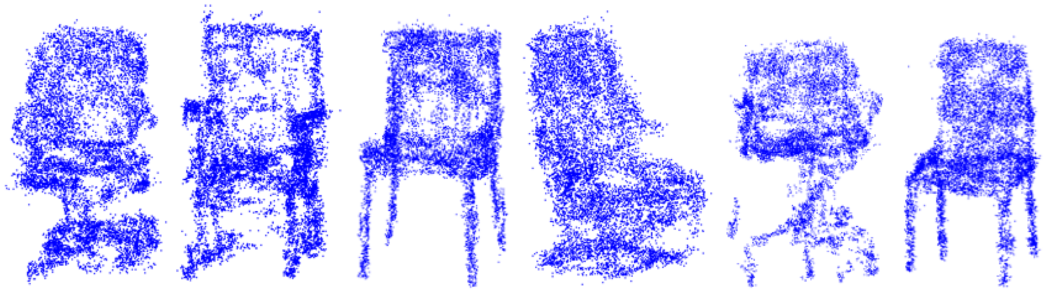


(d) Reconstructed surface (mesh)

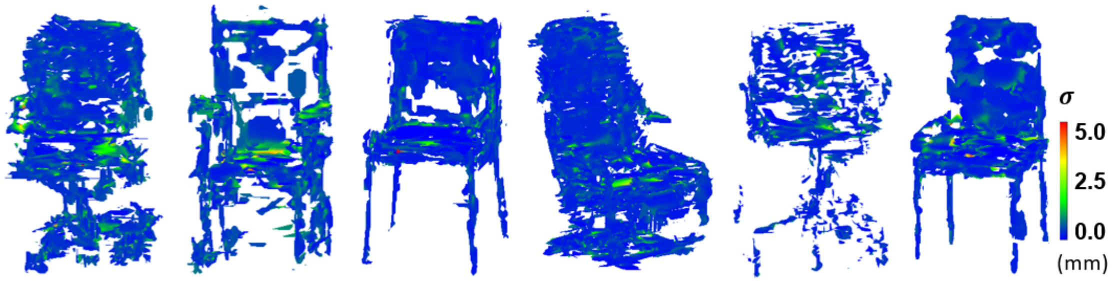
FIGURE 8.4. 3D geometry generation with Transformer backbone



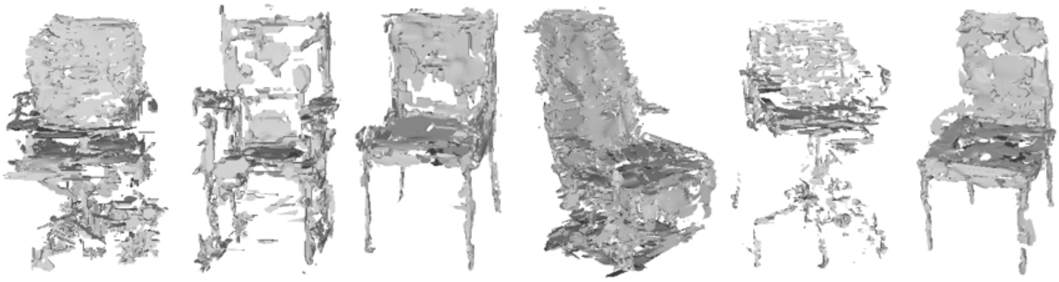
(a) Generated 3D Gaussians: colors denote the mixing weights



(b) Point clouds sampled from the generated model



(c) Uncertainty of the reconstructed surface, visualized by colors



(d) Reconstructed surface (mesh)

FIGURE 8.5. 3D geometry generation with PVCNN backbone and Minimum Matching Distance (MMD). These metrics are calculated using both Chamfer distance (CD) and Earth Mover's Distance (EMD) to quantify the similarity between the generated shapes and the reference dataset, assessing both the quality and diversity of the outputs (Zeng et al. 2022).

- **COV** quantifies the proportion of the reference point cloud that is "covered" by the closest points in the generated point cloud. It is a measure of diversity, indicating the extent to which the generated points represent the variability in the reference set. A higher percentage suggests better coverage. The equation for Coverage is:

  $$\mathrm{COV}(P,Q) = \frac{|\{p \in P : \mathrm{argmin}_{q \in Q} d(p,q)\}|}{|P|} \tag{8.14}$$

  where P and Q denote two sets of point clouds, and in this task, P represents the generated point cloud and Q is the reference dataset. $d(p,q)$ is a distance metric (either CD or EMD) between points p and q.

- **1-NNA** is used to evaluate the distributional similarity between the generated and reference datasets. It is computed as the percentage of points in the generated set for which the nearest point in the reference set is a correct "match" under some classification criterion. The formula for 1-NNA is typically given by:

  $$\text{1-NNA}(P,Q) = \frac{\sum_{p \in P} \mathbf{1}(\mathrm{NN}(p) \in P) + \sum_{q \in Q} \mathbf{1}(\mathrm{NN}(q) \in Q)}{|P| + |Q|} \tag{8.15}$$

  where $\mathbf{1}(\cdot)$ is the indicator function that returns 1 if the condition is true, and 0 otherwise. Here, $\mathrm{NN}(\cdot)$ represents the nearest neighbor of a point.

  An accuracy of 50% suggests perfect indistinguishability under the 1-NN rule, assuming the sizes of the two point sets are equal. This implies that the generative model produces outputs so similar to the real dataset that, on average, each point is just as likely to have its nearest neighbor from within its class as from the other. Accuracy significantly higher or lower than 50% indicates distinguishability between the real and generated datasets.

For consistency in comparisons with the existing baselines, all metrics are computed on point clouds rather than meshed outputs. We also calculated these metrics for a subset of the training dataset to serve as a baseline. The results, detailed in Table 8.1, indicate that the Transformer-based method achieves lower 1-NNA scores and higher Coverage, compared to the PVCNN-based backbone and the other popular baselines, highlighting its superior performance in terms of both quality and diversity in generated 3D point distributions. Note that the results labeled as "Training samples" refer to the evaluation metrics computed on real data samples. These results serve as a baseline and are expected to demonstrate strong performance.

Although this experiment is a pure generation application on our proposed 3D Gaussian representation, the experimental results also show the potential of the inverse problem using the proposed 3D Gaussian to reconstruct unseen parts with a guided 3D diffusion model, and in the meantime, provide the uncertainty of the reconstruction.

8.3.3 Completion

In this experiment, we reconstruct the complete representation from partial observations, specifically point clouds. The partial point clouds are generated by rendering depth images of the test 3D mesh sample from a single viewpoint, as illustrated in Figure 8.6.

TABLE 8.1. Quantitative comparison of 3D point cloud synthesis

	1-NNA (↓, %)		COV (↑, %)	
	CD	EMD	CD	EMD
Transformer-backbone	**58.67**	**55.08**	46.00	49.83
PVCNN-backbone	80.33	81.83	41.83	39.67
Training samples	59.67	60.67	**53.00**	**54.67**
PointFlow(Yang et al. 2019)	62.84	60.57	46.91	48.40
SoftFlow(H. Kim et al. 2020)	59.21	60.05	41.39	47.43
SetVAE (Jinwoo et al. 2021)	58.84	60.57	46.83	44.26
DPF-Net (Klokov et al. 2020)	62.00	58.53	44.71	48.79
DPM (Luo and Hu 2021)	60.05	74.77	44.86	35.50
r-GAN (Achlioptas et al. 2018)	83.69	99.70	24.27	15.13

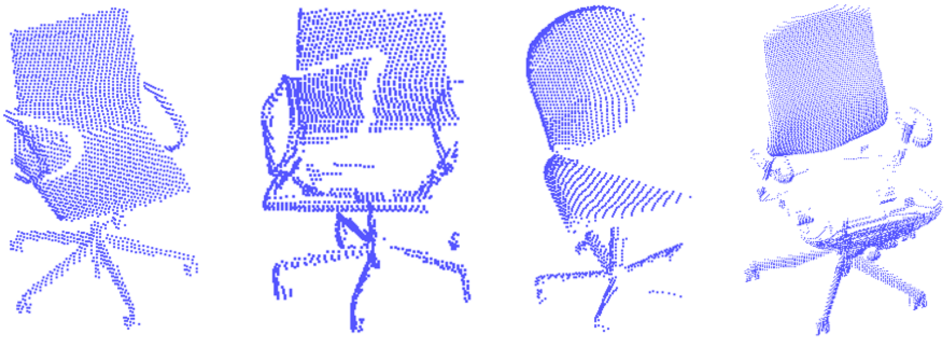


FIGURE 8.6. Examples of incomplete measurements

We evaluate and compare two approaches for reconstruction: posterior sampling and the optimization-based method introduced in Section 8.2.

Posterior Sampling

In practical experiments, posterior sampling using GMM likelihood as geometric constraints for point cloud generation does not work well. The likelihood function as a loss function for optimization is highly non-convex and contains numerous local minima. Thus, optimizing GMM parameters using a gradient-based method is quite sensitive to the parameter initialization. Since the current initialization is a random sample of the distribution, which might be far from the optimal case, the gradient-based optimization for the likelihood function can take many more steps than the limited steps in diffusion models. Thus, the posterior sampling process may become trapped, potentially compromising the quality of the results.

Also, in the diffusion model, only limited inference steps are used for the reverse sampling process (e.g., around 1000 inference steps are commonly used). While this step count is often sufficient for the standard diffusion process, it might be suboptimal when optimizing the GMM likelihood function. If the likelihood requires a finer-grained iterative process—meaning that its gradients need more iterations to effectively guide the posterior toward the global optimum—then a fixed budget of 1000 steps may not provide enough resolution for the sampling process to fully converge.

As an illustrative example, Figure 8.7 presents the optimization of a GMM using a purely gradient-based stochastic gradient descent (SGD) approach. The first five subfigures depict the evolution of the Gaussian components—represented as ellipsoids—over different optimization steps, with color indicating the relative mixing weights of each component. The final subfigure shows the GMM obtained via the EM algorithm, serving as a reference for comparison. As observed, even after 10,000 steps of gradient-based optimization, the GMM parameters remain suboptimal relative to those obtained through EM. This highlights the challenges associated with optimizing GMMs using standard SGD, particularly due to the highly non-convex nature of the likelihood landscape.

Figures 8.8 and 8.9 illustrate two cases of diffusion posterior sampling guided by GMM likelihoods, using the same sample data as shown in Figure 8.7 for conditioning. From left to right, the subfigures visualize the denoising trajectory at time steps $t = 1000, 800, 600, 400, 200$, and the final generated sample x_0 at $t = 0$. When employing a small-scale factor for the GMM likelihood guidance, the influence of the likelihood on the generative process is negligible; the final output remains largely governed by the prior diffusion model and bears little resemblance to the input partial observation. In contrast, when a larger scale factor is applied, the denoising process deviates from the learned data manifold, resulting in samples that no longer resemble the training distribution. In both scenarios, the generated outputs fail to meaningfully incorporate the information from the partial observations, indicating a limitation in directly applying GMM likelihood guidance within the diffusion sampling process.

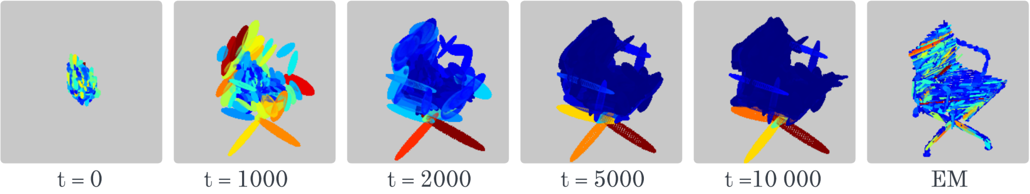


FIGURE 8.7. GMM optimization

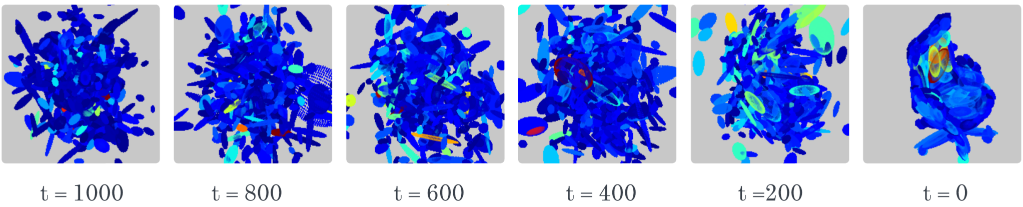


FIGURE 8.8. Posterior sampling: $\lambda_p = 0.5$

Optimization-based Completion

Experimental results of SDS-based optimization in the 3D setting reveal a key difference between diffusion models trained with different objectives. Specifically, 3D diffusion models trained with sample reconstruction loss are able to encode a meaningful shape prior, which can be recovered through SDS optimization in the absence of observation guidance. In contrast, models trained with noise or score-based loss fail to yield meaningful shapes under the same optimization procedure. These results are visualized in Figures 8.10 and 8.11, which show the evolution of the Gaussian parameters representing the 3D model from iteration 0 to 30,000.

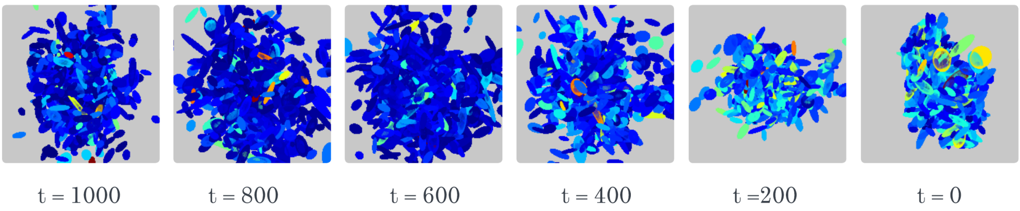


FIGURE 8.9. Posterior sampling: $\lambda_p = 5$

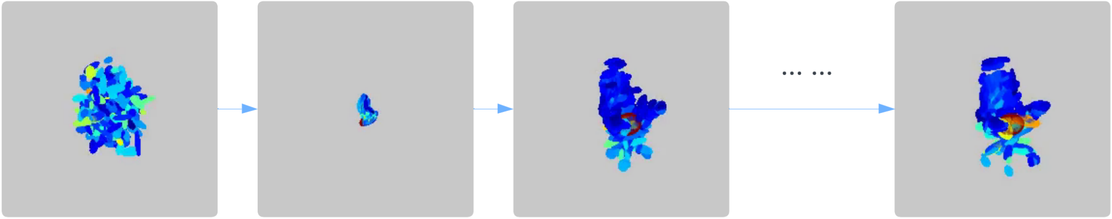


FIGURE 8.10. Pure SDS optimization with a diffusion model trained on sample errors

Figure 8.10 demonstrates that the diffusion model trained on sample-based loss captures a prior that can be effectively leveraged through SDS loss to obtain coherent 3D structures. Conversely, Figure 8.11 shows that optimization using the SDS loss on a model trained with noise-prediction loss does not converge to a plausible shape, indicating that the learned prior in this case is not readily extractable through this method.

Based on these findings, the model trained with sample-based loss was adopted for all subsequent SDS optimization experiments involving observation guidance.

The above qualitative results highlight the effectiveness of leveraging a 3D diffusion model in conjunction with SDS loss as a shape prior for 3D completion tasks. To further evaluate this approach, we present completion results that incorporate both the learned diffusion prior and partial observation guidance. Figure 8.12 illustrates a representative example.

In this experiment, the objective is to reconstruct the full 3D representation of an object (chair) from a partial point cloud observed from a single viewpoint. The first sub-figure of Figure 8.12 shows the input observation, where noticeable portions of the object's surface are missing due to occlusion or limited visibility. The middle panel displays the result of the optimization process, which combines the SDS loss from the diffusion prior with a likelihood loss based on the partial observation. The Gaussian parameters are initialized

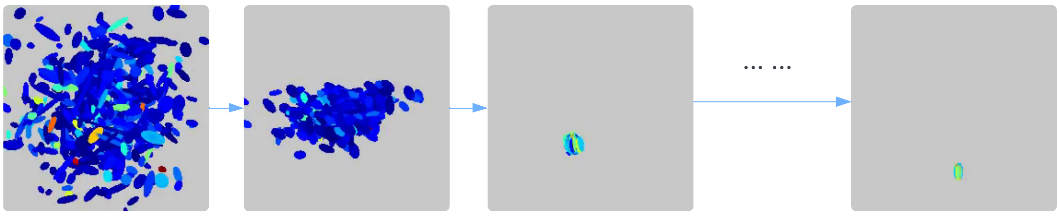


FIGURE 8.11. Pure SDS optimization with a diffusion model trained on noise errors

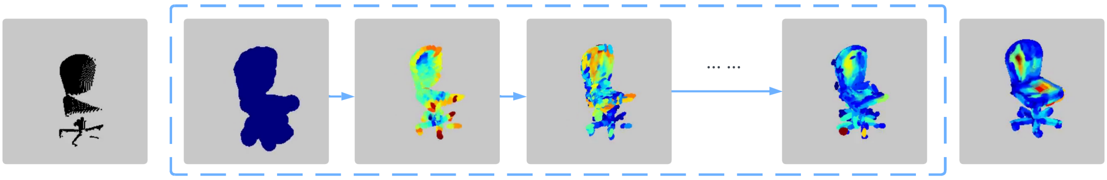


FIGURE 8.12. SDS optimization with the guidance of the partial observations

to roughly capture the observed shape: means are set by subsampling the observed points, covariances are initialized as isotropic Gaussians scaled according to the distance to nearest neighboring points, and mixing weights are uniformly initialized as $1/K$. The right panel provides the ground-truth 3D Gaussian representation for comparison, with color indicating the magnitude of the mixing weights.

As shown, the optimization is able to recover the overall shape structure, including previously missing regions such as the right surface of the chair. However, while the method demonstrates promising reconstruction capability, the final result remains suboptimal and does not fully capture the fine geometric details of the ground-truth representation.

8.3.4 Discussion

The experimental results presented in Section 8.3.2 demonstrate that the proposed 3D Gaussian representation, when combined with a Transformer-based diffusion model, yields strong performance in pure 3D generation tasks on instance-level objects. The associated uncertainty measures provide valuable insights into the quality of the generated geometries. However, the reported uncertainty primarily captures the intrinsic uncertainty of the Gaussian representation itself, while the errors introduced by the generative diffusion model, such as distributional mismatch, mode collapse, or underfitting, are not explicitly accounted for in the final uncertainty estimates.

To address this limitation, two complementary directions can be explored: (1) developing methods to quantify the uncertainty inherent in the diffusion models themselves, and (2) minimizing generative errors by improving the design and training of the diffusion backbone.

Quantifying uncertainty in diffusion models remains an open and challenging area of research. Since the sampling process involves stochastic differential equations or discrete noise transitions, modeling uncertainty propagation throughout the generative process requires advanced techniques such as ensemble methods and Bayesian diffusion formulations.

On the other hand, improving the generative quality requires massive and diverse training data, as well as a carefully constructed architecture and training pipeline that ensures stable and expressive generation. Beyond standard diffusion models, exploring alternative generative architectures—such as hybrid models that combine diffusion and autoregressive mechanisms—may offer additional benefits in improving generation quality.

As detailed in Section 8.3.3, the task of partial map completion using the proposed 3D Gaussian representation poses significant challenges. This is primarily due to the probabilistic and uncertainty-aware nature of the underlying GMM representation, which makes it difficult to directly employ point-wise probability as a guiding loss function during generation. In particular, optimizing the GMM likelihood is highly non-convex, resulting in instability and limited reliability of gradient-based optimization methods.

In contrast, much of the current research in differentiable 3D rendering leverages the straightforward Mean Squared Error (MSE) loss in the image domain. This loss function, due to its smoother and more convex landscape, enables more stable and efficient optimization. This advantage stems from the differentiable rendering pipeline, which connects 3D representations to 2D image observations.

However, in tasks where modeling the uncertainty of the prediction is essential, rather than producing a deterministic point estimate, MSE loss is often inadequate. In such scenarios, log-likelihood-based losses offer a principled way to learn from probabilistic representations. Therefore, developing a more stable and smooth surrogate loss tailored to distribution-based map representations remains an important direction for future research.

Furthermore, in our experiments, we observed that SDS-based optimization for sample generation using diffusion models only succeeded when the diffusion model was trained to minimize data sample error. In contrast, training based on noise prediction led to poor generation performance. This discrepancy highlights a fundamental difference in optimization behavior and underscores further investigation into the characteristics of diffusion models under different training objectives.

To extend the 3D generation framework from instance-level objects to large-scale outdoor environments, we consider two distinct scenarios.

The first scenario involves transferring to structured building models. This is a natural progression, as buildings can be treated as detailed objects like other instance-level objects used in the previous experiments. A crucial step in this case is to curate or simulate a large, diverse set of clean and complete geometric data representing various building types and styles.

The second scenario addresses the more complex and unstructured urban environment. Successfully managing this task requires the development of a more sophisticated world model capable of capturing heterogeneous and irregular geometries under real-world variability. A world model refers to a generative or predictive system that encodes the underlying spatial and temporal structure of the environment, often in a compact or latent representation. Training such a model typically requires massive, diverse datasets and substantial computational resources. Nevertheless, integrating a world model into this task represents a promising research direction for enhancing the robustness and expressiveness of 3D generative mapping and the completion of missing areas in complex environments.

Chapter 9

Conclusion and Outlook

9.1 Summary and Conclusion

In recent advancements in spatial mapping and 3D representation, there has been a growing need for robust, uncertainty-aware methodologies that can adapt to complex urban environments and dynamically update in response to new data. Traditional mapping techniques often struggle with accurately quantifying uncertainty and efficiently processing large-scale data, particularly in structured urban areas where precise and reliable maps are crucial for various applications, such as localization and autonomous navigation.

Building on this need, our previous chapters introduced an innovative 3D map representation that combines GMMs with GPs in a Bayesian framework. This approach leverages the strengths of both methods: GMMs act as informative priors for capturing the global distributions of environmental features, while GPs enable precise modeling of local fine-grained surface structures along with well-calibrated uncertainty estimates. This integration is further optimized through hierarchical structures, derivative observations, and spatial decomposition techniques, such as Octree partitioning, which enhance both the accuracy and computational efficiency of large-scale 3D mapping in urban scenes.

The research begins with the development of an accurate and uncertainty-aware building model tailored for structured objects in urban environments, as introduced in Chapter 5. This method segments buildings into local coordinate frames, where the local surface is modeled using a 2.5D representation. Within each frame, 1D GMMs are employed to capture dominant planar surface structures, followed by local GPs to refine non-planar regions, thereby enhancing the overall geometric fidelity. The resulting models can be converted into probabilistic occupancy maps, enabling evaluation against benchmark approaches using real-world LiDAR point clouds. Experimental results demonstrate that the proposed method achieves the highest AUC scores, validating its effectiveness in both surface modeling and uncertainty quantification.

In Chapter 6, the proposed approach is extended to full 3D representations using implicit SDFs, enabling the modeling of more general and complex geometries beyond the constraints of structured building surfaces. Compared to the previous method, this extension incorporates higher-dimensional GMMs with hierarchical structures and direct regression of the SDF, facilitating the transition from local 2.5D surfaces to comprehensive 3D models. The hierarchical GMMs serve as non-stationary priors for gradient-aware GP inference within a Bayesian framework. The integration is also compatible with alternative GP variants, such as the Logarithm-GP. Additionally, an incremental update mechanism is introduced, allowing new observations to be efficiently incorporated into the map.

Furthermore, the surface reconstruction process and the associated uncertainty derived from the proposed 3D map representations are rigorously analyzed. The Marching Cubes algorithm is employed to extract explicit mesh surfaces from the implicit SDF representations, enabling an evaluation of reconstruction quality and uncertainty quantification.

Comprehensive experiments using both the real-world LUCOOP dataset (Axmann et al. 2023) and synthetic data from the CARLA simulator demonstrate the proposed 3D mapping framework's strengths. The results show that the method achieves the lowest RMSE for surface accuracy, the highest log-likelihood scores for uncertainty estimation, and competitive efficiency in terms of training and inference time when compared to state-of-the-art methods.

Applications of this 3D mapping framework have been extended to uncertainty-aware localization tasks, as well as to 3D generation and completion tasks aimed at filling missing regions with associated uncertainty estimates.

In the localization tasks, using the 3D Gaussian maps achieves superior accuracy compared to the standard NDT approach. Detailed analyses are also provided and reveal insight into the propagation of map uncertainty into pose estimation.

To explore the proposed map framework's potential in 3D generation and completion, we integrate it with state-of-the-art deep generative modeling. A diffusion-based generative model is employed to produce probabilistic 3D geometries using the proposed Gaussian representations. Evaluations on public synthetic instance-level datasets show state-of-the-art performance in terms of generation quality. Furthermore, experiments are conducted to use the model as a prior for reconstructing 3D shapes from partial data, and the results suggest both the challenge and the potential of using the proposed method in this task. Although the experiments are evaluated on synthetic, instance-level datasets, the results clearly demonstrate the framework's potential for future application and transferability to real-world outdoor environments.

9.2 Outlook

To provide a comprehensive and enriched outlook for future research based on the innovative proposals of 3D mapping techniques discussed in this thesis, and considering the rapid advancements in uncertainty-aware 3D mapping of urban scenes, a broadened perspective on potential research avenues is presented in the following.

Uncertainty-Aware Neural Rendering

With the advent of deep-learning methods and innovations in neural radiance fields, exploring uncertainty-aware mappings for rendered radiance fields presents an exciting frontier. This could involve extending popular representations such as NeRF and GS to include robust uncertainty quantifications, enabling more accurate and reliable visual renderings. Such developments could revolutionize fields such as virtual reality, augmented reality, and cinematic visual effects by providing more realistic and physically accurate scenes.

Enhancing 3D Generation with Deep Generative Models

As discussed in Chapter 8, despite strong performance in 3D generation, the proposed method has the limitations: the uncertainty estimates exclude diffusion-specific errors (e.g., mode collapse), and GMM-based optimization suffers from non-convexity, necessitating more stable surrogate losses. Future work should integrate uncertainty quantification for deep generative models (e.g., via Bayesian frameworks), enhance generative quality through hybrid architectures (e.g., integrate diffusion with autoregressive models) and diverse datasets.

Scaling Generation to Real-World Outdoor Environments

Additionally, building on the current use of object-level 3D generation, extending deep generative models to handle large-scale outdoor scene reconstructions offers a vast research domain. First, by collecting and preparing a sufficient training dataset of outdoor buildings, the method can be transferred directly from the object-level 3D assets to structured 3D buildings. Second, in more challenging and unstructured urban environments, a world model with strong generative and predictive capabilities can be employed to provide rich, irregular, and complex prior information for 3D reconstruction and completion. Although training such models demands large-scale, diverse datasets and significant computational resources, the potential benefits are substantial. Incorporating a world model into 3D generative mapping frameworks—capable of reconstructing and completing large-scale scenes with a high degree of uncertainty awareness—would represent a major advancement, with profound implications for safety-critical autonomous systems.

Improving Uncertainty Estimation in Localization

As analyzed in Chapter 7, the proposed mapping framework significantly improves localization accuracy, yet the associated uncertainty estimates are underestimated. This issue arises in part from the presence of highly correlated points, particularly those lying on the same planar surface and exerting similar influence on pose estimation. In future work, a more in-depth investigation is needed to address this limitation.

One promising direction is to develop covariance models that explicitly incorporate geometric relationships between points—such as planar alignment or structural consistency—rather than relying solely on distance-based correlation. This could help to better capture the true uncertainty by accounting for redundancies in the data.

Also, a principled framework should be explored that minimizes optimization-induced errors while leveraging the probabilistic nature of the map representation and its spatial correlations. For instance, reducing the number of effective points used in optimization—without compromising pose accuracy—could improve both computational efficiency and uncertainty estimation.

Toward Dynamic Environment Mapping

The current 3D mapping predominantly focuses on static scenes. There is a significant opportunity to extend these methodologies to dynamic environments, where objects and structural elements evolve over time. Incorporating models that represent the evolution of scenes over time can improve predictive accuracy and planning in autonomous systems. This temporal dynamics within the mapping framework can also be captured by the world model mentioned above.

Enhancing Scalability and Computational Efficiency

While the current framework provides a solid foundation, scaling to accommodate larger datasets remains a challenge. Future work could focus on leveraging GPU optimization effectively for the proposed mapping framework, which entails refining existing algorithms and exploring advanced parallel computing techniques. Utilization of cloud-based architectures could also be a pivotal area, potentially enhancing data processing speeds and reducing the computational load on local systems. This would not only improve the efficiency but also expand the applicability of the 3D mapping solutions to more extensive and complex datasets.

Multimodal Sensor Integration

The integration of heterogeneous sensor data, such as LiDAR, radar, and visual inputs, within the proposed 3D mapping framework is another promising direction. Enhancing sensor fusion capabilities could significantly support the robustness and accuracy of the models, especially in environments with complex sensory inputs. Advanced sensor fusion techniques would allow the framework to provide more reliable and detailed mappings, critical for applications in areas with high sensory variability.

List of Figures

1.1 3D maps 2
1.2 Typical LiDAR point clouds captured in urban environments: colors indicate the semantic labels 3

2.1 LoD2 with roof surface (orange) and wall surface (grey) 8
2.2 Occupancy map 9
2.3 An example of the OctoMap (Hornung et al. 2013) 10
2.4 SDF of a sphere 11
2.5 GMM examples 17
2.6 Impacts of varying hyperparameters: l and σ 24
2.7 Distances to be minimized in both cases 32
2.8 2D NDT grid with fitted Gaussians. 33

4.1 Real world data 48
4.2 Comparison of the Velodyne scan and the Riegl point clouds 48
4.3 Trajectories 49
4.4 LARD Hilbert Map Dataset 49
4.5 Simulation in urban scenario 50
4.6 3D Dataset: ShapeNet (Chang et al. 2015) 51
4.7 Building segmentation 52
4.8 Reconstruct 3D point clouds from the mesh 52

5.1 Depiction of a building model where for each location an uncertainty can be queried (indicated by color). 55
5.2 Illustration of the surface map 56
5.3 Overall workflow. 57
5.4 Example: RANSAC segmentation of a building 58
5.5 Example: GMM segmentation of a building 59
5.6 Local frame 60
5.7 GMM for plane modelling ($K = 5$) 61
5.8 GP for surface modelling 62
5.9 Visualization of different stages of the mapping process. 63
5.10 Surface depth layers 64
5.11 Example of the prior variance 65
5.12 Examples of peak detection for different histogram intervals. 68
5.13 Qualitative results of the building models: uncertainties are denoted by colors. Redder colors indicate higher uncertainties while blue colors denote low uncertainties. 69
5.14 PR curves and AUC comparison: the AUC value of each curve is given in the legend. 70
5.15 Uncertainty evaluation: change of AUC with varying uncertainty thresholds and distances between the predicted points and original training points. 71
5.16 PR curves and AUC comparison of the LARD Dataset. 71

5.17 Computational time. 72

6.1 Overall workflow. 76
6.2 Visualization of the error ellipsoids of Gaussian components in HGMM. (a) is the data. (b) shows four initial Gaussians, and they split into the next level in (c). The new components are checked by the principal curvature to see if they should be further split. (d) presents the final models. 77
6.3 Virtual point generation 79
6.4 Selected training points for GP: (a) shows 16 Gaussian components of GMM. (b) presents the differences between estimated signed distances and measured surface points. The green color indicates small errors as shown in the color bar. The points with large errors are selected for GP training in (c). (d) shows the GP inference results. 81
6.5 Update of the map surface 86
6.6 RMSE with different sub-sampling resolutions 88
6.7 RMSE with different spacing distances 88
6.8 Qualitative results extracted by the marching cubes algorithm 89
6.9 Accuracy evaluation with RMSEs. 90
6.10 Uncertainty evaluation with Log-likelihood. 90
6.11 Accuracy evaluation with RMSEs. 92
6.12 Uncertainty evaluation with Log-likelihood. 92
6.13 Evaluation of the GMMGP framework with LogGP variant. 93
6.14 Computational time: note that blue triangles show the HGMM prior estimation while the orange triangles denote the total computation time as the sum of HGMM and GP. 94

7.1 SDF-based alignment 99
7.2 Errors in pose estimation 105
7.3 Trajectory and Covariance Ellipses 106
7.4 χ^2-test statistic of 3D pose errors 106
7.5 χ^2-test statistics with full covariance matrix 107
7.6 SDF residual calibration 108

8.1 3D Gaussian representation 111
8.2 An illustration of the 3D Gaussian Diffusion framework with the forward diffusing process and the reverse denoising process 112
8.3 GMMs training samples 118
8.4 3D geometry generation with Transformer backbone 119
8.5 3D geometry generation with PVCNN backbone 120
8.6 Examples of incomplete measurements 122
8.7 GMM optimization 123
8.8 Posterior sampling: $\lambda_p = 0.5$ 123
8.9 Posterior sampling: $\lambda_p = 5$ 124
8.10 Pure SDS optimization with a diffusion model trained on sample errors . 124
8.11 Pure SDS optimization with a diffusion model trained on noise errors . . . 124
8.12 SDS optimization with the guidance of the partial observations 125

List of Tables

4.1 List of datasets used in this dissertation. 47
7.1 Localization Accuracy 104
8.1 Quantitative comparison of 3D point cloud synthesis 122

List of Abbreviations

1-NNA 1-Nearest Neighbor Accuracy

3D Three-dimensional

AIC Akaike's Information Criterion

AUC Area Under the Curve

AR Augmented Reality

ARD Automatic Relevance Determination

BCM Bayesian Committee Machines

BIC Bayesian Information Criterion

BIM Building Information Models

CARLA Car Learning to Act

CD Chamfer distance

CLIP Contrastive Language-Image Pre-training

CNN Convolutional Neural Networks

DDPM Denoising Diffusion Probabilistic Models

DEM Digital Elevation Models

DiT Diffusion Transformer

EDF Euclidean Distance Field

EM Expectation-Maximization

EMD Earth Mover's Distance

GAN Generative Adversarial Networks

GIS Geographical Information Science

GMM Gaussian Mixture Model

GMR Gaussian Mixture Regression

GNN Graph Neural Networks

GPIS Gaussian Process Implicit Surface

GP Gaussian Process

GS Gaussian splatting

HGMM Hierarchical Gaussian Mixture Model

ICP Iterative Closest Point

KL Kullback–Leibler

LiDAR Light Detection and Ranging

LoD2 Level of Detail 2

MAE Mean Absolute Error

MAP Maximum A Posteriori

MC Monte Carlo

MCL Monte Carlo Localization

MDL Minimum Description Length

MLE Maximum Likelihood Estimation

MLP Multilayer Perceptron

MMD Minimum Matching Distance

MSE Mean Squared Error

MVN Multivariate Normal Distribution

NDT Normal Distribution Transform

NeRF Neural Radiance Field

OctoMap Octree-based Occupancy Map

OOD out-of-distribution

PCN Point Completion Network

PDF Probability Density Function

PR Precision-Recall

RANSAC Random Sample Consensus

RBF Radial Basis Functions

RMSE Root Mean Squared Error

ROC Receiver Operating Characteristic

SDE Stochastic Differential Equations

SDF Signed Distance Field

SDS Score Distillation Sampling

SGD stochastic gradient descent

SLAM Simultaneous Localization and Mapping

TIN Triangulated Irregular Networks

TLS Terrestrial Laser Scanning

TSDF Truncated Signed Distance Fields

VAE Variational Auto-encoders

VR Virtual Reality

Bibliography

Achlioptas, Panos, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas (2018). "Learning representations and generative models for 3d point clouds". In: *International Conference on Machine Learning*, pp. 40–49.

Akaike, Hirotugu (1974). "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6, pp. 716–723.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein GAN". In: *34th International Conference on Machine Learning*, pp. 214–223.

Axmann, Jeldrik, Rozhin Moftizadeh, Jingyao Su, Benjamin Tennstedt, Qianqian Zou, Yunshuang Yuan, Dominik Ernst, Hamza Alkhatib, Claus Brenner, and Steffen Schön (2023). "LUCOOP: Leibniz university cooperative perception and urban navigation dataset". In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 1–8.

Barron, Jonathan T., Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman (2022). "Mip-nerf 360: Unbounded anti-aliased neural radiance fields". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5470–5479.

Biber, Peter and Wolfgang Strasser (2003). "The normal distributions transform: a new approach to laser scan matching". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3, pp. 2743–2748.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387310738.

Blei, David M. and Michael I. Jordan (2006). "Variational inference for Dirichlet process mixtures". In: *Bayesian Analysis* 1, pp. 121–143.

Brenner, Claus (2016). "Scalable estimation of precision maps in a mapreduce framework". In: *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 1–10.

Brock, Andrew, Theodore Lim, James M. Ritchie, and Nick Weston (2016). "Generative and discriminative voxel modeling with convolutional neural networks". In: *arXiv preprint arXiv:1608.04236*.

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 1877–1901.

Bylow, Erik, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers (2013). "Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions". English. In: *Proceedings of Robotics: Science and Systems*. Vol. 9.

Cai, Ruojin, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan (2020). "Learning gradient fields for shape generation". In: *Proceedings of the 16th European Conference on Computer Vision*, pp. 364–381.

Chang, Angel X., Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu (2015). *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep.

Chen, Guikun and Wenguan Wang (2024). *A Survey on 3D Gaussian Splatting*. arXiv: `2401.03890`.

Chen, Zhiqin and Hao Zhang (2019). “Learning Implicit Fields for Generative Shape Modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948.

Chen, Zilong, Feng Wang, Yikai Wang, and Huaping Liu (2024). “Text-to-3d using gaussian splatting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21401–21412.

Cheng, Yen-Chi, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui (2023). “Sdfusion: Multimodal 3d shape completion, reconstruction, and generation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4456–4465.

Chibane, Julian, Thiemo Alldieck, and Gerard Pons-Moll (2020). “Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6970–6981.

Chibane, Julian, Aymen Mir, and Gerard Pons-Moll (2020). “Neural unsigned distance fields for implicit function learning”. In: *Advances in Neural Information Processing Systems* 33, pp. 21638–21652.

Chou, Gene, Yuval Bahat, and Felix Heide (2023). “Diffusion-SDF: Conditional Generative Modeling of Signed Distance Functions”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2262–2272.

Chung, Hyungjin, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye (2023). “Diffusion posterior sampling for general noisy inverse problems”. In: *Proceedings of the 11th International Conference on Learning Representations*.

Çiçek, Özgün, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger (2016). “3D U-Net: learning dense volumetric segmentation from sparse annotation”. In: *Proceedings of the 19th International Conference on Medical Image Computing and Computer-Assisted Intervention–MICCAI*, pp. 424–432.

Curless, Brian and Marc Levoy (1996). “A volumetric method for building complex models from range images”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pp. 303–312.

Dai, Pinxuan, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu (2024). “High-quality Surface Reconstruction using Gaussian Surfels”. In: *Proceedings of the ACM SIGGRAPH 2024 Conference papers*. Association for Computing Machinery, pp. 1–11.

Davis, Jesse and Mark Goadrich (2006). “The relationship between Precision-Recall and ROC curves”. In: *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240.

Dhawale, Aditya and Nathan Michael (2020). “Efficient Parametric Multi-Fidelity Surface Mapping.” In: *Robotics: Science and Systems*, pp. 1–9.

Doherty, Kevin (2019). “Learning-Aided 3D Occupancy Mapping for Mobile Robots”. In: *IEEE Transactions on Robotics*.

Doherty, Kevin, Jinkun Wang, and Brendan Englot (2017). “Bayesian generalized kernel inference for occupancy map prediction”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3118–3124.

Dong, Haolin, Jincheng Yu, Yuanfan Xu, Zhilin Xu, Zhaoyang Shen, Jiahao Tang, Yuan Shen, and Yu Wang (2022). “MR-GMMapping: Communication efficient multi-robot mapping system via Gaussian mixture model”. In: *IEEE Robotics and Automation Letters* 7.2, pp. 3294–3301.

Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun (2017). “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16.

Dragiev, Stanimir, Marc Toussaint, and Michael Gienger (2011). "Gaussian process implicit surfaces for shape estimation and grasping". In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2845–2850.

Eckart, Benjamin, Kihwan Kim, and Jan Kautz (2018). "HGMR: Hierarchical gaussian mixtures for adaptive 3d registration". In: *Proceedings of the European conference on computer vision*, pp. 705–721.

Eckart, Benjamin, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz (2016). "Accelerated generative models for 3D point cloud data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5497–5505.

Ehambram, Aaronkumar, Luc Jaulin, and Bernardo Wagner (2022). "Hybrid interval-probabilistic localization in building maps". In: *IEEE Robotics and Automation Letters* 7.3, pp. 7059–7066.

Elfes, Alberto (1989). "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6, pp. 46–57.

Engel, Paulo Martins and Milton Roberto Heinen (2010). "Incremental learning of multivariate gaussian mixture models". In: *Advances in Artificial Intelligence: 20th Brazilian Symposium on Artificial Intelligence*, pp. 82–91.

Gadelha, Matheus, Rui Wang, and Subhransu Maji (2018). "Multiresolution tree networks for 3d point cloud processing". In: *Proceedings of the European Conference on Computer Vision*, pp. 103–118.

Gan, Lu, Ray Zhang, Jessy W. Grizzle, Ryan M. Eustice, and Maani Ghaffari (2020). "Bayesian Spatial Kernel Smoothing for Scalable Dense Semantic Mapping". In: *IEEE Robotics and Automation Letters* 5.2, pp. 790–797.

Gao, Yuan and Wei Dong (2023). "An Integrated Hierarchical Approach for Real-Time Mapping With Gaussian Mixture Model". In: *IEEE Robotics and Automation Letters* 8.11, pp. 6891–6898.

Genova, Kyle, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser (2020). "Local deep implicit functions for 3d shape". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4857–4866.

Gentil, Cedric L., Othmane-Latif Ouabi, Lan Wu, Cedric Pradalier, and Teresa Vidal-Calleja (2024). "Accurate Gaussian-Process-Based Distance Fields With Applications to Echolocation and Mapping". In: *IEEE Robotics and Automation Letters* 9.2, pp. 1365–1372.

Goel, Kshitij, Nathan Michael, and Wennie Tabib (2023). "Probabilistic point cloud modeling via self-organizing Gaussian mixture models". In: *IEEE Robotics and Automation Letters* 8.5, pp. 2526–2533.

Goel, Kshitij and Wennie Tabib (2023). "Incremental Multimodal Surface Mapping via Self-Organizing Gaussian Mixture Models". In: *IEEE Robotics and Automation Letters* 8.12, pp. 8358–8365.

Goodfellow, Ian J, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

Gropp, Amos, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman (2020). "Implicit geometric regularization for learning shapes". In: *arXiv preprint arXiv:2002.10099*.

Guédon, Antoine and Vincent Lepetit (2024). "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5354–5363.

Guizilini, Vitor and Fabio Ramos (2016). "Large-scale 3D scene reconstruction with hilbert maps". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3247–3254.

Han, Luxin, Fei Gao, Boyu Zhou, and Shaojie Shen (2019). "FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots". In: *Proceedings* *of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4423–4430.

Hansen, Mark H. and Bin Yu (2001). "Model Selection and the Principle of Minimum Description Length". In: *Journal of the American Statistical Association* 96.454, pp. 746–774.

Hensman, James, Nicolò Fusi, and Neil D. Lawrence (2013). "Gaussian processes for Big data". In: *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, pp. 282–290.

Hinton, Geoffrey E. (2002). "Training products of experts by minimizing contrastive divergence". In: *Neural Computation* 14.8, pp. 1771–1800.

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 6840–6851.

Hornung, Armin, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard (2013). "OctoMap: an efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous Robots* 34.3, pp. 189–206.

Huang, Binbin, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao (2024). "2D Gaussian Splatting for Geometrically Accurate Radiance Fields". In: *ACM SIGGRAPH Special Interest Group on Computer Graphics and Interactive Techniques Conference*. SIGGRAPH '24: Special Interest Group on Computer Graphics and Interactive Techniques Conference, pp. 1–11.

Huang, Huaiyang, Haoyang Ye, Yuxiang Sun, and Ming Liu (2020). "GMMLoc: Structure Consistent Visual Localization With Gaussian Mixture Models". en. In: *IEEE Robotics and Automation Letters* 5.4, pp. 5043–5050.

Huang, Tao, Heng Peng, and Kun Zhang (2017). "Model selection for Gaussian Mixture Models". In: *Statistica Sinica* 27.1, pp. 147–169.

Huang, Tianxin, Zhiwen Yan, Yuyang Zhao, and Gim Hee Lee (2025). *ComPC: Completing a 3D Point Cloud with 2D Diffusion Priors*.

Huang, Yifeng and Kamal Gupta (2008). "RRT-SLAM for motion planning with motion and map uncertainty for robot exploration". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1077–1082.

Isaacson, Seth, Pou-Chun Kung, Mani Ramanagopal, Ram Vasudevan, and Katherine A. Skinner (2023). *LONER: LiDAR Only Neural Representations for Real-Time SLAM*.

Ivan, Jean-Paul A., Todor Stoyanov, and Johannes A. Stork (2022). "Online Distance Field Priors for Gaussian Process Implicit Surfaces". In: *IEEE Robotics and Automation Letters* 7.4, pp. 8996–9003.

Jadidi, Ghaffari M., Jaime V. Miro, and Gamini Dissanayake (2018). "Gaussian processes autonomous mapping and exploration for range-sensing mobile robots". In: *Autonomous Robots* 42.2, pp. 273–290.

Jadidi, Ghaffari M., Jaime V. Miro, Rafael Valencia, and Juan Andrade-Cetto (2014). "Exploration on continuous Gaussian process frontier maps". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6077–6082.

Javanmardi, Ehsan, Yanlei Gu, Mahdi Javanmardi, and Shunsuke Kamijo (2019). "Autonomous vehicle self-localization based on abstract map and multi-channel LiDAR in urban area". In: *IATSS research* 43.1, pp. 1–13.

Jiang, Chiyu, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. (2020). "Local implicit grid representations for 3d scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6001–6010.

Jinwoo, Kim, Yoo Jaehoon, Lee Juho, and Hong Seunghoon (2021). "Setvae: Learning hierarchical composition for generative modeling of set-structured data". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15059–15068.

Jun, Heewoo and Alex Nichol (2023). “Shap-e: Generating conditional 3d implicit functions”. In: *arXiv preprint arXiv:2305.02463*.

Karras, Tero, Samuli Laine, and Timo Aila (2019). “A Style-Based Architecture for GANs”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410.

Kasten, Yoni, Ohad Rahamim, and Gal Chechik (2024). “Point cloud completion with pretrained text-to-image diffusion models”. In: *Advances in Neural Information Processing Systems* 36.

Kerbl, Bernhard, Georgios Kopanas, Thomas Leimkühler, and George Drettakis (2023). “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4, pp. 139–1.

Khan, Mouhyemen, Akash Patel, and Abhijit Chatterjee (2020). “Multi-Sparse Gaussian Process: Learning based Semi-Parametric Control”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5327–5334.

Kim, Hyeongju, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim (2020). “Softflow: Probabilistic framework for normalizing flow on manifolds”. In: *Advances in Neural Information Processing Systems* 33, pp. 16388–16397.

Kim, Soohwan and Jonghyuk Kim (2014). “Recursive Bayesian Updates for Occupancy Mapping and Surface Reconstruction”. In: *Proceedings of the Australasian Conference on Robotics and Automation*.

— (2015). “GPmap: A unified framework for robotic mapping based on sparse Gaussian processes”. In: *Field and service robotics*, pp. 319–332.

Kingma, Diederik P and Max Welling (2014). “Auto-Encoding Variational Bayes”. In: *Proceedings of the 2nd International Conference on Learning Representations*.

Klokov, Roman, Edmond Boyer, and Jakob Verbeek (2020). “Discrete point flow networks for efficient point cloud generation”. In: *Proceedings of the 16th European Conference*.

Kolbe, Thomas H (2009). “Representing and exchanging 3D city models with CityGML”. In: *3D geo-information sciences*, pp. 15–31.

Lee, Bhoram, Clark Zhang, Zonghao Huang, and Daniel D. Lee (2019). “Online Continuous Mapping using Gaussian Process Implicit Surfaces”. In: *Proceedings of the International Conference on Robotics and Automation*, pp. 6884–6890.

Li, Chun-Liang, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov (2018). “Point cloud gan”. In: *arXiv preprint arXiv:1810.05795*.

Li, Peter Zhi Xuan, Sertac Karaman, and Vivienne Sze (2024). “Gmmap: Memory-efficient continuous occupancy map using gaussian mixture model”. In: *IEEE Transactions on Robotics* 40, pp. 1339–1355.

Li, Yangyan, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen (2018). “Pointcnn: Convolution on x-transformed points”. In: *Advances in neural information processing systems* 31.

Liao, Ziwei, Binbin Xu, and Steven L Waslander (2024). “Toward General Object-level Mapping from Sparse Views with 3D Diffusion Priors”. In: *8th Annual Conference on Robot Learning*.

Liu, Haitao, Jianfei Cai, Yi Wang, and Yew Soon Ong (2018). “Generalized robust Bayesian committee machine for large-scale Gaussian process regression”. In: *International Conference on Machine Learning*, pp. 3131–3140.

Liu, Minghua, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su (2024). “One-2-3-45++: Fast Single Image to 3D Objects with Consistent Multi-View Generation and 3D Diffusion”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10072–10083.

Liu, Minghua, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su (2024). "One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization". In: *Advances in Neural Information Processing Systems* 36.

Liu, Ruoshi, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick (2023). "Zero-1-to-3: Zero-shot one image to 3d object". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9298–9309.

Liu, Zhijian, Haotian Tang, Yujun Lin, and Song Han (2019). "Point-Voxel CNN for Efficient 3D Deep Learning". In: *Advances in Neural Information Processing Systems*.

Lorensen, William E. and Harvey E. Cline (1998). "Marching cubes: A high resolution 3D surface construction algorithm". In: *Seminal graphics: pioneering efforts that shaped the field*, pp. 347–353.

Luo, Shitong and Wei Hu (2021). "Diffusion Probabilistic Models for 3D Point Cloud Generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2837–2845.

Lyu, Xiaoyang, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi (2024). "3DGSR: Implicit Surface Reconstruction with 3D Gaussian Splatting". In: *ACM Transactions on Graphics* 43.6.

Magnusson, Martin, Achim Lilienthal, and Tom Duckett (2007). "Scan registration for autonomous mining vehicles using 3D-NDT". In: *Journal of Field Robotics* 24.10, pp. 803–827.

Marriott, Richard T., Alexander Pashevich, and Radu Horaud (2018). "Plane-extraction from depth-data using a Gaussian mixture regression model". In: *Pattern Recognition Letters* 110, pp. 44–50.

Martens, Wolfram, Yannick Poffet, Pablo Ramón Soria, Robert Fitch, and Salah Sukkarieh (2017). "Geometric Priors for Gaussian Process Implicit Surfaces". In: *IEEE Robotics and Automation Letters* 2.2, pp. 373–380.

Melas-Kyriazi, Luke, Christian Rupprecht, and Andrea Vedaldi (2023). "Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12923–12932.

Melkumyan, Arman and Fabio Ramos (2009). "A sparse covariance function for exact Gaussian process inference in large datasets". In: *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 1936–1942.

Mescheder, Lars, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger (2019). "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470.

Metzer, Gal, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or (2023). "Latent-nerf for shape-guided generation of 3d shapes and textures". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12663–12673.

Miao, Yu, Alan Hunter, and Ioannis Georgilas (2021). "Parameter reduction and optimisation for point cloud and occupancy mapping algorithms". In: *Sensors* 21.21, p. 7004.

Mildenhall, Ben, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng (2021). "NeRF: representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1, pp. 99–106.

Mo, Shentong, Enze Xie, Ruihang Chu, Lanqing Hong, Matthias Niessner, and Zhenguo Li (2023). "Dit-3d: Exploring plain diffusion transformers for 3d shape generation". In: *Advances in neural information processing systems* 36, pp. 67960–67971.

Moravec, Hans and Alberto Elfes (1985). "High resolution maps from wide angle sonar". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. Vol. 2, pp. 116–121.

Mu, Yuxuan, Xinxin Zuo, Chuan Guo, Yilin Wang, Juwei Lu, Xiaofeng Wu, Songcen Xu, Peng Dai, Youliang Yan, and Li Cheng (2025). "Gsd: View-guided gaussian splatting diffusion for 3d reconstruction". In: *Proceedings of the European Conference on Computer Vision*, pp. 55–72.

Müller, Norman, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner (2023). "DiffRF: Rendering-Guided 3D Radiance Field Diffusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4328–4338.

Museth, Ken (2013). "VDB: High-resolution sparse volumes with dynamic topology". In: *ACM transactions on graphics* 32.3.

Nam, Gimin, Mariem Khlifi, Andrew Rodriguez, Alberto Tono, Linqi Zhou, and Paul Guerrero (2022). "3d-ldm: Neural implicit 3d shape generation with latent diffusion models". In: *arXiv preprint arXiv:2212.00842*.

Newcombe, Richard A., Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon (2011). "KinectFusion: Real-time dense surface mapping and tracking". In: *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136.

Nichol, Alex, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen (2022). "Point-e: A system for generating 3d point clouds from complex prompts". In: *arXiv preprint arXiv:2212.08751*.

Nichol, Alexander Quinn and Prafulla Dhariwal (2021). "Improved denoising diffusion probabilistic models". In: *International conference on machine learning*, pp. 8162–8171.

O'Callaghan, Simon, Fabio T. Ramos, and Hugh Durrant-Whyte (2009). "Contextual occupancy maps using Gaussian processes". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1054–1060.

O'Callaghan, Simon T. and Fabio T. Ramos (2012). "Gaussian process occupancy maps". In: *The International Journal of Robotics Research* 31.1, pp. 42–62.

O'Meadhra, Cormac, Wennie Tabib, and Nathan Michael (2018). "Variable resolution occupancy mapping using gaussian mixture models". In: *IEEE Robotics and Automation Letters* 4.2, pp. 2015–2022.

Oleynikova, Helen, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran (2016). "Continuous-time trajectory optimization for online UAV replanning". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5332–5339.

Oleynikova, Helen, Alexander Millane, Zachary Taylor, Enric Galceran, Juan Nieto, and Roland Siegwart (2016). "Signed distance fields: A natural representation for both mapping and planning". In: *RSS 2016 workshop: geometry and beyond-representations, physics, and scene understanding for robotics*.

Oleynikova, Helen, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto (2017). "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1366–1373.

Ortiz, Joseph, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam (2022). "isdf: Real-time neural signed distance fields for robot perception". In: *arXiv preprint arXiv:2204.02296*.

Pan, Yue, Yves Kompis, Luca Bartolomei, Ruben Mascaro, Cyrill Stachniss, and Margarita Chli (2022). "Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5331–5338.

Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2021). “Normalizing flows for probabilistic modeling and inference”. In: *Journal of Machine Learning Research* 22.57, pp. 1–64.

Park, Jeong Joon, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove (2019). “Deepsdf: Learning continuous signed distance functions for shape representation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174.

Pearson, Erik, Kevin Doherty, and Brendan Englot (2022). “Improving obstacle boundary representations in predictive occupancy mapping”. In: *Robotics and Autonomous Systems* 153, p. 104077.

Peters, Torben and Claus Brenner (2019). “Automatic generation of large point cloud training datasets using label transfer”. In: *Proceedings of the 39th Annual Science and Technology Conference DGPF*, pp. 68–82.

Poole, Ben, Ajay Jain, Jonathan T Barron, and Ben Mildenhall (2022). “Dreamfusion: Text-to-3d using 2d diffusion”. In: *arXiv preprint arXiv:2209.14988*.

Pumarola, Albert, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer (2021). “D-nerf: Neural radiance fields for dynamic scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10318–10327.

Qi, Charles R, Hao Su, Kaichun Mo, and Leonidas J Guibas (2016). “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *arXiv preprint arXiv:1612.00593*.

Qi, Charles R., Li Yi, Hao Su, and Leonidas J Guibas (2017). “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *arXiv preprint arXiv:1706.02413*.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever (2021). “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*, pp. 8748–8763.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”. In: *arXiv preprint arXiv:1511.06434*.

Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever (2021). “Zero-Shot Text-to-Image Generation”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139, pp. 8821–8831.

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian processes for machine learning*. The MIT Press.

Ratliff, Nathan, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa (2009). “CHOMP: Gradient optimization techniques for efficient motion planning”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 489–494.

Ren, Zhiyuan, Minchul Kim, Feng Liu, and Xiaoming Liu (2024). “TIGER: Time-Varying Denoising Model for 3D Point Cloud Generation with Diffusion Process”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9462–9471.

Roessle, Barbara, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, Angela Dai, and Matthias Nießner (2024). “L3dg: Latent 3d gaussian diffusion”. In: *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11.

Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer (2022). “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695.

Saarinen, Jari P., Henrik Andreasson, Todor Stoyanov, and Achim J. Lilienthal (2013). “3D normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments". In: *The International Journal of Robotics Research* 32.14, pp. 1627–1644.

Saharia, Chitwan, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. (2022). "Photorealistic text-to-image diffusion models with deep language understanding". In: *Advances in neural information processing systems* 35, pp. 36479–36494.

Schnabel, Ruwen, Roland Wahl, and Reinhard Klein (2007). "Efficient RANSAC for point-cloud shape detection". In: *Computer graphics forum* 26.2, pp. 214–226.

Schwarz, Gideon (1978). "Estimating the dimension of a model". In: *The Annals of Statistics*, pp. 461–464.

Senanayake, Ransalu and Fabio Ramos (2017). "Bayesian Hilbert maps for dynamic continuous occupancy mapping". In: *Conference on Robot Learning*, pp. 458–471.

Shan, Tixiao, Jinkun Wang, Brendan Englot, and Kevin Doherty (2018). "Bayesian Generalized Kernel Inference for Terrain Traversability Mapping". In: *Proceedings of the 2nd Conference on Robot Learning*. Vol. 87, pp. 829–838.

Shim, Jaehyeok, Changwoo Kang, and Kyungdon Joo (2023). "Diffusion-based signed distance fields for 3d shape generation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20887–20897.

Sitzmann, Vincent, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein (2020). "Implicit neural representations with periodic activation functions". In: *Advances in neural information processing systems* 33, pp. 7462–7473.

Smola, Alex and Peter Bartlett (2000). "Sparse Greedy Gaussian Process Regression". In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press.

Snelson, Edward and Zoubin Ghahramani (2005). "Sparse Gaussian Processes using Pseudo-inputs". In: *Advances in Neural Information Processing Systems*. Vol. 18. MIT Press.

Sohl-Dickstein, Jascha, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli (2015). "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *International Conference on Machine Learning*, pp. 2256–2265.

Song, Jiaming, Chenlin Meng, and Stefano Ermon (2021). *Denoising Diffusion Implicit Models*. arXiv preprint arXiv:2010.02502.

Song, Yang, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole (2020). "Score-based generative modeling through stochastic differential equations". In: *arXiv preprint arXiv:2011.13456*.

Srivastava, Shobhit and Nathan Michael (2016). "Approximate continuous belief distributions for precise autonomous inspection". In: *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 74–80.

— (2018). "Efficient, multifidelity perceptual representations via hierarchical gaussian mixture models". In: *IEEE Transactions on Robotics* 35.1, pp. 248–260.

Sucar, Edgar, Shikun Liu, Joseph Ortiz, and Andrew J Davison (2021). "imap: Implicit mapping and positioning in real-time". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6229–6238.

Sung, Hsi Guang (2004). *Gaussian mixture regression and classification*. PhD dissertation, Rice University.

Tabib, Wennie, Kshitij Goel, John W Yao, Mosam Dabhi, Curtis Boirum, and Nathan Michael (2019). "Real-Time Information-Theoretic Exploration with Gaussian Mixture Model Maps". In: *Robotics: Science and Systems*, pp. 1–9.

Tabib, Wennie, Cormac O'Meadhra, and Nathan Michael (2018). "On-Manifold GMM Registration". en. In: *IEEE Robotics and Automation Letters* 3.4, pp. 3805–3812.

Tancik, Matthew, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben P. Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar (2022). "Block-NeRF: Scalable

Large Scene Neural View Synthesis". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 8238–8248.

Tang, Jiaxiang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu (2025). "Lgm: Large multi-view gaussian model for high-resolution 3d content creation". In: *European Conference on Computer Vision*, pp. 1–18.

Tang, Jiaxiang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng (2023). "Dreamgaussian: Generative gaussian splatting for efficient 3d content creation". In: *arXiv preprint arXiv:2309.16653*.

Tang, Junshu, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen (2023). "Make-It-3D: High-Fidelity 3D Creation from A Single Image with Diffusion Prior". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France, pp. 22762–22772.

Thrun, Sebastian, Christian Martin, Yufeng Liu, Dirk Hahnel, Rosemary Emery-Montemerlo, Deepayan Chakrabarti, and Wolfram Burgard (2004). "A real-time expectation-maximization algorithm for acquiring multiplanar maps of indoor environments with mobile robots". In: *IEEE Transactions on Robotics and Automation* 20.3, pp. 433–443.

Titsias, Michalis (2009). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*. Vol. 5, pp. 567–574.

van den Oord, Aaron, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). "Pixel Recurrent Neural Networks". In: *International Conference on Machine Learning*, pp. 1747–1756.

van den Oord, Aaron, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu (2016). "Conditional Image Generation with PixelCNN Decoders". In: *Advances in Neural Information Processing Systems*. Vol. 29, pp. 4790–4798.

van den Oord, Aaron, Oriol Vinyals, and Koray Kavukcuoglu (2017). "Neural Discrete Representation Learning". In: *Advances in Neural Information Processing Systems*. Vol. 30.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems*.

Vega-Brown, William R., Marek Doniec, and Nicholas G. Roy (2014). "Nonparametric Bayesian inference on multivariate exponential families". In: *Advances in Neural Information Processing Systems*. Vol. 27.

Wang, Haochen, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich (2023). "Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12619–12629.

Wang, Jinkun and Brendan Englot (2016). "Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1003–1010.

Wang, Peng, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang (2021). "NeuS: learning neural implicit surfaces by volume rendering for multi-view reconstruction". In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*.

Wang, Yue, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon (2019). "Dynamic graph CNN for learning on point clouds". In: *ACM Transactions on Graphics* 38.5, pp. 1–12.

Wiesmann, Louis, Tiziano Guadagnino, Ignacio Vizzo, Nicky Zimmerman, Yue Pan, Haofei Kuang, Jens Behley, and Cyrill Stachniss (2023). "LocNDF: Neural Distance Field Mapping for Robot Localization". In: *IEEE Robotics and Automation Letters* 8.8, pp. 4999–5006.

Williams, Oliver and Andrew Fitzgibbon (2007). "Gaussian process implicit surfaces". In: *Gaussian Processes in Practice Workshop*.

Wu, Jiajun, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum (2016). "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems* 29.

Wu, Lan, Cedric L. Gentil, and Teresa Vidal-Calleja (2023). "Pseudo Inputs Optimisation for Efficient Gaussian Process Distance Fields". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7249–7255.

Wu, Lan, Cedric Le Gentil, and Teresa Vidal-Calleja (2025). "VDB-GPDF: Online Gaussian Process Distance Field With VDB Structure". In: *IEEE Robotics and Automation Letters* 10.1, pp. 374–381.

Wu, Lan, Ki M. B. Lee, Cedric L. Gentil, and Teresa Vidal-Calleja (2023). "Log-GPIS-MOP: A Unified Representation for Mapping, Odometry, and Planning". In: *IEEE Transactions on Robotics* 39.5, pp. 4078–4094.

Wu, Lan, Ki M. B. Lee, Liyang Liu, and Teresa Vidal-Calleja (2021). "Faithful Euclidean Distance Field From Log-Gaussian Process Implicit Surfaces". In: *IEEE Robotics and Automation Letters* 6.2, pp. 2461–2468.

Xie, Jianwen, Yifei Xu, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu (2021). "Generative pointnet: Deep energy-based learning on unordered point sets for 3d generation, reconstruction and classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14976–14985.

Yang, Guandao, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan (2019). "Pointflow: 3d point cloud generation with continuous normalizing flows". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4541–4550.

Yariv, Lior, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall (2023). "BakedSDF: Meshing Neural SDFs for Real-Time View Synthesis". In: *ACM SIGGRAPH 2023 Conference Proceedings*.

Yi, Taoran, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang (2023). "Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors". In: *arXiv preprint arXiv:2310.08529*.

Yu, Mulin, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai (2024). "Gsdf: 3dgs meets sdf for improved rendering and reconstruction". In: *Advances in Neural Information Processing Systems*.

Yu, Zehao, Torsten Sattler, and Andreas Geiger (2024). "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes". In: *ACM Transactions on Graphics* 43.6, pp. 1–13.

Yuan, Wentao, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert (2018). "Pcn: Point completion network". In: *Proceedings of IEEE International Conference on 3D Vision*, pp. 728–737.

Zeng, Xiaohui, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. (2022). "Lion: Latent point diffusion models for 3d shape generation". In: *Advances in Neural Information Processing Systems* 35, pp. 10021–10039.

Zhao, Cheng, Su Sun, Ruoyu Wang, Yuliang Guo, Jun-Jun Wan, Zhou Huang, Xinyu Huang, Yingjie Victor Chen, and Liu Ren (2024). "TCLC-GS: Tightly Coupled LiDAR-Camera Gaussian Splatting for Autonomous Driving". In: *European Conference on Computer Vision*, pp. 91–106.

Zhao, Hengshuang, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun (2021). "Point Transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268.

Zhong, Xingguang, Yue Pan, Jens Behley, and Cyrill Stachniss (2023). “SHINE-Mapping: Large-Scale 3D Mapping Using Sparse Hierarchical Implicit Neural Representations”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8371–8377.

Zhong, Yiming, Xiaolin Zhang, Yao Zhao, and Yunchao Wei (2024). “DreamLCM: Towards High Quality Text-to-3D Generation via Latent Consistency Model”. In: *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 1731–1740.

Zhou, Hongyu, Jiahao Shao, Lu Xu, Dongfeng Bai, Weichao Qiu, Bingbing Liu, Yue Wang, Andreas Geiger, and Yiyi Liao (2024). “Hugs: Holistic urban 3d scene understanding via gaussian splatting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21336–21345.

Zhou, Linqi, Yilun Du, and Jiajun Wu (2021). “3d shape generation and completion through point-voxel diffusion”. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5826–5835.

Zhu, Zihan, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys (2022). “NICE-SLAM: Neural Implicit Scalable Encoding for SLAM”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12776–12786.

Zou, Qianqian, Claus Brenner, and Monika Sester (2023). “Gaussian Process Mapping of Uncertain Building Models With GMM as Prior”. In: *IEEE Robotics and Automation Letters* 8.10, pp. 6579–6586.

Zou, Qianqian and Monika Sester (2021). “Incremental map refinement of building information using LiDAR point clouds”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 43, pp. 277–282.

— (2022). “Uncertainty representation and quantification of 3d building models”. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives); XLIII-B2-2022* 43, pp. 335–341.

— (2024). “3D Uncertain Implicit Surface Mapping Using GMM and GP”. In: *IEEE Robotics and Automation Letters* 9.11, pp. 10559–10566.

Acknowledgements

The path to completing this dissertation has been marked by both professional challenges and personal adversity. I would like to express my deepest gratitude to my supervisor, Professor Monika Sester, for her kindness, unwavering support, critical feedback, and inspiring guidance throughout this research journey. Her commitment to academic excellence and meticulous attention to detail have profoundly shaped the direction and quality of my work over the past four years.

I am also sincerely grateful to Professor Claus Brenner for his insightful suggestions and thought-provoking discussions, which broadened my perspective and enhanced the rigor of this work. I greatly valued our conversations and the many new ideas that emerged through them, which significantly enriched the depth of this research.

My appreciation also extends to the professors and peers of the DFG-funded research training group, "Integrity and Collaboration in Dynamic Sensor Networks" (i.c.sens.), for the stimulating exchanges—particularly during regular professional presentations and workshops—which provided both intellectual challenge and a strong sense of community, which were quite helpful to my academic development. I would also like to thank my colleagues at the Institute of Cartography and Geoinformatics at Leibniz University Hannover for their collaborative spirit, engaging discussions, and technical support throughout my doctoral studies.

I am especially thankful to the many professors and researchers from both the robotics and geoinformatics communities for their valuable insights and feedback on my work during conference presentations and discussions, both virtually and in person, including at ISPRS, IEEE ICRA, IEEE IV, and CFE-CMStatistics.

To my family and friends, I owe my heartfelt thanks for their encouragement and support, particularly during the difficult periods of the COVID-19 pandemic. Their thoughtful conversations, emotional support, and timely distractions sustained me through the more demanding phases of this journey, making it not only possible but also meaningful and fulfilling.

Finally, I would like to express my sincere appreciation to the faculty and staff at the university for the indispensable assistance in managing resources and platforms that support my experimental work, which was critical to the success of this dissertation.

Qianqian Zou

Curriculum Vitae

Personal Information

Place and Date of Birth: Zhejiang, China - 31 January 1994

Education

09.2020-05.2025	**PhD Student** Institute of Cartography and Geoinformatics Leibniz University Hannver, Germany Project: i.c.sens., Integrity and Collaboration in Dynamic Sensor Networks Thesis Topic: 3D Mapping with Probabilistic Uncertainty Measures Advisor: Prof. Dr.-Ing. habil. Monika Sester
10.2017-07.2020	**Master of Science in Earth-oriented Space Science and Technology (ESPACE)** Technical University of Munich, Germany Specialization: Satellite Navigation and Remote Sensing Advisor: Prof. Urs Hugentobler
09.2016-07.2020	**Master of Science in Geoinformation Science** Wuhan University, China Advisor: Prof. Luliang Tang
09.2012-07.2016	**Bachelor of Science in Geoinformation Science** Wuhan University, China

Experience

06.2019-05.2020	**R&D Engineer, Intern** Robert Bosch GmbH, Germany
12.2018-04.2019	**Working student of the German Aerospace Agency Research Project** Technical University of Munich, Germany
09.2016-03.2018	**Student Researcher, Chinese National High-tech R&D Program** Wuhan University, China
07.2015-10.2015	**GIS software developer, Intern** Leadmap, China

Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

(Eine vollständige Liste der Wiss. Arb. ist beim Geodätischen Institut, Nienburger Str. 1, 30167 Hannover erhältlich.)

Nr. 390	DOROZYNSKI, Mareike Marianne:	Image Classification and Retrieval in the Context of Silk Heritage using Deep Learning (Diss. 2023)
Nr. 391	KNABE, Annike:	New Concepts for Gravity Field Recovery using Satellites (Diss. 2023)
Nr. 392	KALIA, Andre:	Landslide activity detection based on nationwide Sentinel-1 PSI datasets (Diss. 2023)
Nr. 393	BROCKMEYER, Marco:	Modellierung von Bodenbewegungen anhand heterogener Messverfahren am Beispiel der niedersächsischen Landesfläche (Diss. 2023)
Nr. 394	ZHANG, Mingyue:	Characteristics and Benefits of Differential Lunar Laser Ranging (Diss. 2023)
Nr. 395	DENNIG, Dirk:	Entwicklung eines kinematischen Profilvermessungssystems am Beispiel Kranbahnvermessung (Diss. 2024)
Nr. 396	FUEST, Stefan:	Nudging travelers to societally favorable routes by means of cartographic symbolization (Diss. 2024)
Nr. 397	MOFTIZADEH, Rozhin:	Advanced Particle Filtering for Vehicle Navigation based on Collaborative Information (Diss. 2024)
Nr. 398	VASSILEVA, Magdalena Stefanova:	Satellite Radar Interferometry for Geohazards: from ground deformation to processes understanding (Diss. 2024)
Nr. 399	MALINOVSKAYA, Anna:	Statistical Process Monitoring of Networks (Diss. 2024)
Nr. 400	BANNERT, Jörn:	Der Einfluss von Straßenverkehrslärm und Umgehungsstraßen auf Grundstücks-werte in Ortslagen - Bestimmung mittels Expertenbefragung nach der Delphi-Methode (Diss. 2024)
Nr. 401	AXMANN: Jeldrik:	Maximum consensus localization using LiDAR (Diss. 2024)
Nr. 402	TENNSTEDT, Benjamin:	Concept and Evaluation of a Hybridization Scheme for Atom Interferometers and Inertial Measurement Units (Diss. 2024)
Nr. 403	HAKE, Frederic:	Schadenserkennung an Bauwerken mittels maschinellem Lernens (Diss. 2025)
Nr. 404	KARIMIDOONA, Ali:	On Integrity Prediction for Network-RTK Positioning in Urban Environments (Diss. 2025)
Nr. 405	ORTEGA, Mabel:	Domain Adaptation for Deforestation Detection in Remote Sensing: Addressing Class Imbalance and Performance Estimation (Diss. 2025)
Nr. 406	KUPRIYANOV, Alexey:	Investigation of Optical Accelerometry and Novel Satellite Formations for Future Gravimetry Missions (Diss. 2025)
Nr. 407	BREVA, Yannick:	On the Observation Quality of Robot-based GNSS Antenna Calibration for Determining Codephase Corrections (Diss. 2025)
Nr. 408	GUO, Zelong:	Co- and Post-seismic Slip Models Inferred from InSAR Geodesy (Diss. 2025)
Nr. 409:	YUAN, Yunshuang:	Collective Perception: A Fully-Sparse Deep Learning Framework for Multi-Agent Data Fusion (Diss. 2025)
Nr. 410	SU, Jingyao:	Towards interval-based autonomous integrity monitoring: Error bounding and uncertainty propagation (Diss. 2025)
Nr. 411	VINCENT, Asha:	Clock Networks for Geodetic Applications (Diss. 2025)
Nr. 412	KAMALASANAN, Vinu:	Control of Walking Behavior in Shared Spaces using Augmented Reality (Diss. 2025)
Nr. 413	TRUSHEIM, Philipp:	Kooperative Positionierung mittels Bündelausgleichung mit dynamischen Objekten (Diss. 2025)
Nr. 414	KRÖGER, Johannes:	Estimation and Validation of multi-GNSS multi-Frequency Phase Center Corrections (Diss. 2025)
Nr. 415	RUWISCH, Fabian:	GNSS Feature Maps – Robust Lane-level Accurate GNSS Navigation In Urban Trenches (Diss. 2025)
Nr. 416	WANG, Wandi:	Multi-Sensor Remote Sensing Time Series Analysis of Landslide Processes (Diss. 2025)
Nr. 417	ZOU, Qianqian:	3D mapping with probabilistic uncertainty measures (Diss. 2025)