Towards Urban Digital Twins With Gaussian Splatting, Large-Language-Models, and Cloud Mapping Services

by

Kyle (Yilin) Gao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2025

© Kyle (Yilin) Gao 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	**Dr. Michael Yang, Professor** Dept. of Computer Science, University of Bath
Supervisor:	**Dr. Jonathan Li, Professor** Dept. of Geography & Environmental Management, Cross Appointed — Dept. of Systems Design Engineering, University of Waterloo
Supervisor:	**Dr. Linlin Xu, Assistant Professor** Dept. of Geomatics Engineering, University of Calgary Adjunct — Dept. of Systems Design Engineering, University of Waterloo
Internal Member:	**Dr. Alexander Wong, Professor** Dept. of Systems Design Engineering, Cross-Appointed — David R. Cheriton School of Computer Science, Cross-Appointed — School of Optometry and Vision Science, Cross-Appointed — Dept. of Mechanical and Mechatronics Engineering, University of Waterloo
Internal Member:	**Dr. Sirisha Rambathla, Assistant Professor** Dept. of Management Sciences, Cross-Appointed — Dept. of Systems Design Engineering, University of Waterloo
Internal-External Member:	**Dr. Michael A. Chapman, Professor** Dept. of Civil Engineering, Toronto Metropolitan University Adjunct — Dept. of Geography & Environmental Management, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This doctoral thesis follows the manuscript option per the Department of Systems Design Engineering's graduate guidelines. It includes three required manuscripts in Chapters 3, 4, and 5, based on multi-authored papers with minor formatting adjustments. An additional case study is included in Section 5.5 of Chapter 5. The background and literature review sections contain excerpts from my survey manuscript.

As the lead author of these papers and manuscripts, I was responsible for conceptualization, methodology, code writing, validation experiments, investigation, data curation, writing, and visualization. My co-authors (Dening Lu, Hongjie He, Liangzhi Li, Nan Chen) contributed to manuscript review & editing and suggestions. My supervisors, Prof. Jonathan Li and Prof. Linlin Xu, provided funding and guidance throughout the process and feedback on draft manuscripts. The references for the four research papers/preprints and the survey manuscript are listed below:

1. **Kyle Gao**, Dening Lu, Hongjie He, Linlin Xu, Jonathan Li, (Published in 2025, manuscript completed early 2024). "Enhanced 3D Urban Scene Reconstruction and Point Cloud Densification using Gaussian Splatting and Google Earth Imagery", IEEE Transactions on Geoscience and Remote Sensing, DOI: 10.1109/TGRS.2025.3536169.

2. **Kyle Gao**, Liangzhi Li, Hongjie He, Dening Lu, Linlin Xu, Jonathan Li, 2024. "Gaussian Building Mesh (GBM): Extract a Building's 3D Mesh with Google Earth and Gaussian Splatting", submitted to IEEE Transactions on Geoscience and Remote Sensing (under peer-review). arXiv preprint arXiv:2501.00625.

3. **Kyle Gao**, Dening Lu, Liangzhi Li, Nan Chen, Hongjie He, Linlin Xu, Jonathan Li, 2025. "Digital Twin Buildings: 3D Modeling, GIS Integration, and Visual Descriptions Using Gaussian Splatting, ChatGPT/Deepseek, and Google Maps Platforms", submitted to IEEE Geoscience and Remote Sensing Letters (under peer-review). arXiv preprint arXiv:2502.05769.

4. **Kyle Gao**, Dening Lu, Liangzhi Li, Nan Chen, Hongjie He, Linlin Xu, Jonathan Li, 2025. "Instructor-Worker Large Language Model System for Policy Recommendation: a Case Study on Air Quality Analysis of the January 2025 Los Angeles Wildfires", submitted to Journal of Applied Earth Observation and Geoinformation (under peer-review). arXiv preprint arXiv:2503.00566.

5. **Kyle Gao**, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, Jonathan Li, 2022. "NeRF: Neural Radiance Field in 3D Vision, A Comprehensive Review". arXiv preprint arXiv:2210.00379.

Abstract

Urban Digital Twin cities are powerful tools leveraging 3D modelling, multi-source data, and online geographic information systems to assist urban planning, infrastructure management, and decision-making. However, traditional 3D reconstruction methods face significant limitations in meeting the demands of next-generation Digital Twins. These conventional approaches struggle with scalability, lack photorealism, and struggle with computational efficiency. This thesis addresses these challenges by proposing a comprehensive framework that leverages recent advances in AI-driven 3D reconstruction and large language models (LLMs), paving the way for efficient, accurate, and user-accessible Urban Digital Twins of the future.

To address the challenge of generating accurate and efficient large-scale outdoor 3D point clouds from remote sensing images, this thesis introduces a novel pipeline that combines Google Earth imagery with Gaussian Splatting techniques. This approach enables dense and photorealistic 3D reconstruction at the city scale, significantly improving view synthesis quality and reconstruction accuracy over previous methods. The proposed framework is demonstrated through the reconstruction of the City of Waterloo, representing one of the first large-scale applications of Gaussian Splatting in urban modelling.

For the challenge of efficiently generating precise 3D mesh representations of individual buildings, a new pipeline is developed that integrates open-set foundational object segmentation models with the latest Gaussian Splatting techniques. The pipeline leverages Google Earth Studio, allowing users to generate detailed 3D building models from basic inputs such as an address, postal code, geographic coordinates, or place name.

This thesis presents the Digital Twin Buildings framework to improve the accessibility and usability of digital twin systems. It combines mesh-based 3D reconstruction with cloud mapping services and a novel multi-agent system powered by large language models. This framework enables the automated retrieval, visualization, and analysis of 3D building models alongside GIS-integrated data. It supports natural language interactions, automated code generation, and multi-modal analytics. A case study on air quality analysis during a major wildfire event illustrates the system's potential to support decision-making and interdisciplinary urban research.

By unifying these contributions, this thesis demonstrates a significant step toward accurate and efficient digital twin systems for urban modelling. It provides a blueprint for developing the next generation of Urban Digital Twins accessible to users through AI-assisted tools supporting natural language interactions.

Acknowledgements

I would like to thank my parents first and foremost, who have supported me along every step in pursuit of this PhD. This journey would not have been possible without you. You worked tirelessly when we first immigrated to Canada more than twenty years ago to provide a better future for me and my little sister. I hope you can look at my academic journey with pride. Whenever you're ready, please have a relaxing retirement. Go have fun and travel! You deserve it after these long years of hard work!

I would like to thank my maternal grandmother, who came to Canada fifteen years ago to take care of my little sister and me while my parents worked long hours in pursuit of the Canadian immigrant's dream. Your Szechuan cuisine will always be in my core memory as the most authentic and delicious Chinese food; no restaurant can ever compare! You have lived a difficult life. Please take good care of your health!

I would like to thank my supervisors, Professor Jonathan Li and Professor Linlin Xu. Your guidance and support made me the scholar I am today. I did not realize it during the first two years of my PhD. But now, every time I look at my CV and reflect upon what I have accomplished, I realize how much you have done for me. You will forever have my gratitude.

I would like to thank my friends and colleagues, especially Hongjie He and Dening Lu. We have shared many laughs and many moments of inspiration. I hope our friendship lasts a lifetime! I hope you find great success! I hope your futures are filled with happiness!

To my little sister. – The future may seem bleak and uncertain at times. Don't worry. The world will always have a place for exceptional people, and you are one of them. When AI takes over, you will be assigned a good position by our AI overlords!

Dedication

I dedicate this thesis to my paternal grandfather.

You endured a lifetime of poverty and strife as a subsistence farmer in 20th-century China. Despite not being afforded the opportunity for a good education, you valued learning and the pursuit of knowledge above all, teaching yourself calculus starting with just an elementary school-level education. The world is an unfair place. If you had been born in the 21st century, or a more privileged time and place, you would have enjoyed a career as a great scholar.

I still remember the sadness in your eyes when I told you, by that hospital bed, that I did not wish to pursue a PhD. I am a blunt and direct person, and I apologize for disappointing you in those final moments. Here we are five years later, at the end of my PhD journey. It is unfortunate that you were not able to witness my thesis defence in person, nor see my recent accomplishments. I know that it would have brought you great joy! I hope you can look down upon me and be proud that my work has reached thousands of scholars around the world.

If I become a scholar of international renown, I hope to immortalize your name in this thesis as a dedication for future students and scholars to find.

To my grandfather, **Runxiang Gao**.

Table of Contents

Examining Committee ii

Author's Declaration iii

Statement of Contributions iv

Abstract vi

Acknowledgements vii

Dedication viii

List of Figures xiv

List of Tables xvi

List of Abbreviations xvii

List of Symbols xix

1 Introduction 1

1.1 Motivation 3

1.2 Research Objectives 4

1.3 Contributions 6

1.4 Outline of the Thesis 8

2 Background and Related Works **10**

2.1 Relevant 3D Representations . . . 10

2.1.1 Explicit Representations . . . 11

2.1.2 Implicit Representations . . . 13

2.1.3 Truncated Signed Distance Function Integration . . . 14

2.2 Photogrammetry and Structure from Motion Preprocessing . . . 17

2.3 Novel View Synthesis . . . 18

2.3.1 Rendering . . . 18

2.3.2 Differentiable Volume Rendering and NeRF . . . 19

2.3.3 Differentiable Rasterization and Gaussian Splatting . . . 20

2.4 3D Gaussian Splatting . . . 21

2.4.1 Gaussian Distributions . . . 21

2.4.2 Gaussian Splatting . . . 21

2.4.3 2D Gaussian Splatting . . . 30

2.5 Large Language Models and Transformers . . . 35

2.6 Evaluation Metrics . . . 37

2.7 Google Earth Studio . . . 39

2.8 3D Reconstruction from 2D Images . . . 40

2.9 Developments in Neural Radiance Fields . . . 40

2.9.1 Geometry Extraction . . . 41

2.9.2 NeRF in Remote Sensing and/or Urban Environments . . . 42

2.10 Developments in Gaussian Splatting . . . 42

2.10.1 Gaussian Splatting in Remote Sensing and Urban Environments . . . 43

2.10.2 3D Geometry Extraction . . . 44

2.11 Developments in Large Language Models . . . 44

3 Enhanced 3D Urban Scene Reconstruction and Point Cloud Densification using Gaussian Splatting and Google Earth Imagery **47**

3.1 Introduction . . . 48

3.2 Methodology . . . 49

3.2.1 Study Area . . . 49

3.2.2 Google Earth Studio . . . 50

3.2.3 Multi-View-Stereo Dense 3D Reconstruction . . . 52

3.2.4 Gaussian Splatting . . . 53

3.2.5 Metrics . . . 58

3.3 Experiments and Discussion . . . 59

3.3.1 Novel View Synthesis in the Study Area . . . 60

3.3.2 3D Novel View Synthesis of Bungee-NeRF Scenes . . . 60

3.3.3 3D Reconstruction of the Study Area . . . 64

3.3.4 Experiment with Google Earth Studio's Simulated Illumination . . . 68

3.3.5 Discussions . . . 70

3.4 Chapter Summary . . . 73

4 Gaussian Building Mesh: Extract a Building's 3D Mesh with Google Earth and Gaussian Splatting **75**

4.1 Introduction . . . 76

4.2 Additional Background . . . 78

4.2.1 SAM2 and Grounding DINO . . . 78

4.2.2 GS2Mesh . . . 79

4.3 Methodology . . . 81

4.3.1 Google Earth Studio Dataset . . . 82

4.3.2 Building Segmentation and Mesh Refinement . . . 82

4.3.3 2DGS+ . . . 87

4.3.4 TSDF Fusion and Mesh Extraction . . . 87

4.4 Metrics . . . 88
4.5 Experiments and Discussion . . . 89
4.5.1 Preliminary Experiments . . . 89
4.5.2 Mask Refinement . . . 90
4.5.3 View Synthesis . . . 91
4.5.4 3D Building Mesh Extraction . . . 93
4.5.5 Ablation Studies . . . 95
4.5.6 Discussion . . . 96
4.6 Chapter Summary . . . 98

5 Digital Twin Buildings: 3D Modelling, GIS Integration, and Visual Descriptions Using Gaussian Splatting 99
5.1 Introduction . . . 100
5.2 Additional Background . . . 102
5.2.1 ChatGPT/Deepseek Application Programming Interface . . . 102
5.2.2 Google Maps Platform . . . 103
5.2.3 Vision Language Models . . . 103
5.3 Methodology . . . 105
5.3.1 GBM Mesh Extraction . . . 105
5.3.2 Google Maps Platform Integration . . . 106
5.3.3 Multi-Agent LLM Analysis of Multi-View/Scale Images . . . 107
5.3.4 Automated Multi-Agent Code Generation and Execution . . . 109
5.3.5 Natural Language Processing Metrics . . . 111
5.4 Experiments and Discussion . . . 113
5.4.1 Keyword Extraction . . . 113
5.4.2 Captioning . . . 116
5.4.3 Visualization . . . 117
5.4.4 Discussion of Digital Twin Buildings . . . 117

5.5 Application: Air Quality Analysis . . . 118
5.5.1 Google Maps Platform Air Quality API . . . 118
5.5.2 Los Angeles Wildfire . . . 119
5.5.3 Multi-Agent Analysis of Air Quality . . . 122
5.5.4 Multi-Agent Health Recommendation and Benchmark . . . 125
5.5.5 Discussion of the Application-based Case Study . . . 130
5.6 Chapter Summary . . . 131

6 Conclusions and Recommendations for Future Research 133
6.1 Conclusions . . . 133
6.2 Recommendations for Future Research . . . 135
6.2.1 3D Urban Reconstruction . . . 135
6.2.2 LLM-based Geospatial Data Analysis . . . 137

References 139

List of Figures

2.1 Diagram of the constituents of a 3D mesh. 12
2.2 Diagram of TSDF depth projection. 15
2.3 Per-voxel triangle mesh in Marching Cubes. 16
2.4 Diagram of the 3DGS framework. 22
2.5 Diagram of 3DGS densification. 27
2.6 Diagram of 2D Gaussian Splats 30
2.7 Diagram of the Transformer architecture. 36

3.1 Plot of camera poses, and reconstruction for the study area. 50
3.2 Ground truth, generated images, and visualization of Gaussian means of our Waterloo scene at different altitudes and orientations. 59
3.3 Ground truth vs rendered images of the New York and San Francisco scenes at different altitudes and orientations. 62
3.4 Novel view synthesis (rendered images) comparison of NeRF-based results. 63
3.5 Multi-View-Stereo densification results visualization. 65
3.6 Comparison of 3DGS point cloud densification. 66
3.7 Post cropping and registration visualization of reference point cloud and deviation in local Hausdorff distance from reference. 67
3.8 Effect of sun illumination condition on novel view synthesis with high tilt angle on the test set. 68

4.1 Flow chart of GBM. 78

4.2 Example of SAM-2 mask inconsistencies. 84
4.3 Example of the effects of depth blurring. 88
4.4 Example of DLNR failure case. 89
4.5 Example of mask refinement. 91
4.6 Visualization of 2D novel view synthesis results. 92
4.7 Visualization of 3D coloured mesh results. 94

5.1 Diagram of the proposed Digital Twin Building framework. 100
5.2 Diagram of a typical VLM framework. 105
5.3 Diagram of our Google Map Platform API Services Integration Tool. . . . 107
5.4 Diagram of Multi-Agent LLM processing multi-view-multi-scale images. . . 108
5.5 Instructor-Worker System for automated cloud-based dataset and data stream retrieval. 109
5.6 Box plot of image-to-keyword perplexity. 114
5.7 Box plot of CLIP Scores per model (mean in green). 114
5.8 Box plot of BLIP Scores per model (mean in green). 115
5.9 Box plot of PAC Scores per model (mean in green). 115
5.10 Average CLIP score of each building image captioning model for each different scene. 116
5.11 Visualization of DTB results. 118
5.12 PM levels comparisons in three select cities. 120
5.13 BERTScore distribution comparison for health recommendations. 127
5.14 Bar graph of BERTScores for health recommendations by day (air quality) and health class. 128

List of Tables

2.1 Real valued spherical harmonic functions Ylm on the unit sphere ($r = 1$) in Cartesian coordinates up to degree 3 30

3.1 Google Earth Studio City Scenes Information 52

3.2 Training set PSNR, and Test set PSNR/SSIM/LPIPS results across region of interest and various BungeeNeRF scenes. 64

3.3 PSNR, LPIPS, and SSIM benchmarks on BungeeNeRF New York and San Francisco scenes. 64

3.4 Comparison of initial sparse point cloud and 3DGS densified point cloud vs. MVS densified point cloud as reference. 68

3.5 Performance Metrics for Different Sky and Sunlight Conditions 69

3.6 Point Cloud Distance Metrics Comparison Against Reference Point Cloud (Bright-Uniform). 69

4.1 Comparison of 2D Novel View Synthesis Metrics Between Gaussian Splatting Modules 93

4.2 Comparison of 3D-SSIM Scores Between our Pipeline and GS2Mesh 94

4.3 Ablation Study on 2DGS+ 95

5.1 Table of Commonly Used Deepseek and OpenAI LLMs, compiled on 2025-01-31. 102

5.2 Google Maps Platform Air Quality Pollutant List, Abbreviation, and Unit 119

5.3 Air Quality Retrieval Neighbourhoods and Coordinates 121

List of Abbreviations/Glossery

Common abbreviations:

- (3D/2D)GS: (3D/2D) Gaussian Splatting
- (D1, D2) MSE: (Point-to-Point, Point-to-Surface) Mean Squared Error
- API: Application Programming Interface
- BERT: Bidirectional Encoder Representations from Transformers
- CLIP: Contrastive Language Image Pretraining
- DINO: Self-DIstillation with NO Labels
- DLNR: Stereo Matching Network with Decouple LSTM (Long Short Term Memory) and Normalization Refinement
- GIS: Geographic Information Science/Systems
- GPT: Generative Pretrained Transformer
- GPU: Graphical Processing Unit
- GS2Mesh: Gaussian-Splatting-to-Mesh
- LLM: Large Language Model
- LPIPS: Learned Perceptual Image Patch Similarity
- LiDAR: Light Detection and Ranging (Scanner)
- MLP: Multi Layer Perceptron

- MVS: Multi-View-Stereo
- NLP: Natural Language Processing
- NeRF: Neural Radiance Field
- PSNR: Peak Signal to Noise Ratio
- RGB/rgb: Red Green Blue
- SAM: Segment Anything Model
- SH: Spherical Harmonics
- SSIM: Structural Similarity Index Measure
- SfM: Structure from Motion
- TSDF: Truncated Signed Distance Function

Abbreviations coined in this thesis:

- 2DGS+: 2D Gaussian Splatting Plus, specifically this fork of 2DGS
- DTB: Digital Twin Building
- GBM: Gaussian Building Mesh

List of Symbols

Common variables:

- x: (lowercase math script) a scalar
- $\mathbf{x}$: (boldfont) a vector
- M: (uppercase math script) a matrix
- σ: standard deviation or density (context dependent).
- μ: mean
- Σ: covariance matrix
- d: distance
- θ, ϕ: polar and azimuthal angles (physics convention)
- r: radius
- W: world to camera matrix (extrinsic matrix)
- L: a loss function
- λ: a loss function weight hyperparameter

Common symbols and operations:

- exp(.), $e^{(.)}$: exponential
- log(.): natural logarithm

- ∇: gradient
- $\odot$: element-wise multiplication
- $|\mathbf{x}|, ||\mathbf{x}||$: Norm of a vector $\mathbf{x}$, 2-norm unless specified otherwise
- $\times$: cross product (when applied to 3D vectors) or Cartesian product (when applied to sets)
- $\bigcup$: union
- $\sum$: sum
- $\prod$: product
- $\rightarrow$: a mapping or a transformation
- $\uparrow$: used to indicate a metric where higher is better
- $\downarrow$: used to indicate a metric where lower is better

Chapter 1

Introduction

Digital twin systems are digital representations of physical systems combining digital (3D) modelling with integrated data and data analytics. Urban Digital Twins are digital representations of cities that integrate multi-source real-time data and cross-domain analytics with 3D modelling. I.e. Urban Digital Twins are digital twin systems of cities focusing on sensor integration and interconnections of multiple relevant information systems (usually via the internet). Urban Digital Twins can capture complex interactions between infrastructure and human activities by leveraging the Internet of Things (IoT) connected sensor systems and Geographic Information Systems (GIS). Unlike static 3D city models, Urban Digital Twins are continuously updated with real-time data or internet-connected data streams, providing an evolving digital representation of the physical world. Predictive data analytics is used to enhance decision-making, allowing planners, policymakers, and other stakeholders to visualize, simulate, and optimize urban systems in a virtual environment. This seamless integration of sensor data and digital modelling makes Urban Digital Twins a powerful tool for tackling modern urban challenges [1, 2].

3D modelling is the first step towards the creation of Urban Digital Twins. The reconstruction and modelling of 3D structures from 2D images have gained significant attention due to recent advancements in photorealistic view synthesis methods with 3D reconstruction capabilities. This is an interdisciplinary research area that bridges computer vision, computer graphics, and photogrammetry. It has diverse applications across multiple domains related to digital twins, including autonomous navigation facilitated by 3D scene understanding [3], remote sensing, and photogrammetry for creating 3D maps crucial for navigation, urban planning, and administration [4]. This field of research has seen multiple breakthroughs in recent years.

The integration of cloud platforms and large-language model (LLM) platforms significantly enhances the efficiency and accessibility of Urban Digital Twins. Cloud platforms can provide both the data and the computational backbone for Urban Digital Twins, enabling real-time ingestion, storage, and processing of massive datasets—ranging from high-resolution 3D city scans to sensor streams and remote sensing imagery [5]. By leveraging distributed cloud architectures, cloud-based computing resources ensure cost-efficient scalability for complex simulations, such as traffic flow modelling or disaster response planning. Furthermore, cloud-based geospatial services (e.g., Google Earth Engine, Google Maps Platform, Azure Maps) enrich Urban Digital Twins with up-to-date contextual layers, such as weather patterns or land-use changes, creating dynamic, multi-source urban representations from data streams. Complementing this infrastructure, LLMs introduce advanced natural language processing (NLP) capabilities to Urban Digital Twins. LLMs automate semantic analysis of textual/image inputs to identify emerging urban issues and generate human-readable reports or policy recommendations. They also enable intuitive, conversational querying for code generation in Urban Digital Twins, allowing non-technical users to interact with complex urban models using natural language. Together, cloud plat-

forms and LLMs advance the analytical capabilities of Urban Digital Twins by integrating multi-modal data streams and automating insight generation. This in turn supports evidence-based decision-making in complex, dynamic urban environments.

1.1 Motivation

This thesis is motivated by exploring, adapting, and improving very recent developments in computer vision and natural language processing to help create the next generation of urban digital twin systems. Recent advances in 3D computer vision and natural language processing (NLP) have revolutionized how complex systems are modelled and analyzed [6, 7]. By leveraging and building on these advances, this thesis improves accuracy, efficiency, and user accessibility in digital twin systems modelling of urban environments.

Recent breakthroughs in 3D scene reconstruction, such as Neural Radiance Fields (NeRF) [8] and Gaussian Splatting [6], enable photorealistic 3D environments to be learned from 2D images, capturing accurate dense geometry and photorealistic lighting. This is unlike traditional methods reliant on LiDAR or photogrammetry typically used in established Urban Digital Twins. Concurrently, the impact of large language models (LLMs) on most aspects of information technology cannot be overstated. For Urban Digital Twins, LLMs can transform large amounts of text and image data into actionable insights and even automate code generation, improving user accessibility. The rise of cloud-based mapping services, GIS services, and computing services supports scalable, real-time geospatial data processing, lowers deployment barriers, and acts as a reliable data source stream.

Established Urban Digital Twins systems typically use classical photogrammetry methods for 3D modelling, and in general, do not leverage state-of-the-art advances in AI

systems (such as newer large language models (LLM) and LLM-based AI systems) [9]. **Although these Urban Digital Twin Systems are well-established, the aforementioned recent innovations in computer vision and natural language processing offer great opportunities for improving accuracy, efficiency, and accessibility in both the 3D modelling and the data analytics aspects** [10]. Specifically, Gaussian Splatting-based approaches can improve the accuracy and efficiency of both large and small-scale 3D geometry generation, in addition to creating photorealistic renders for downstream applications. LLM systems can generate code and perform large-scale data analysis with simple natural language instructions, greatly improving efficiency and user accessibility.

This thesis unites Gaussian Splatting, novel LLM systems, and cloud mapping platform integration in a local-scale (urban) digital twin system framework that contributes towards building a blueprint for the next generation of more accurate, efficient, and accessible Urban Digital Twins.

1.2 Research Objectives

The overarching aim is prototyping the next generation of digital twin systems for urban environments focusing on large-scale 3D reconstruction, small-scale 3D reconstruction, and AI-powered data analytics. This thesis explores and advances the application of Gaussian Splatting (GS) in large-scale 3D point cloud reconstruction and local-scale mesh extraction. It also explores the digital twin modelling capabilities of recent developments in large language models and vision language models and designs LLM systems for user accessible automated data analytics. These explorations resulted in the design of novel information systems and computer vision frameworks with demonstrable improvements over previous

methods.

The thesis research is structured into three projects, each addressing a distinct challenge or phase in the (urban) digital twin system prototyping.

The first objective was to explore the capabilities of Gaussian Splatting in large-scale urban 3D modelling (as dense 3D point clouds) to replace classical 3D photogrammetry for Urban Digital Twins, as well as to benchmark against NeRF-based view synthesis method as a potential alternative. For this objective, I developed and evaluated a large-scale 3D point cloud extraction and novel view synthesis method using Google Earth Studio imagery and Gaussian Splatting methods, and was one of the earliest to use Gaussian Splatting in a large-scale remote sensing context. The tested Gaussian Splatting methods were orders of magnitude faster than similar NeRF methods, and significantly more photorealistic. The resulting 3D point cloud densification was also more geometrically accurate.

Focusing on the scale of individual buildings, the second objective was to create an efficient and precise Gaussian Splatting-based 3D mesh extraction framework. For this objective, I created the Gaussian Building Mesh (GBM) pipeline, which leverages an enhanced version of 2DGS for 3D mesh extraction. This includes refining segmentation and masking techniques using state-of-the-art deep learning models to improve the quality and accuracy of Gaussian Splatting-based reconstructions. Compared to the then state-of-the-art Gaussian Splatting-based mesh extraction framework, my method produced cleaner 3D mesh, is more robust to difficult scenes, and has faster training speed.

The third objective was to integrate Gaussian Splatting-based 3D models with cloud mapping platform-based Geographic Information Systems (GIS)

and large language models (LLMs) to enable intelligent and accessible digital twin systems. This involves developing a framework for retrieving, visualizing, and analyzing multi-modal geospatial data using a newly designed multi-agent LLM system with code automation and intelligent message passing for data analysis.

By addressing these objectives, this thesis contributes to the advancement of AI systems for real-world geospatial applications, especially contributing to prototyping the next generation of Urban Digital Twin.

1.3 Contributions

The research in this thesis is divided into three separate projects, each presented in their respective chapters, which make up the body of this thesis. Part of the background theory and literature review contributes to my survey manuscript (or more precisely, is constituted from): "NeRF: Neural Radiance Field in 3D Vision, Introduction and Review" and its additional chapters on Gaussian Splatting.

Research 1: "Enhanced 3D Urban Scene Reconstruction and Point Cloud Densification using Gaussian Splatting and Google Earth Imagery" presents a large-scale 3DGS-based point cloud extraction method and is the first work to propose using Google Earth Studio as a data source for remote sensing-based Gaussian Splatting training. It is also one of the earliest works on large-scale remote sensing-based Gaussian Splatting. This project served as my exploration of remote sensing-based Gaussian Splatting and its performance under different scales and conditions for remote sensing urban scenes.

Research 2: "Gaussian Building Mesh (GBM): Extracting a Building's 3D Mesh with Google Earth and Gaussian Splatting" is a pipeline that allows for fast and accurate extrac-

tion of a building's 3D mesh from its name, address, postal code, or geographic coordinates. The proposed GBM uses 2DGS+, an unpublished version of 2DGS that incorporates many state-of-the-art improvements to Gaussian Splatting. Key contributions include the creation of the pipeline, which uses SAM2 with GroundingDINO for building mask extraction; improvements to the masking module, which enable the training of Gaussian Splatting on masked images; and the development of a mask refinement procedure that reduces noise and improves mask geometry. This resulted in a pipeline that is more robust, accurate, and faster than the previous state-of-the-art.

Research 3: "Digital Twin Buildings: 3D Modelling, GIS Integration, and Visual Descriptions Using Gaussian Splatting, ChatGPT/Deepseek, and Google Maps Platforms," presents a framework named Digital Twin Buildings as a prototype digital twin system at the individual building scale, although it can easily be extended to city-scale. Starting from a building's address, coordinates, place name, or postal code, the framework uses GBM, the pipeline presented in Chapter 4, to extract its 3D model. The 3D building model is then integrated with Google Maps Platforms and a newly designed multi-agent large language model (LLM) system via Python API bindings. The LLM-based data analytics are demonstrated in experiments involving the analysis of multi-view, multi-scale images extracted from Map Platform API calls and processed using a multi-agent LLM system. The LLM system is also capable of code generation and execution from natural language-based user prompts, greatly improving user-accessibility.

An additional **case study** on a practical application of Digital Twin Building's with cloud-mapping integration and multi-LLM data analytics + code generation-execution is included in Chapter 5.5. These experiments were not part of the original manuscript for Paper 3, but contributed towards an independent publication. The Digital Twin Building framework was used to automatically retrieve and analyze air quality data from the Los

Angeles California Pacific Palisades area, from January-09-2025 to February-09-2025, a time period in which the neighbourhood was devastated by wildfires. This data is compared against two baselines over the same time period and analyzed to benchmark the policy recommendation capabilities of the LLM system.

1.4 Outline of the Thesis

The body of the thesis consists of the three aforementioned research projects, each contributing to a preprint manuscript or peer-reviewed paper. The first two projects are presented in Chapters 4 and 5, focusing on 3D reconstruction and novel view synthesis. The third project, detailed in Chapter 6, explores cloud mapping service integration and LLM-based data analysis.

Chapter 1 introduces and motivates the research in this thesis.

Chapter 2 provides the background knowledge and literature review for this thesis.

Chapter 3 presents early experiments on urban 3D Gaussian Splatting which resulted in the peer-reviewed paper "Enhanced 3D Urban Scene Reconstruction and Point Cloud Densification using Gaussian Splatting and Google Earth Imagery" (**Research 1**).

Chapter 4 presents a novel 3D building mesh extraction pipeline and related experiments which resulted in the manuscript "Gaussian Building Mesh (GBM): Extract a Building's 3D Mesh with Google Earth and Gaussian Splatting" (**Research 2**).

Chapter 5 presents Digital Twin Building, a novel framework building modelling, and related experiments, which resulted in the manuscript "Digital Twin Buildings: 3D Modelling, GIS Integration, and Visual Descriptions Using Gaussian Splatting, ChatGPT/Deepseek, and Google Maps Platforms" (**Research 3**). An additional set of experiments

on the analysis of air quality data using the Digital Twin Building framework is appended as Section 5.5 of Chapter 5 (**Case Study**).

Chapter 6 contains additional discussions, and suggestions for future research directions, and concludes this thesis.

Chapter 2

Background and Related Works

This chapter presents the background and the literature review, with a large focus on Gaussian Splatting. The Gaussian Splatting section (Section 2.4) is meant to be read as a reference and is more mathematically detailed than most introductions to the subject. The section also assumes knowledge of homogeneous coordinates and camera projection. The other sections can be read as short introductions to the background knowledge and a review of the literature.

2.1 Relevant 3D Representations

This section provides a brief introduction to the 3D representations relevant to this thesis including point clouds, voxels, 3D meshes, radiance fields, and signed distance functions.

2.1.1 Explicit Representations

Point Clouds

Point clouds are explicit 3D representations in the form of an unordered list of points. Each point is an n-tuple $(x, y, z, ...)$, with (x, y, z) denoting spatial coordinates. The additional point cloud features can include point colour (r, g, b), local normal (n_x, n_y, n_z), and/or feature vector $\vec{f}$ which can greatly vary depending on the data source and application.

Point clouds are permutation-invariant representations (more specifically, point cloud processing algorithms *should be* permutation invariant), as they are an unordered list of 3D points and point features. Depending on the source, a point cloud of a scene can often exhibit density variation due to the nature of its acquisition during scanning or photogrammetric reconstruction. Point clouds are often products of 3D scanning via LiDAR(s) or structured light scanners, or they are generated using photogrammetry, simultaneous localization and mapping (SLAM), or structure from motion (SfM).

Voxels

Voxels (volumetric pixels) are the three-dimensional analogue of 2D image pixels. In voxel representations, numerical values (e.g., colours, density or occupancy, feature vectors) are stored on a discrete three-dimensional array. Voxels are fixed-resolution representations: once the voxelization resolution is chosen, the space is subdivided into voxels of identical volume. The voxel 3D-indexing location corresponds to the data's spatial location. Many 2D pixel-based computer vision algorithms generalize well to 3D voxels since images are 2D voxel equivalent representations. However, the curse of dimensionality in three dimensions imposes a much harsher trade-off between resolution and memory/storage requirements in

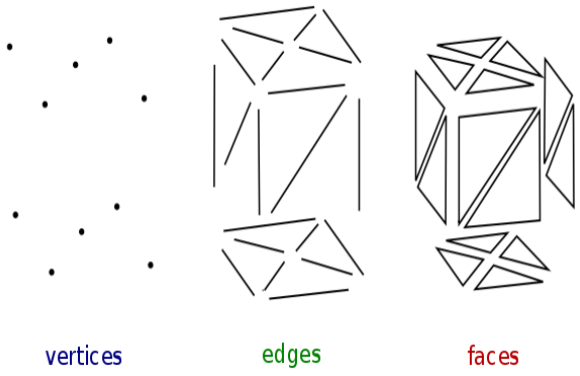


Figure 2.1: Diagram of the constituents of a 3D mesh, adapted from [11].

three dimensions than in two dimensions.

Meshes

3D meshes are explicit representations of scene geometry as 3D surfaces. These surfaces are stored as a collection of vertices, edges, and faces that define the surface of a polyhedral representation of an object or a scene (see Figure 2.1). The faces are typically triangular, but quadrilateral meshes and other polygonal meshes are also commonly used, replacing triangles with other polygons. Meshes can also be coloured or textured using colour and texture maps, which can enable photorealistic rendering of the 3D object. Because algorithms for mesh generation, mesh intersection, collision detection, rendering, and physics-based rigid and non-rigid body simulations are well understood, 3D meshes are widely used in computer graphics, 3D modelling, numerical simulations and scientific numerical modelling.

2.1.2 Implicit Representations

Implicit representations are mathematical functions which represent 3D structures, typically as field functions $f : R^3 \rightarrow V$ mapping 3D spatial coordinates in R^3 to some set of values V. In theory, implicit representations are resolution-independent, as the function can be queried at any level of discreteness, allowing for the capture of details at any scale. Implicit representations are often used in computer graphics and 3D animations/rendering. Most relevant to this thesis, implicit representations are fundamental to Neural Radiance Fields. In Neural Radiance Fields, multi-layer perceptrons (MLPs) are used to learn and store the 3D representation field functions.

Density and Radiance Fields

A density field is a function

$$\sigma : R^3 \rightarrow [0, 1] \tag{2.1}$$

mapping 3D coordinates (x, y, z) to a volume density value in the interval [0,1]. The scene can be represented by a single density field describing the geometry of the scene. Alternatively, multiple density fields can be used by assigning a density field to each object or object-class of interest in the scene.

A radiance field is a function

$$F : R^3 \times S^2 \rightarrow R^4 \tag{2.2}$$

mapping 3D coordinates and camera view direction (x, y, z, θ, ϕ) to local volume density value and local view-dependent colour value (σ, r, g, b). These radiance fields can be used to generate photorealistic images of the scene via volume rendering.

Neural Radiance Field (NeRF) methods typically use simple neural networks to learn

the scene's radiance field(s). These neural networks are typically multi-layer perceptrons (MLPs) with a directionless density branch and a directional colour branch.

Signed Distance Functions

Signed Distance Functions given by,

$$f_{sdf} : R^3 \rightarrow R \tag{2.3}$$

map a 3D coordinate (x, y, z) to the signed distance d to the nearest surface. d is the positive distance if the point (x, y, z) lies outside the oriented surface, and d is the negative distance if the point lies inside. Surfaces are defined at $f_{sdf} = 0$. As such, root-finding algorithms can be used to find surfaces defined by signed distance functions. Similar to radiance fields and density fields, both scene-based or object-based signed distance functions can be used. Certain NeRF methods replace neural density fields with neural signed distance functions to better learn and represent 3D surface geometry. Signed distance functions are also fundamental to the Truncated Signed Distance Function integration meshing algorithm.

2.1.3 Truncated Signed Distance Function Integration

The Truncated Signed Distance Function integration (TSDF) algorithm [12] produces a 3D mesh from a depth maps-based or signed distance function-based representation. This method is extensively used to mesh Gaussian Splatting models which can rasterize depth maps in parallel with RGB images. In short, with TSDF, a fixed-resolution voxel grid is first defined and then populated with truncated signed distance values. Then a mesh is

generated on a per-voxel basis.

In more detail, a truncation threshold distance δ is defined by the user. Starting with 2D depth maps, at each voxel, for each depth image, the depth values are projected onto the voxel as a truncated signed distance value $d_i = z_{voxel_i} - z_{observed}|_{voxel_i}$ truncated at amplitude δ. This process is illustrated in Fig. 2.2. The voxel's truncated SDF value is updated as a weighted average for each new depth map. This is given by

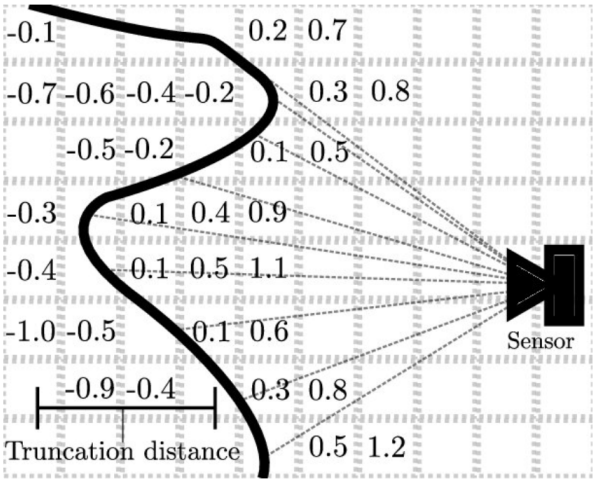


Figure 2.2: Diagram of TSDF depth projection, sourced from [13]. The zero-curve (zero-surface in 3D) of the signed distance function used to represent geometry is drawn as a bold line.

$$\text{TSDF}(\mathbf{v}) = \frac{\sum_i w_i \cdot \text{trunc}(d_i)}{\sum_i w_i}, \quad \text{trunc}(d) = \begin{cases} \delta & \text{if } d > \delta, \\ -\delta & \text{if } d < -\delta, \\ d & \text{otherwise} \end{cases} \tag{2.4}$$

where the depth map weights w_i depend on the implementation (e.g. uniform weights for Gaussian Splatting vs. moving averages for RGB-D camera). Alternatively, if available,

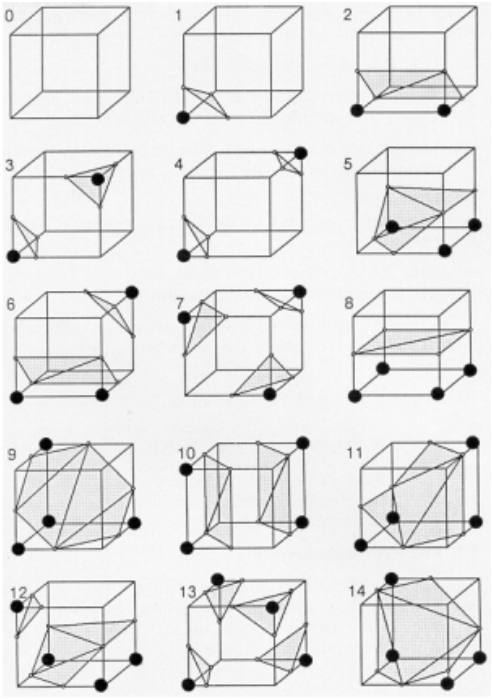


Figure 2.3: Per-voxel triangle mesh based on vertex configuration, sourced from [14].

an SDF-based 3D representation can be evaluated and truncated at the voxel locations to populate the voxel.

Once the voxel is populated with TSDF values, the marching cubes [14] algorithm is used to extract a 3D mesh. For the marching cubes algorithm, each TSDF voxel is flagged as "inside" ($TSDF$ value ≤ 0) or "outside" ($TSDF$ value > 0) the surface we wish to extract. Then, a new voxel grid is defined using the TSDF voxels as vertex values. The binary "inside/outside of surface" checks at the eight voxel corners define 256 possible configurations (2^8 configurations, canonically reduced to 15 due to symmetries). The voxel grid is populated with mesh triangles based on corner configurations using a look-up table. This is shown in Fig. 2.3. By meshing every cube in the voxel grid, the output is a 3D mesh representation of the surface defined by the TSDF step.

2.2 Photogrammetry and Structure from Motion Preprocessing

The standard implementations of Gaussian Splatting and NeRF rely on COLMAP [15] for preprocessing 2D images to extract image poses and/or initial 3D points [6, 8]. This Structure from Motion (SfM) preprocessing takes a collection of unordered images with unknown camera poses, and outputs the camera pose of each image, as well as a sparse point cloud. In remote sensing, COLMAP would be called a photogrammetry method and is conceptually similar to many commonly used photogrammetry libraries and software packages. Like most SfM methods, COLMAP SfM consists of the following steps.

Feature Extraction: In this step, for each image, key points are identified, and robust camera viewpoint-invariant local features are assigned to these key points. Image pairs with potentially overlapping points are then identified.

Geometric Verification: A scene graph is constructed, with images as nodes and edges connecting overlapping images. This is done by verifying potentially overlapping image pairs by estimating a valid homography using a robust estimation technique, such as a variant of Random Sample Consensus (RANSAC) by [16].

Image Registration: Starting from an initial image pair whose key points are triangulated into 3D, new images with overlap (as determined by the scene graph) are added to the scene. This is achieved by solving the Perspective-n-Point problem [16], which estimates camera poses given a set of 3D points and their 2D projections. This step robustly estimates the pose of the newly registered image.

Triangulation: Given key points viewed from two images with known poses, the key points are triangulated [17] into 3D. Newly registered images extend the scene by enabling

additional key points to be triangulated into the 3D reconstruction.

Error Correction: To correct errors in registration and triangulation, bundle adjustment [18] is performed. This involves jointly optimizing the camera poses and 3D points by minimizing the squared error of the reprojection of the 3D points onto the image planes.

COLMAP [15] introduced various innovations, including improvements to geometric verification, enhanced robustness in initialization and triangulation, a next-best-view selection method, and an iterative, more efficient bundle adjustment method. These advancements resulted in the COLMAP SfM library.

2.3 Novel View Synthesis

Novel view synthesis is a field in computer vision with the objective of rendering 2D images of a scene under arbitrary poses given source images with specific poses.

2.3.1 Rendering

Rendering is the process of creating a 2D image from a 3D representation. In the context of this thesis, we will cover two families of rendering techniques, rasterization vs. volume rendering.

Both NeRF and Gaussian Splatting methods are fundamentally based on two core concepts: differentiable rendering/rasterization and learning-based 3D representation. Differentiable rendering and rasterization enable the computation of gradients during the creation of 2D images from a 3D representation. This, in turn, allows for the learning and refinement of the 3D representation through loss function optimization, similar to neural network training.

2.3.2 Differentiable Volume Rendering and NeRF

(Direct) Volume rendering is also known as volumetric ray tracing and volumetric ray casting. For a to-be-rendered image with a camera pose, a ray is traced through each pixel and through the 3D scene. Using a simple camera model, such as the pinhole model, the ray is traced from the camera's centre of projection and travels through a pixel on the image plane of the to-be-rendered image, which is positioned between the camera and the volume being rendered. As the ray travels through the scene, local density and colour are accumulated along the ray and α-blended to create the pixel's colour. This process is repeated for each pixel of the to-be-rendered image. [19]

This is given by

$$C(\mathbf{r}) = \int_{t_1}^{t_2} T(t) \cdot \sigma(\mathbf{r}(t)) \cdot \mathbf{c}(\mathbf{r}(t), \mathbf{d}) \cdot dt, \tag{2.5}$$

where $\sigma(\mathbf{r}(t))$ and $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ represent the volume density and colour at point $\mathbf{r}(t)$ along the camera ray $\mathbf{r}$ with viewing direction $\mathbf{d}$, and dt represents the differential parametrized distance element travelled by the ray at each integration step. $T(t)$ represents the accumulated transmittance, which is the probability that the ray travels from t_1 to t without any obstruction. It is expressed as:

$$T(t) = \exp(-\int_{t_1}^{t} \sigma(\mathbf{r}(u)) \cdot du). \tag{2.6}$$

Eqs. (2.5) and (2.6) can be discretized in various ways.

In Neural Radiance Fields [19], a typical approach is to use non-deterministic stratified sampling. This is done by sub-dividing the ray into N equally spaced bins, and randomly drawing a sample from each bin. Then, Eq. (2.5) can be approximated as

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i T_i \mathbf{c}_i, \text{ where } T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j) \tag{2.7}$$

where δ_i is the distance from sample i to sample $i+1$. $(\sigma_i, \mathbf{c_i})$ are the density and colour evaluated along the sample point i given the ray. α_i the transparency/opacity from alpha compositing at sample point i, is given by

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i). \tag{2.8}$$

In NeRF, this process is differentiable, ensuring gradient flow from the optimization objective to the radiance fields stored inside multi-layer perceptrons. This allows for the learning of radiance field-based 3D representation of the scene using gradient descent-based optimization.

2.3.3 Differentiable Rasterization and Gaussian Splatting

(3D) Rasterization is the process of directly mapping 3D shapes and geometric primitives to 2D images (as opposed to tracing a ray from the image into the 3D scene). The mapping is typically a fixed function within the graphical pipeline, but usually involves a 3D to 2D projection step and a colouring step. In the context of this thesis, we are interested in the Gaussian Splatting tile-based rasterization, which is explained in more detail in Section 2.4.2.

2.4 3D Gaussian Splatting

2.4.1 Gaussian Distributions

A 3D Gaussian distribution is an extension of the 1D Gaussian distribution to three dimensions. It describes a probability distribution over three-dimensional space. Analogous to a 1D Gaussian distribution being characterized by a mean μ and variance σ^2, a 3D Gaussian distribution is characterized by a positional mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$.

A random vector $\mathbf{x} = (x_1, x_2, x_3)^T$ follows a 3D Gaussian distribution if it has the probability density function:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{3/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.9}$$

where:

- $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)^T$ is the **mean vector** in spatial position.
- $\boldsymbol{\Sigma}$ is the **covariance matrix**:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \mathrm{Cov}(x_1, x_2) & \mathrm{Cov}(x_1, x_3) \\ \mathrm{Cov}(x_2, x_1) & \sigma_2^2 & \mathrm{Cov}(x_2, x_3) \\ \mathrm{Cov}(x_3, x_1) & \mathrm{Cov}(x_3, x_2) & \sigma_3^2 \end{bmatrix}. \tag{2.10}$$

2.4.2 Gaussian Splatting

Gaussian Splatting is used as the foundation for 3D reconstruction and view-synthesis in this thesis. 3D Gaussian Splatting (3DGS) [20], the original Gaussian Splatting method, is explained in this section. The pipeline is illustrated in Figure 2.4

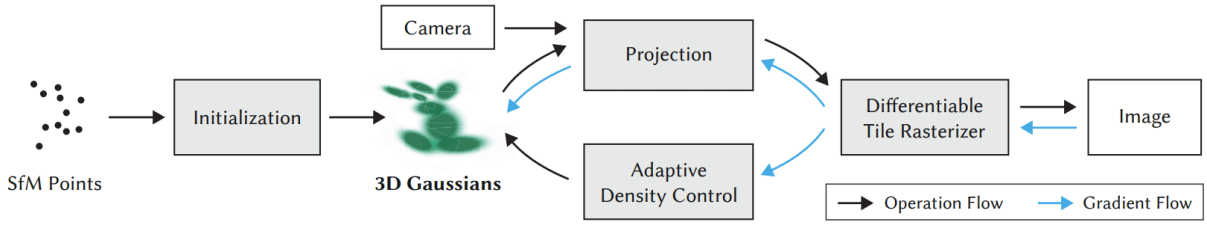


Figure 2.4: Diagram of the 3DGS framework and training process, forward flow and gradient flow, sourced from [20].

From 2D images of a scene, 3D Gaussian Splatting learns and represents the scene geometry using unnormalized 3D Gaussian functions as geometric primitives. These are sometimes called Gaussian Splats. Each Gaussian is characterized by its mean $\boldsymbol{\mu} \in R^3$ and 3×3 covariance matrix Σ, which are learnable parameters.

$$G(\mathbf{x}) = \left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right). \tag{2.11}$$

Learnable lighting parameters are also assigned to each Gaussian. The scene lighting and colour are learned as third-degree spherical harmonics coefficients for each colour channel attached to each Gaussian (48 parameters). Additionally, each Gaussian is assigned a local (conic) opacity σ (1 parameter). Combined with the 3D mean/location (3 parameters) and covariance matrix (3+4 parameters), this results in a total of 59 trainable parameters per Gaussian. The 3D covariance matrix Σ is learned as a 3D scaling vector $\mathbf{S}$, and a rotation represented by a quaternion $H = (r, i, j, k)$, which can be then used to reconstruct a 3D rotation matrix R as follows.

$$R = \begin{bmatrix} 1-2(j^2+k^2) & 2(ij-kr) & 2(ik+jr) \\ 2(ij+kr) & 1-2(i^2+k^2) & 2(jk-ir) \\ 2(ik-jr) & 2(jk+ir) & 1-2(i^2+j^2) \end{bmatrix}. \tag{2.12}$$

The 3D covariance matrix is then given by

$$\Sigma = RSS^TR^T \tag{2.13}$$

where S is the diagonal matrix with $\mathbf{S}$ on the diagonals.

A sparse initial point cloud and training image camera poses are first computed using a structure from motion library such as COLMAP [15]. A Gaussian is initialized at each point in the sparse point cloud at the beginning of training. The Gaussian Splatting model is then trained using the differentiable tile-based rasterizer.

Rasterization

The tile-based rasterizer splits the image into 16×16 tiles. For each tile, a view frustum is projected into the 3D scene. 3D Gaussians are accumulated/assigned per tile according to their overlap with the view frustum, and are projected into 2D by mapping their covariance matrices Σ and means.

This mapping is essentially a rotation and translation mapping from world-space coordinates to camera space coordinates and a projection mapping from camera space coordinates to image coordinates (ray-space coordinates). Starting in homogenous coordinates in world-space, this is given by

$$\Sigma' = JW\Sigma W^TJ^T \tag{2.14}$$

where W is the view transformation and J is the Jacobian of the affine approximation of the projective transformation. In homogenous coordinates, the view transformation matrix W is given as a translation vector appended to a rotation matrix.

$$W = [R|\mathbf{t}] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_x \\ R_{21} & R_{22} & R_{23} & T_y \\ R_{31} & R_{32} & R_{33} & T_z \end{bmatrix}. \tag{2.15}$$

The $W \in R^{4\times3}$ matrix transforms a 3D point from world-space coordinates (as homogenous coordinates) to a 3D point in camera space coordinates (as Cartesian coordinates).

$$\begin{bmatrix} \mu_x^c \\ \mu_y^c \\ \mu_z^c \end{bmatrix} = W \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \\ 1 \end{bmatrix}. \tag{2.16}$$

Although the original EWA splatting paper [21] defines the equations for generalized camera projections, the standard Gaussian Splatting only implements the model for pinhole camera models. For a pinhole camera with focal lengths f_x, f_y, J is given by

$$J = \begin{bmatrix} \frac{f_x}{\mu_z^c} & 0 & -\frac{f_x \cdot \mu_x^c}{(\mu_z^c)^2} \\ 0 & \frac{f_y}{\mu_z^c} & -\frac{f_y \cdot \mu_y^c}{(\mu_z^c)^2} \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.17}$$

The third row and third column of Σ' are then dropped to obtain the 2D covariance matrix in the imaging plane. The 2D mean is simply the 3D mean projected onto the imaging plane using the full camera projection. For a pinhole camera with intrinsic matrix P, this

is given by

$$\mu_{2D} = PW\mu_{3D}. \tag{2.18}$$

As a final coordinate transformation, the screen-space is normalized to height and width 1, into the Normalized Device Coordinates (NDC) via simple rescaling.

Gaussians are then sorted according to tile and by depth. For each pixel in a tile, the pixel's colour is generated via alpha blending accumulating in-scene direction-dependent colour using the learned spherical harmonics coefficients (Section 2.4.2). This is given by

$$\hat{C}(u,v) = \sum_{i=1}^{N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \tag{2.19}$$

where $\mathbf{c}_i$ denotes the ith Gaussian's directionally dependent colour and α_i denotes its alpha value. The alpha-value α_i at a pixel location (u,v) is given by evaluating the 2D Gaussian, scaled by its associated learned opacity o_i scaled by the sigmoid function σ.

$$\alpha_i(u,v) = \sigma(o_i)G_i^{2D}(u,v) \tag{2.20}$$

where $G^{2D}(x)$ is the Gaussian (2.11) projected into 2D and onto the image plane via (2.14) and rescaled to NDC. Although the standard Gaussian Splatting method does not use ray-tracing but instead uses a projection-based rasterization method, we note the similarity of the Gaussian Splatting alpha-blending equation given by Eq. (2.19) to the NeRF volume rendering equation given by Eq. (2.7).

The rasterizer generates an image which is compared to the ground truth images using a photometric L_1 loss and L_{D-SSIM}, a difference of Structural Similarity Index Measure (D-SSIM) [22] loss via

$$L = (1-\lambda)L_1 + \lambda L_{D-SSIM} \tag{2.21}$$

with λ being an adjustable weighing parameter defaulting to 0.2.

This entire rasterization process is differentiable, ensuring gradient flow from the training objectives to the trainable parameters. The 59 trainable parameters are optimized via back-propagation and a gradient descent algorithm variants such as [23].

Densification and Pruning

3D Gaussian Splatting also densifies/grows new Gaussians in regions with high view-space positional gradient (threshold $\tau pos > 2.0 \times 10^{-4}$ as default). These regions correspond to neighbourhoods with missing geometric features and regions with a few Gaussians covering large areas of the scene.

In more detail, suppose a m pixels are covered by the i-th Gaussian. The standard densification criterion in Gaussian Splatting is based on a threshold of the magnitude of the per-Gaussian view-space positional gradient $\nabla_{\mu_{i_{2D}}} L = (\frac{\partial}{\partial u_i}, \frac{\partial}{\partial v_i})L$, summed over all m pixels covered by the Gaussian. This is given by

$$|g_i| = |\sum_{j}^{m} (\frac{\partial L_j}{\partial u_i}, \frac{\partial L_j}{\partial v_i})|. \tag{2.22}$$

Low-variance Gaussians with view-space position gradients are duplicated. On the other hand, high variance Gaussians are split into two with standard deviation divided by a factor of 1.6. This process is illustrated in Figure 2.5. Unimportant Gaussians are also pruned. Gaussians that are essentially transparent with opacity less than some user-defined threshold ($\alpha < \epsilon_a$, default value at $5e-3$) are deleted. Every 3000 iterations (or some other number of the user's choosing), every Gaussian's opacity is set to zero, then allowed to be re-optimized, and then culled where needed. This process controls the number of

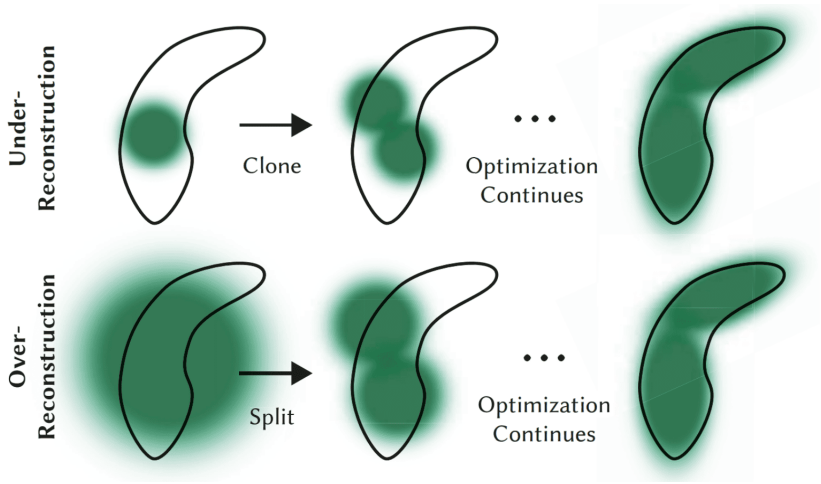


Figure 2.5: Diagram of 3DGS densification, sourced from [20]. Both types of regions are identified by thresholding per-Gaussian view-space positional gradients. An over-reconstruction Gaussian has large covariances, an under-reconstruction Gaussian has small covariances.

floater artifacts, and helps control the total number of Gaussians. This densification and density control process can allow for point cloud reconstruction of similar density and quality compared to a dense SfM/SLAM reconstruction.

RGB Spherical Harmonics

If we wish to add a directional dependence to a local colour, we might express this as 3-channel plenoptic colour field/light field at some fixed spatial position $\mathbf{x_0} = (x_0, y_0, z_0)$

$$\mathbf{C}(x = x_0, y = y_0, z = z_0, \phi, \theta) = (r_{\mathbf{x_0}}(\phi, \theta), g_{\mathbf{x_0}}(\phi, \theta), b_{\mathbf{x_0}}(\phi, \theta)), \tag{2.23}$$

where $r_{\mathbf{x_0}}, g_{\mathbf{x_0}}, b_{\mathbf{x_0}}$ are real-valued functions for each of the respective colour channels, and (θ, ϕ) are the polar and azimuthal viewing angles. The domain of these directional r, g, b functions is the surface of a sphere, denoted by S^2, parametrized by the two angles. The use

of these directionally dependent colour functions instead of constant RGB colour values is one of the reasons why images synthesized using Gaussian Splatting are much more photorealistic than images rendered from a coloured point cloud.

The spherical harmonics form an orthonormal basis for L_2 functions $f : S^2 \to C$ on the surface of a sphere. This is analogous to the Fourier series representation of periodic functions on an interval. This means that any such functions $f : S^2 \to C$, such as the aforementioned $r_{\mathbf{x_0}}, g_{\mathbf{x_0}}, b_{\mathbf{x_0}}$ can be expressed as an infinite series

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} k_l^m Y_l^m(\theta, \phi). \tag{2.24}$$

k_l^m is the spherical harmonic coefficient of degree l and order m, and is given by

$$k_l^m = \int_{S^2} f(\theta, \phi) Y_l^m(\theta, \phi) dS \tag{2.25}$$

where dS is the differentiable surface element, given by $dS = r \sin\theta d\theta d\phi$ on a sphere. With the appropriate coefficients, any colour functions can be represented. These coefficients (up to degree 3 typically) are the parameters which are learned during Gaussian Splatting training to store the local directionally dependent red, green, and blue colour intensity at the location of each Gaussian primitive.

Y_l^m is the complex spherical harmonics function of degree l and order m. Y_l^m is defined in terms of the associated Legendre polynomial P_l^m and is given by

$$Y_l^m(\theta, \phi) = \sqrt{\frac{(2l+1)}{4\pi} \frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi}, \tag{2.26}$$

The real valued spherical harmonics functions Y_{lm} which are used to represent actual colours

are defined in terms of the complex valued spherical harmonics functions as

$$Y_{lm} = \begin{cases} \sqrt{2}\ \mathrm{Im}[Y_l^{|m|}], & \text{if } m < 0, \\ Y_l^0, & \text{if } m = 0, \\ \sqrt{2}\ \mathrm{Re}[Y_l^m], & \text{if } m > 0. \end{cases} \tag{2.27}$$

Standard Gaussian Splatting approaches use 3 channels of spherical harmonics functions up to degree $l = 3$, and use the Plenoxels [24] implementation of spherical harmonics colours with the Cartesian coordinate conversion by expressing (θ, ϕ) as a directional unit vector with $\hat{\mathbf{d}} = (x_d, y_d, z_d)$ with $r^2 = x_d^2 + y_d^2 + z_d^2 = 1$. This coordinate conversion can be achieved by using Euler's formula, trigonometric formulae, as well as the spherical-Cartesian coordinate transformations.

$$\begin{aligned} x &= r \sin\theta \cos\phi, \\ y &= r \sin\theta \sin\phi, \\ z &= r \cos\theta, \\ e^{i\phi} &= \cos\phi + i \sin\phi = \frac{x + iy}{\sqrt{x^2 + y^2}}. \end{aligned} \tag{2.28}$$

The spherical harmonic functions up to degree 3 are listed in Table 2.1. For ease of notation, we drop the subscript denoting unit vector component ($x_d \to x$). Shown in the table is the Gaussian Splatting implementation, which is based on [25], with the following simplifications: $r \to 1$, and the following coordinate transformations: $x \to -x$, $y \to -y$.

Table 2.1: Real valued spherical harmonic functions Ylm on the unit sphere ($r = 1$) in Cartesian coordinates up to degree 3

	$l = 0$	$l = 1$	$l = 2$	$l = 3$
$m = -3$				$-\frac{1}{4}\sqrt{\frac{35}{2\pi}}(3x^2 - y^2)y$
$m = -2$			$\frac{1}{2}\sqrt{\frac{15}{\pi}}xy$	$\frac{1}{2}\sqrt{\frac{105}{\pi}}xyz$
$m = -1$		$-\frac{1}{2}\sqrt{\frac{3}{\pi}}y$	$-\frac{1}{2}\sqrt{\frac{15}{\pi}}yz$	$-\frac{1}{4}\sqrt{\frac{21}{2\pi}}y(4z^2 - x^2 - y^2)$
$m = 0$	$\frac{1}{2}\sqrt{\frac{1}{\pi}}$	$\frac{1}{2}\sqrt{\frac{3}{\pi}}z$	$\frac{1}{4}\sqrt{\frac{5}{\pi}}(2z^2 - x^2 - y^2)$	$\frac{1}{4}\sqrt{\frac{7}{\pi}}(2z^2 - 3x^2 - 3y^2)z$
$m = 1$		$-\frac{1}{2}\sqrt{\frac{3}{\pi}}x$	$-\frac{1}{2}\sqrt{\frac{15}{\pi}}xz$	$-\frac{1}{4}\sqrt{\frac{21}{\pi}}(4z^2 - x^2 - y^2)x$
$m = 2$			$\frac{1}{4}\sqrt{\frac{15}{\pi}}(x^2 - y^2)$	$\frac{1}{4}\sqrt{\frac{105}{\pi}}(x^2 - y^2)z$
$m = 3$				$-\frac{1}{4}\sqrt{\frac{35}{2\pi}}(x^2 - 3y^2)x$

2.4.3 2D Gaussian Splatting

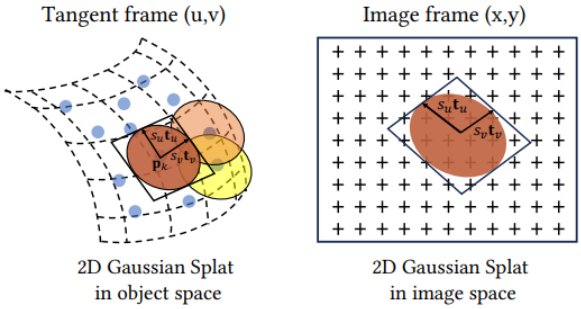


Figure 2.6: Diagram of 2D Gaussian Splats, sourced from [26]. 2D Gaussian Splatting uses surfaced aligned 2D Gaussian primitives embedded in 3D to represent the 3D scene. A 2D Gaussian is represented by it's 3D position $\mathbf{p_k}$, it's scale s_u, s_v, and it's orientation $\mathbf{t_u}, \mathbf{t_v}$.

2D Gaussian Splatting (2DGS) [26] enhances the standard 3D Gaussian Splatting (3DGS) method, improving the quality of 3D surface geometry. The key innovation in 2D Gaussian Splatting is the use of 2D planar Gaussians instead of 3D Gaussians for

scene representation. Like standard 3DGS, 2DGS employs Gaussian primitives that store spherical harmonic coefficients for each colour channel, local transparency α, and 3D location/positional mean $\mathbf{p}_k$. However, unlike 3D Gaussian primitives, 2D Gaussian primitives have two scalar values to represent variance (s_u, s_v) and two tangent vectors $(\mathbf{t}_u, \mathbf{t}_v)$ whose cross product results in the normal vector that defines orientation (see figure 2.6). Depth maps can be accurately rendered using the projected depth value. The 2D Gaussian is evaluated on the local (u,v) tangent plane

$$G(u, v) = \exp\left(-\frac{1}{2}(u^2 + v^2)\right). \tag{2.29}$$

The (u, v) plane in (3+1)D world coordinates is parametrized by

$$P(u, v) = \mathbf{p_k} + s_u\mathbf{t_u}u + s_v\mathbf{t_v}v = H(u, v, 1, 1)^T. \tag{2.30}$$

$$H = \begin{bmatrix} s_u\mathbf{t_u} & s_v\mathbf{t_v} & 0 & \mathbf{p_k} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.31}$$

The 4×4 H matrix contains the geometric information of the Gaussian and takes the role of the covariance matrix. The tangent-space to image-ray mapping is given by

$$\mathbf{x} = (xz, yz, z, z,)^T = WP(u, v) = WH(u, v, 1, 1)^T \tag{2.32}$$

where W is the combined 4×4 matrix of world to camera and camera to screen.

Importantly, instead of the tile-frustrum-based rasterization method, a ray-splat intersection is used for rasterization, making 2DGS slightly more similar to NeRF than the standard 3DGS. To compute ray-splat intersections, a pixel ray at image coordinate

$\mathbf{x} = (x, y)$ is defined as the intersection of two orthogonal planes in projective space:

$$\text{x-fixing plane:} \quad \mathbf{h}_x = (-1, 0, 0, x)^T \tag{2.33}$$

$$\text{y-fixing plane:} \quad \mathbf{h}_y = (0, -1, 0, y)^T \tag{2.34}$$

The (x, y) pixel's x-fixing plane is the plane that fixes the x-coordinates of a pixel in image space. It has a normal vector $(-1, 0, 0)^T$ (parallel to the X axis) and is offset by x. The plane itself is parallel to the $Y - Z$ plane (hence, the plane is defined by a fixed $X = x$ and free $Y - Z$ coordinates). The plane equation given a normal vector $\mathbf{n}$ is given by

$$\mathbf{n} \cdot (X, Y, Z, d)^T = 0. \tag{2.35}$$

For the x-fixing plane, this is given by

$$-1X + 0Y + 0Z + x = 0 \implies X = x. \tag{2.36}$$

This results in the homogenous coordinate representation of $\mathbf{h}_x = (-1, 0, 0, x)^T$. $\mathbf{h}_y$ follows similarly. These are transformed into the local (tangent frame) uv-coordinate system of the 2D Gaussian using the transpose of the world-to-image matrix $(WH)^T$:

$$\mathbf{h}_u = (WH)^T \mathbf{h}_x \tag{2.37}$$

$$\mathbf{h}_v = (WH)^T \mathbf{h}_y \tag{2.38}$$

The intersection point in local coordinates (u, v) lies on both transformed planes, resulting

in the following constraints:

$$\mathbf{h}_u \cdot (u, v, 1, 1)^T = 0 \tag{2.39}$$

$$\mathbf{h}_v \cdot (u, v, 1, 1)^T = 0 \tag{2.40}$$

Solving for the intersection point $(u(\mathbf{x}), v(\mathbf{x}))$ yields the coordinates:

$$u(\mathbf{x}) = \frac{h_u^2 h_v^4 - h_u^4 h_v^2}{h_u^1 h_v^2 - h_u^2 h_v^1} \tag{2.41}$$

$$v(\mathbf{x}) = \frac{h_u^4 h_v^1 - h_u^1 h_v^4}{h_u^1 h_v^2 - h_u^2 h_v^1} \tag{2.42}$$

where h_u^i, h_v^i are the i-th elements of $\mathbf{h}_u$ and $\mathbf{h}_v$. Once $(u(\mathbf{x}), v(\mathbf{x}))$ are found, they can be substituted back into Eq. (2.29) to evaluate the Gaussian at the pixel's ray, and in Eq. (2.32) to obtain the depth value z. The rasterization process is similar to standard 3DGS, with a tile-based alpha-blending-based rasterizer with frustrum culling. However, the evaluation of 2D Gaussians is computed with ray-splat intersection as $G(\mathbf{x})$ at ray $\mathbf{x}$.

$$\mathbf{c}(\mathbf{x}) = \sum_{i=1} (c_i)\sigma_i G_i(\mathbf{x}) \prod_{j=1}^{i-1} (1 - \sigma_j) G_j(\mathbf{x}) \tag{2.43}$$

To handle degenerate cases, such as when a Gaussian appears collapsed due to a slanted view, a screen-space low-pass filter is applied:

$$G(\hat{\mathbf{x}}) = \max\left(G(\mu(\mathbf{x})),\ G\left(\frac{\mathbf{x} - \mathbf{c}}{\sigma}\right)\right).$$

Here, $\mathbf{c}$ is the projected center of the Gaussian, and $\sigma = \frac{\sqrt{2}}{2}$ ensures stability and sufficient pixel coverage.

This ray-splat intersection calculated depth z, the standard Gaussian Splatting loss function is modified with an additional depth distortion loss L_d and a normal consistency loss L_n.

$$L_d = \sum_{ij} w_i w_j |\mathbf{z_i} - \mathbf{z_j}| \tag{2.44}$$

where w_i is the blending weight of the i-th ray-splat intersection given by (2.19), and z_i is its ray depth.

$$L_n = \sum_{i} w_i (1 - \mathbf{n}_i^T \mathbf{n}) \tag{2.45}$$

w_i is the same blending weight as before. $\mathbf{n}_i$ denotes the normal vector of Gaussians i oriented towards the camera, $\mathbf{n}$ is the depth map's gradient-based normal given by

$$\mathbf{n}(x, y) = \frac{\nabla_x \mathbf{p_s} \times \nabla_y \mathbf{p_s}}{|\nabla_x \mathbf{p_s} \times \nabla_y \mathbf{p_s}|} \tag{2.46}$$

where $\mathbf{p_s}$ is the median intersection point denoting the location at which the accumulated opacity reaches 0.5. L_n aligns the 2D splats to surfaces by aligning 2D splat normals to depth map normals. L_d minimizes the distance between ray-splat intersections, concentrating splats along the ray. The final loss is given by a weighted sum of all three losses:

$$L = L_c + \lambda_d L_d + \lambda_n L_n \tag{2.47}$$

where L_c is the photometric loss given by Eq. (2.21).

2.5 Large Language Models and Transformers

Large Language Models (LLMs) are deep learning models designed to process and generate human-like text. These models are trained on vast amounts of textual data and leverage transformer architectures [27] to understand text-based context, semantics, and linguistic structures. These models, designed for Natural Language Processing (NLP), are typically trained to interpret text-based prompts and generate text-based outputs in a human-like manner.

LLMs have achieved widespread adoption since 2023, having significantly advanced natural language processing (NLP) tasks, including text generation, translation, summarization, and question-answering. Beyond basic image and text interpretation, these models recently exhibited expert-level problem-solving in various scientific and engineering domains [28, 29].

The development of LLMs is driven by improvements in computational power, large-scale datasets, and optimization techniques. They employ self-supervised learning and fine-tuning strategies to adapt to domain-specific applications. However, despite their capabilities, LLMs pose challenges related to computational efficiency, interpretability, and ethical concerns, such as bias and misinformation.

Due to their large size, LLMs often face hardware constraints for local deployment. While popular LLM providers such as OpenAI and Deepseek, provide web browser interfaces for their models, they also offer Application Programming Interfaces (APIs). These APIs enable client-side software or code to query LLMs hosted on OpenAI or Deepseek servers, facilitating large-scale data processing without requiring human-in-the-loop manipulations via browser interfaces.

Transformers [27], proposed in 2017, are the primary neural networks that power large

language models. The Transformer architecture is built using attention layers and feed-forward layers (along with some other minor neural network design components). An encoder-decoder Transformer as originally proposed is illustrated in Figure 2.7, although modern Transformers are often only encoder-based or decoder-based.

The attention mechanism in transformers enables the model to weigh the relevance of different tokens in the input sequence dynamically. It does this by computing a weighted sum of value vectors, where each weight is determined by the similarity between a query vector (derived from the current token) and key vectors (derived from all tokens). This

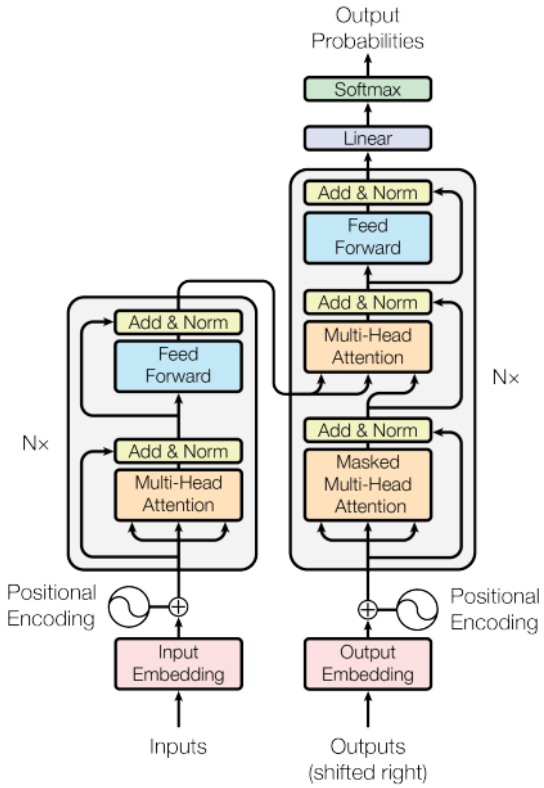


Figure 2.7: The original Transformer [27] architecture comprised of a Transformer Encoder and a Transformer Decoder.

process, known as scaled dot-product attention, allows the model to capture long-range dependencies and contextual nuances irrespective of the tokens' positions in a low inductive bias architecture. By using multiple attention heads, the transformer can learn to focus on various aspects of the data simultaneously, greatly enhancing its performance across a wide range of tasks. The equations for the multi-head attention layer are as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.48}$$

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V \tag{2.49}$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.50}$$

This architecture, comprised of attention layers and feed-forward layers, has been shown to be very scalable, more so than any other mainstream neural architecture so far. This scalability allowed for the building of large language models at the billion or trillion parameter scale.

2.6 Evaluation Metrics

Novel view synthesis metrics are used throughout multiple chapters. These are introduced here. Chapter specific metrics are introduced in the respective chapters. To assess the quality of synthesized/rendered/rasterized 2D images, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [22], and Learned Perceptual Image Patch Similarity (LPIPS) [30] are used as full-reference image assessment metrics. These

metrics compare generated images to ground truth images. PSNR is a good indicator of the presence of noise and visual artifacts, whereas SSIM and LPIPS have been shown to correlate better with human judgment of the visual similarity of an image to its reference image.

PSNR measures the difference between the original image Y and the synthesized image X in terms of the peak signal power and the power of the noise that affects the quality of the image, with higher PSNR suggesting greater image quality with less noise and fewer visual artifacts. This is given by

$$\text{PSNR(X,Y)} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE(X,Y)}} \right) \tag{2.51}$$

where MAX denotes the dynamic range/maximum possible pixel value (255 for 8-bit integers, 1.0 for floating point numbers), and MSE denotes the pixel-wise mean squared error between reference and compared images.

SSIM [22] takes into account the structural information of the images, such as luminance, contrast, and structure, and is highly correlated to human perception. It computes the similarity based on local patterns, measuring the consistency of structural information between the two images, with higher SSIM suggesting greater visual similarity between ground truth and generated image. SSIM = 1 indicates perfect agreement. This is given by

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.52}$$

where μ_x, μ_y are the pixel-intensity-based local means, and σ_x^2, σ_{xy} are local variances and covariances. $C_1, C2$ are user-defined hyper-parameters defaulting to $C1 = (0.01\text{MAX})^2$, $C2 = (0.03\text{MAX})^2$, with MAX denoting the dynamic range. The local pixel intensity statistics are calculated inside a sliding window Gaussian kernel and averaged across the

image.

LPIPS [30] computes the similarity between feature maps of the ground truth and synthesized image using pretrained convolutional neural networks. LPIPS has been shown to correlate well with human judgment in various image quality assessment tasks, with LPIPS = 0 indicating perfect agreement between the deep visual features of ground truth and synthesized image. This is given by

$$\text{LPIPS}(x, y) = \sum_{l} \frac{1}{H_l W_l} \sum_{h=1}^{H_l} \sum_{w=1}^{W_l} \omega_l \|(f_l(x)_{h,w} - f_l(y)_{h,w})\|_2^2 \tag{2.53}$$

where $f_l(.)$ denotes the convolutional feature map at layer l with height H_l and width W_l and ω_l denotes a layer-wise weight parameter with different defaults according to which pretrained convolutional neural network is used.

2.7 Google Earth Studio

Google Earth Studio [31] is a web-based animation tool that utilizes Google Earth's satellite imagery and 3D terrain data for geospatial visualizations. It enables users to extract images from specified camera poses at chosen locations, leveraging Google Earth's geographic data. The imagery is sourced from various government and commercial datasets, including but not limited to Landsat/Copernicus, TerraMetrics, Airbus, NOAA, U.S. Navy, USGS, and Maxar. Images from these sources are combined with 3D tiles on the Google Cloud back-end to generate 2D multi-view remote sensing images. In this thesis, I use Google Earth Studio to acquire multi-view remote sensing images of a building or a neighbourhood based on its address, postal code, place name, or geographic coordinates along a defined camera path.

2.8 3D Reconstruction from 2D Images

Classical 3D reconstruction from 2D images is well-studied [32]. Photogrammetry is well understood in remote sensing and computer vision and is widely used for academic, industrial, and commercial purposes. For this purpose, multiple software suites and libraries [15, 33–37] are available, with some specifically designed for remote sensing applications. Many of these not only allow for the extraction of a sparse point cloud from 2D images but also generate a 3D mesh. These 3D meshes are often more desirable than point clouds, as they are more photorealistic and better suited for 3D modelling and simulations.

COLMAP [15], in particular, is worth highlighting. It is a photogrammetry and Structure-from-Motion (SfM) library widely used as a preprocessing step for modern novel view synthesis and learning-based 3D rendering methods. It is included in most Gaussian Splatting models' pipelines [6], including my own.

2.9 Developments in Neural Radiance Fields

Neural Radiance Field (NeRF) [19], introduced in 2020, has garnered significant attention in 3D computer vision. In NeRF-based models, the 3D representation consists of a 3D radiance field (directionally dependent colour field) and a 3D density field, both represented as Multi-Layer Perceptrons (MLPs). Differentiable volume rendering is employed to generate 2D images by sampling and integrating local 3D colour, radiance, and density. These radiance and density fields are trained from scratch using a photometric loss function, allowing NeRF models to learn a 3D representation of a scene from 2D images. Significant advancements have been made in this area since 2020, including [38–40], which all improve the fundamental aspects of the NeRF rendering process, resulting in higher

view synthesis quality and rendering speed.

2.9.1 Geometry Extraction

NeRF methods are implicit or hybrid representation methods, which means that their learned 3D representations are either fully or partially stored inside deep neural networks. However, many NeRF-adjacent differentiable volume rendering methods exist [24, 41, 42] that store 3D representations explicitly. The Plenoxels paper is worth highlighting, as Plenoxels was the first recent neural rendering method to use spherical harmonics-based colour encoding for novel view synthesis. The original Gaussian Splatting paper [20], along with the vast majority of subsequent Gaussian Splatting papers (those that do not modify colour encoding), uses the same implementation of spherical harmonics as Plenoxels.

Extracting 3D geometry from NeRF is not straightforward. The naive approach is to voxelize the NeRF density field stored inside the NeRF MLP(s) by querying and discretizing it in a regular grid. In practice, this approach returns poor quality 3D geometry and struggles to properly reproduce solid 3D surfaces. The issues are mainly caused by the smooth implicit density field, which needs to be thresholded at solid surfaces. However, there is no fixed threshold that works well across all scenes. Even within a scene, a fixed threshold often fails to discretize all surfaces properly.

Some methods focus on improving 3D geometry extraction by transforming the implicit representation from a density field to a distance function-based approach [43–46] or occupancy field-based approach [47]. This allows the NeRF MLP to learn and store a signed distance function–like 3D representation, which can be queried directly. These 3D representation fields are then converted into the density fields required for NeRF volume rendering.

2.9.2 NeRF in Remote Sensing and/or Urban Environments

Urban scenes, unbounded, full of transient objects (e.g., pedestrians and cars), and with changing lighting conditions pose a challenge to the learning of 3D scene representation. Methods like NeRF-W [48], Mip-NeRF-360 [49], Block-NeRF [50], and Urban Radiance Fields [51] proposed solutions to some of these problems and are suited for ground-level view synthesis and 3D urban reconstruction. Aerial view 3D reconstruction and view synthesis from remote sensing images have also been attempted with methods like Bungee/City-NeRF [52], Mega-NeRF [53], Shadow NeRF [54], and Sat-NeRF [55]. These methods aimed to solve problems such as stitching local NeRFs into large-scale urban scenes, multi-scale city view synthesis, and shadow-aware scene reconstruction for high-rises.

2.10 Developments in Gaussian Splatting

The original 3D Gaussian Splatting (3DGS) model [20], first proposed in 2023, and subsequent Gaussian Splatting models have largely surpassed NeRF-based approaches over the past year. 3DGS uses a large number of 3D Gaussian distributions (also known as primitives in computer graphics) as a learned 3D representation. In addition to standard 3D Gaussian distribution function parameters, each Gaussian primitive also has directionally dependent colour and opacity α, all of which are trainable parameters. The method uses a differentiable tile-based rasterizer, projecting the 3D Gaussians onto the image to be rasterized, and α-blending the projected Gaussians. This allows the 3D Gaussians representing the scene's radiance and geometry to be learned from scratch using a photometric loss function. Compared to NeRF, Gaussian Splatting models are generally much faster to train and have higher view-synthesis quality, but require more memory and storage.

2.10.1 Gaussian Splatting in Remote Sensing and Urban Environments

Methods such as [56–60] applied Gaussian Splatting to remote sensing or improved Gaussian Splatting for urban scene capture, which addressed challenges arising from unbounded scenes, transient objects, large scene scale, and incomplete view coverage. Hugs [56], which was jointly optimized for view synthesis, semantics, and motion using both static and dynamic Gaussians, achieved excellent results across a wide variety of KITTI [61] benchmark tasks. HO-Gaussian [57], optimized for KITTI-like large-scale driving datasets by eliminating the SfM initialization requirement, an alternative colour encoding method, and introducing a method to ensure object consistency across multiple cameras. Hierarchical-3DGS introduced a hierarchical approach for large-scale street-level scenes, training by separating the scene into multiple chunks and consolidating chunks into an optimized hierarchy-based 3D Gaussian representation. A hierarchy-based rasterization method was also introduced [58].

[59] specifically addressed few-shot Gaussian Splatting training in remote sensing applications, where typical multi-view remote sensing datasets lack viewpoint variety, in contrast to commonly used 3D computer vision benchmark datasets. The authors employed the pseudo-viewpoint augmentation method first proposed in [62], training each input image under both its real and randomly augmented viewpoints. [60] applies Gaussian Splatting for large-scale terrain reconstruction from remote sensing images from the STPLS3D dataset [63], and integration with 3D engines, assessing scalability and applications to 3D simulations.

These methods focus on scene-wide capture and are not designed to extract the 3D mesh of individual objects without further processing.

2.10.2 3D Geometry Extraction

Gaussian Occupancy Field (GOF) [64] modifies the rendering process from rasterization to volume rendering (similar to NeRF models). Camera rays are traced from each pixel to be rendered and intersected with Gaussians. Together with depth-based loss regularization, this results in more accurate 3D geometry.

In contrast, 2D Gaussian Splatting (2DGS) [26], previously introduced in Section 2.4.3, is a significant improvement on 3DGS itself, altering the nature of the Gaussian splats. As a result, 2DGS can replace 3DGS in many pipelines, as demonstrated in this thesis. Moreover, the use of surface-aligned 2D Gaussians as graphical primitives alongside depth regularization yields more accurate 3D geometry reconstruction.

GS2Mesh [65], being an out-of-the-box pipeline leveraging Segment Anything Model object masking, allows for the extraction of a 3D mesh from user prompting without further training or processing. It motivated the research in Chapter 4. GS2Mesh does not fundamentally change the Gaussian Splatting process but builds a 3D mesh extraction pipeline using the standard 3D Gaussian Splatting method.

2.11 Developments in Large Language Models

Large Language Models are focused on large-scale (pre) training on large unsupervised/self-supervised datasets. Popular training schemes include the masked language model scheme [66] for encoder-based architectures and the autoregressive causal language model scheme [67] for decoder-based architectures. More specialized models (as opposed to general-purpose large language models) can sometimes also use the sequence-to-sequence training scheme [68] (e.g. translation models) or contrastive training [69] (e.g. text-image dual

modality models).

The most influential publications and models include the Bi-directional Encoder Representation Transformer (BERT) [66], Generative Pretrained Transformers (GPT-1 to GPT-4 [67, 70–72]), with GPT-3 (2020) being the first truly huge LLM with 175 billion parameters. LLaMA [73–75], focused on efficiency and was released as open-source, which resulted in a great research impact. Subsequent versions [74, 75], focusing on fine-tuning and multi-modality, followed this approach.

Certain models, such as DeepseekV3/R1 and their variants [76, 77], support Optical Character Recognition (OCR, i.e., reading text from images). Models like "ChatGPT-4o" [72] and its variants, and LLaMA 3.2+ [78], support full interpretation and analysis of image content. This can be achieved by using image encoders with large-scale image-text pretraining which allows for embedding of text and image to a joint latent space.

A representative example of this type of training is the Contrastive Language Image Pretraining (CLIP) [69]. CLIP jointly trains an image encoder and a text encoder to align their latent representations in a shared embedding space. The training objective brings representations of predefined matching image–text closer together while pushing apart non-matching pairs using a contrastive loss given by

$$\mathcal{L} = \frac{1}{2}\left[-\frac{1}{N}\sum_{i=1}^{N}\log p_{ii} - \frac{1}{N}\sum_{i=1}^{N}\log p'_{ii}\right] \tag{2.54}$$

where p_{ii} is the probability that image i matches its text, and p'_{ii} is the probability that text i matches its image. This is given by a softmax of cosine similarities between the two respective embeddings.

$$p_{ij} = \frac{\exp\left(s_{ij}/\tau\right)}{\sum_{k=1}^{N} \exp\left(s_{ik}/\tau\right)}. \tag{2.55}$$

where s_{ij} is the cosine similarity of image i and text j given by Eq. (5.5) and τ is a scaling hyperparameter.

A CLIP-trained image encoder can then be used to embed images and image tokens in a way that is compatible with text token embedding for large language models.

Chapter 3

Enhanced 3D Urban Scene Reconstruction and Point Cloud Densification using Gaussian Splatting and Google Earth Imagery

This chapter is based on my research on large-scale urban Gaussian Splatting. This research produced one of the earliest works on this aspect of Gaussian Splatting (despite the fairly recent publication date). The chapter explores the use of Gaussian Splatting for 3D view synthesis with hierarchically arranged remote-sensing scene images and the resulting densification of 3D point clouds. Compared to earlier NeRF-based, MVS-based, and traditional Structure From Motion, the result is more efficient, geometrically accurate, and visually impressive. This chapter seeks to answer the following questions. The chapter is constituted of my research paper eponymous with the chapter title [79].

This chapter is organized as follows: Section 3.1 is the chapter's introduction which motivates and briefs the chapter's research. Section 3.2 is the methodology section which describes the chapter's research. Section 3.3 presents and discusses the chapter's results. Section 3.4 is the chapter's conclusion.

3.1 Introduction

Using only images from Google Earth Studio arranged in a hierarchical camera path comprised of concentric circles of increasing radius found to be ideal for large-scale urban scenes, I train a 3D Gaussian Splatting model and a 2DGS+ model that outperforms previous NeRF-based models in terms of view synthesis results, and traditional point cloud densification in terms of 3D reconstruction. I quantify and benchmark the view synthesis performance on a large-scale urban dataset with 10 cities captured from Google Earth Studio, as well as the region of study. I also extract and densify the 3D geometry of the region of study using 3DGS and compare it against a Multi-View-Stereo dense reconstruction. Additionally, I experiment with the effect of simulated sunlight/time of day on Gaussian Splatting. This chapter aims to answer the following questions.

- How does 3DGS compare with NeRF-based approaches for large-scale urban scenes?
- How does the standard 3DGS approach compare to more modern Gaussian Splatting approaches?
- How well does the extraction of 3D geometry from Gaussian Splatting models as densified point clouds work for urban scenes?

Overall, the Gaussian Splatting-based methods (both recent and older) were found to be more efficient, accurate, and visually impressive than NeRF-based, or traditional methods.

3.2 Methodology

3.2.1 Study Area

The study area is the Kitchener-Waterloo Region in Ontario, Canada, centred at the University of Waterloo. The city of Waterloo has a population of approximately 121 000 according to the 2021 census, and occupy 64.06 km^2 of land [80]. The University of Waterloo lies at 43.472°N, 80.550°W, its main campus occupies 4.50 km^2. At the city scale, the study area is comprised of various land use and land cover features such as urban roads, buildings, agriculture, and other land use, low vegetation, water, mixed temperate forest, and other land cover. For the view synthesis experiment, the study area is centred at the Environment-1 (EV-1) building located at 43.468°N, 80.542°W, and covers roughly an area of 165 km^2. I perform large-scale view synthesis at the city scale, and 3D point cloud comparison at the neighbourhood scale in an area of roughly. The Google Earth images retrieved for the scene are primarily from Landsat/Copernicus, Airbus, Scripps Institution of Oceanography (SIO), and National Oceanic and Atmospheric Administration (NOAA).

The study area lies on the traditional land of Neutral, Anishinaabeg and Haudenosaunee peoples. The study area is situated on the Haldimand Tract, the land promised to the Six Nations that includes six miles on each side of the Grand River.

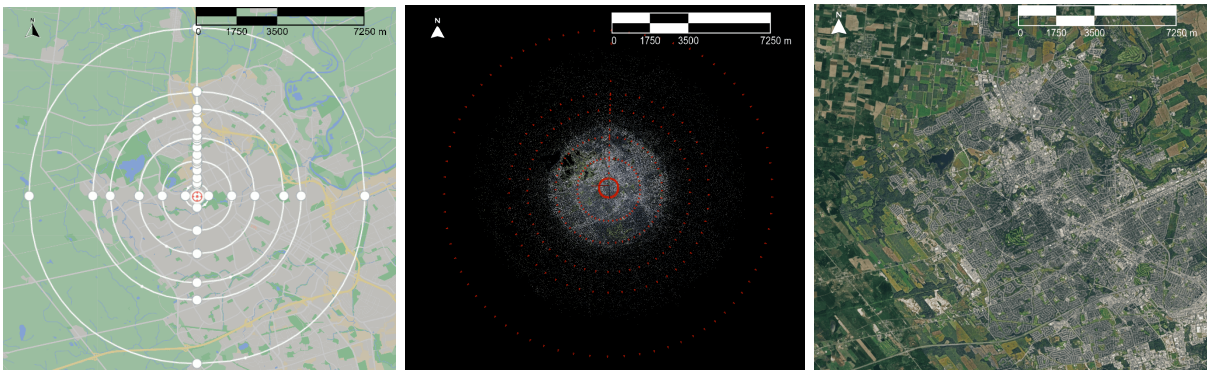


Figure 3.1: Plot of camera poses, and reconstruction for the study area: the Waterloo scene, centred on the EV1 building at the University of Waterloo. **Left**: Google Earth Studio camera path. **Middle**: COLMAP [15] SfM reconstructed sparse point cloud with projected colour and camera poses (as red dots). **Right**: Google Earth © aerial image of the study area, sourced from Airbus.

3.2.2 Google Earth Studio

For the study area, as camera paths, I used seven/(or some other user-defined number) concentric circles at different altitudes, radii and tilt angles centred around the EV-1 building at the University of Waterloo, Waterloo, Ontario, Canada. **This design choice was inspired by BungeeNeRF [81], and is a major contributing factor in the 3D reconstruction results.**

The first of these circles are of has a radius of 500 m, and an elevation of 475 m. The last circle has a radius of 7250 m and an elevation of 3690 m. All images point towards and above (at elevation 390 m) the EV-1 building in the University of Waterloo at 43.468 °N, 80.542 °W. The final circle's images have a tilt angle of approximately 65.5 ° with respect to the horizontal with some deviations (within ~ 0.3 °). I gathered 401 images using Google Earth Studio along the camera path defined using these circles. The study area and camera poses, along with sparse SfM results can be seen in Figure 3.1. During preprocessing, poor

SfM point cloud reconstruction results further than 6 km away from the scene center can be observed, reasonable SfM reconstruction within 6 km, and good SfM reconstruction within 1 km where individual buildings can be identified. The SfM preprocessing resulted in a sparse point cloud with 337382 points which were used to initiate 3D Gaussian functions for 3DGS. This multi-scale Google Earth Studio [31] dataset was inspired by the BungeeNeRF dataset [52], which I also use for a multi-city large-scale view-synthesis benchmark.

For the BungeeNeRF scenes, I used the Google Earth Studio camera paths specified by BungeeNeRF [52]. The BungeeNeRF dataset consists of 10 scenes for 10 cities. Each scene is centred around a particular landmark, with camera paths defined by concentric circles of different orbit radius and elevation, with the scene coverage reaching city-wide at the highest elevation. Detailed information can found in Table 3.1 for the 10 BungeeNeRF scenes and the Waterloo scene. The New York scene centred at 56 Leonard and the San Francisco scene centred at Transamerica were used as main scenes for view reconstruction benchmark in BungeeNeRF [52], and have 459 and 455 images respectively. These two scenes were rendered at 30 frames per second in a 1:30 minute video. All other scenes contain 221 images, and were rendered by fixing a frame limit of 220+1 given the fixed camera path, and were used for additional visualizations. The original BungeeNeRF paper contained two additional scenes (Sidney and Seattle), but the Google Earth Studio camera paths were not provided for these two scenes.

Google Earth Studio allows simulated illumination changes from the sun's position (time of day) and the sky's condition (clear sky/overcast). For the BungeeNeRF scenes' experiments and the experiments in Waterloo, the study area, I used the default option with a clear and uniformly illuminated sky, with no directional illumination from the sun. Additional experiments on the effect of changing illumination with time of day and sky condition can be found in the Appendix.

Table 3.1: Google Earth Studio City Scenes Information

City	Landmark	Lowest (m)	Highest (m)
New York	56 Leonard	290	3,389
San Francisco	Transamerica	326	2,962
Chicago	Pritzker Pavilion	365	6,511
Quebec	Château Frontenac	166	3,390
Amsterdam	New Church	95	2,3509
Barcelona	Sagrada Familia	299	8,524
Rome	Colosseum	130	8,225
Los Angeles	Hollywood	660	12,642
Bilbao	Guggenheim	163	7,260
Paris	Pompidou	159	2,710
Waterloo	EV-1	500	3690

3.2.3 Multi-View-Stereo Dense 3D Reconstruction

The MVS dense reconstruction I used as ground truth/reference geometry of the study area is retrieved from COLMAP's MVS algorithm [82]. This method is based on joint view selection and depth map estimation [83]. The method is summarized as follows.

Depth and normal map estimation: Depth and normal maps are estimated for a reference image by maximizing a joint likelihood function. This function considers all images in a dataset, where some serve as reference and others as sources. It incorporates three key components:

- Occlusion Mapping: Ensures smooth and consistent occlusion handling across spatial regions and optimization iterations.
- Photometric Consistency: Aligns the reference and source images based on pixel similarity.
- Geometric Consistency: Maintains alignment between depth and normal maps across multiple views.

These are used to refine depth and normal maps, ensuring accuracy and consistency across multi-view geometry. For further details on the mathematical formulation and likelihood function optimization, refer to [15].

Filtering and fusion: Photometric and geometric constraints are applied to filter outliers, ensuring that each reference pixel is observed in at least three other images while satisfying these constraints. A directed graph is created with consistent pixels as nodes and edges connecting reference and source images. Fusion starts at the node with the highest support (observed by the most source images). Nodes are recursively collected based on depth, normal, and reprojection error constraints, and their attributes are fused into a dense point cloud. Fused nodes are removed, and the process repeats until the graph is empty. The final result is a dense point cloud with normals, which can be converted into a mesh using methods like Poisson Surface Reconstruction.

3.2.4 Gaussian Splatting

This chapter uses 3D Gaussian Splatting (3DGS) and 2D Gaussian Splatting+ (2DGS+) for large-scale remote sensing urban view synthesis and 3D reconstruction. Although 3DGS and 2DGS are detailed in Chapter 2, they are briefly explained here for completeness. 2DGS+ builds upon 2DGS with multiple improvements and is introduced in more detail in this section.

3DGS

3DGS [20] the original Gaussian Splatting method builds upon EWA splatting [21]. It uses 3D Gaussian distributions as graphical primitives and a projection-based rasterization method for 2D rendering. As learnable parameters, these 3D Gaussians each have a mean

(as 3D in-scene positions), a covariance matrix, as well as attached spherical harmonics coefficients for directionally dependent colour representation (typically up to degree 3, in 3 colour channels), and a Gaussian opacity.

The Gaussians rendered into a 2D image using a tile-based projection-based rasterization method (unlike the ray-tracing-like volume rendering method of NeRF). The Gaussians are mapped to the image-space coordinates. This mapping is essentially a rotation and translation mapping from world-space coordinates to camera space coordinates and a projection mapping from camera space coordinates to image-space coordinates. Starting in homogenous coordinates in world-space, this is given by

$$\Sigma' = JW\Sigma W^T J^T \tag{3.1}$$

where W is the view transformation and J is the Jacobian of the affine approximation of the projective transformation.

Gaussians are then sorted (and culled) according to tile and by depth. For each pixel in a tile, the pixel's colour is generated via alpha blending accumulating in-scene direction-dependent colour using the learned spherical harmonics coefficients (Section 2.4.2). This is given by

$$\hat{C}(u,v) = \sum_{i=1}^{N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j) \tag{3.2}$$

where $\mathbf{c}_i$ denotes the ith Gaussian's directionally dependent colour and α_i denotes its alpha value. The alpha-value α_i at a pixel location (u,v) is given by evaluating the 2D Gaussian, scaled by its associated learned opacity o_i scaled by the sigmoid function σ.

$$\alpha_i(u,v) = \sigma(o_i) G_i^{2D}(u,v) \tag{3.3}$$

where $G^{2D}(x)$ is the Gaussian (Eq. 2.11) projected into 2D dimension and onto the image plane via (Eq. 2.14).

2DGS and 2DGS+

One of the goals of this chapter is to compare the results from 3DGS, the original Gaussian Splatting implementation against a state-of-the-art version that integrates numerous innovations in terms of the densification and pruning processing, the optimization process, and various other improvements from recent Gaussian Splatting research. I decided to use 2DGS+ [84], built upon 2DGS [26] as a representative state-of-the-art method. I describe both 2DGS and 2DGS+ in this subsection.

Compared to 3DGS, 2DGS uses surface-aligned 2D Gaussian distributions as graphical primitives. This results in a slightly different functional form for the Gaussian and its learned parameters. More importantly, the method modifies the standard projection method with a ray-splat intersection-based projection method which is slightly more similar to NeRF-based volume rendering. This is described in more detail in Section 2.4.3, and can be found in [26].

Additionally, 2DGS provides geometry supervision with an additional depth distortion loss L_d and a normal consistency loss L_n. These are given by

$$L_d = \sum_{ij} w_i w_j |z_i - z_j| \tag{3.4}$$

where w_i is the blending weight of the i-th ray-splat intersection given by (2.19), and z_i is

its ray depth as calculated from the ray-splat intersection.

$$L_n = \sum_i w_i(1 - \mathbf{n}_i^T \mathbf{n}) \tag{3.5}$$

w_i is the same blending weight as before. $\mathbf{n}_i$ denotes the normal vector of Gaussians i oriented towards the camera, $\mathbf{n}$ is the depth map's gradient-based normal given by

$$\mathbf{n}(x, y) = \frac{\nabla_x \mathbf{p_s} \times \nabla_y \mathbf{p_s}}{|\nabla_x \mathbf{p_s} \times \nabla_y \mathbf{p_s}|} \tag{3.6}$$

where $\mathbf{p_s}$ is the median intersection point denoting the location at which the accumulated opacity reaches 0.5. L_n aligns the 2D splats to surfaces by aligning 2D splat normals to depth map normals. L_d minimizes the distance between ray-splat intersections, concentrating splats along the ray. The final loss is given by a weighted sum of all three losses:

$$L = L_c + \lambda_d L_d + \lambda_n L_n \tag{3.7}$$

where L_c is the photometric loss given by

$$L = (1 - \lambda)L_1 + \lambda L_{D-SSIM} \tag{3.8}$$

where a pixel-wise L_1 loss and a difference of SSIM [22] loss. 2DGS+ is a state-of-the-art implementation of Gaussian Splatting built on the foundations of 2DGS, created by multiple GitHub contributors. I unofficially refer to this implementation as 2DGS+ [84]. It is an unpublished fork of the 2DGS repository. It combines many recent advances in Gaussian Splatting, improving 2DGS with ideas from AbsGS [85], PixelGS [86], TrimGS [87], AtomGS [88], StableNormal [89], GaussianPro [90], GOF [64], and Taming-3DGS [91].

2DGS+ introduces densification improvements from AbsGS [85] and PixelGS [86]. Namely, a new densification criterion from AbsGS is introduced based on the magnitude of each gradient component. Inspired by PixelGS, pixel coverage is employed as a weight to average the gradients from different views before the densification check. Both methods reduce needle-like artifacts (long floaters).

From TrimGS [87], 2DGS+ integrates a better pruning method by trimming redundant and inaccurate Gaussians using an α-blending weight-based criterion, replacing the original opacity-based criterion. 2DGS+ employs the progressive training method from Gaussian-Pro [90], which includes multi-view-stereo-based densification and depth supervision. This method improves the densification process by leveraging patch-matching techniques and geometric priors to generate new Gaussians with already accurate positions and orientations.

From AtomGS [88], 2DGS+ integrates improved normal supervision by creating a new edge-consistent normal loss coupling a curvature map and an edge map into the normal supervision. This replaces the normal consistency loss of 2DGS.

From GOF [64], 2DGS+ integrates the Appearance Network (a small convolutional neural network that encodes per-image appearance changes within the same scene) and the Cloning with Sampling method, which resamples the new Gaussians' parameters during cloning to prevent stacking post-cloning.

2DGS+ uses Fused-SSIM from Taming-3DGS [91], a much faster implementation of the structural similarity index measure, instead of the PyTorch implementation of SSIM, which is used in the photometric loss term of Gaussian Splatting. Additionally, 2DGS+ supports DUST3R-based [92] initialization as an alternative to COLMAP. DUST3R is generally faster than COLMAP and returns more accurate camera poses, but struggles

with memory limitations when processing multiple images.

3.2.5 Metrics

For novel view synthesis, the metrics are introduced in Section 2.6. For point cloud geometry assessment, point-to-point (D1) mean squared error (MSE), point-to-surface (D2) MSE, Hausdorff distance, and Chamfer distance are used, all of which compare a lower quality point cloud to its reference point cloud [93]. Metrics such as D1 and D2 MSE do not penalize differences in point density, only deviations of existing points from ground truth/reference points. On the other hand, Chamfer and Hausdorff distances better capture the difference between the distribution of points, including differences in point density. D1 and D2 MSE are simple mean squared error formulae. The Hausdorff distance between a reference point cloud B and an assessed point cloud A is given by

$$d_H(A,B) = \max\left\{\max_{a\in A}\min_{b\in B}\|a-b\|, \max_{b\in B}\min_{a\in A}\|b-a\|\right\}. \tag{3.9}$$

This represents the possible maximum distances between any point and its nearest representative in the opposite point cloud. The Chamfer distance on the other hand, is given by the sum of pair-wise distances from any point to its nearest representative in the opposite point cloud.

$$d_{\text{Chamfer}}(A,B) = \sum_{a\in A}\min_{b\in B}\|a-b\|^2 + \sum_{b\in B}\min_{a\in A}\|b-a\|^2. \tag{3.10}$$

Point clouds are registered to each other prior to calculating these metrics. The two point clouds are aligned as closely as possible to each other using rotations, translations, and potentially re-scaling.

3.3 Experiments and Discussion

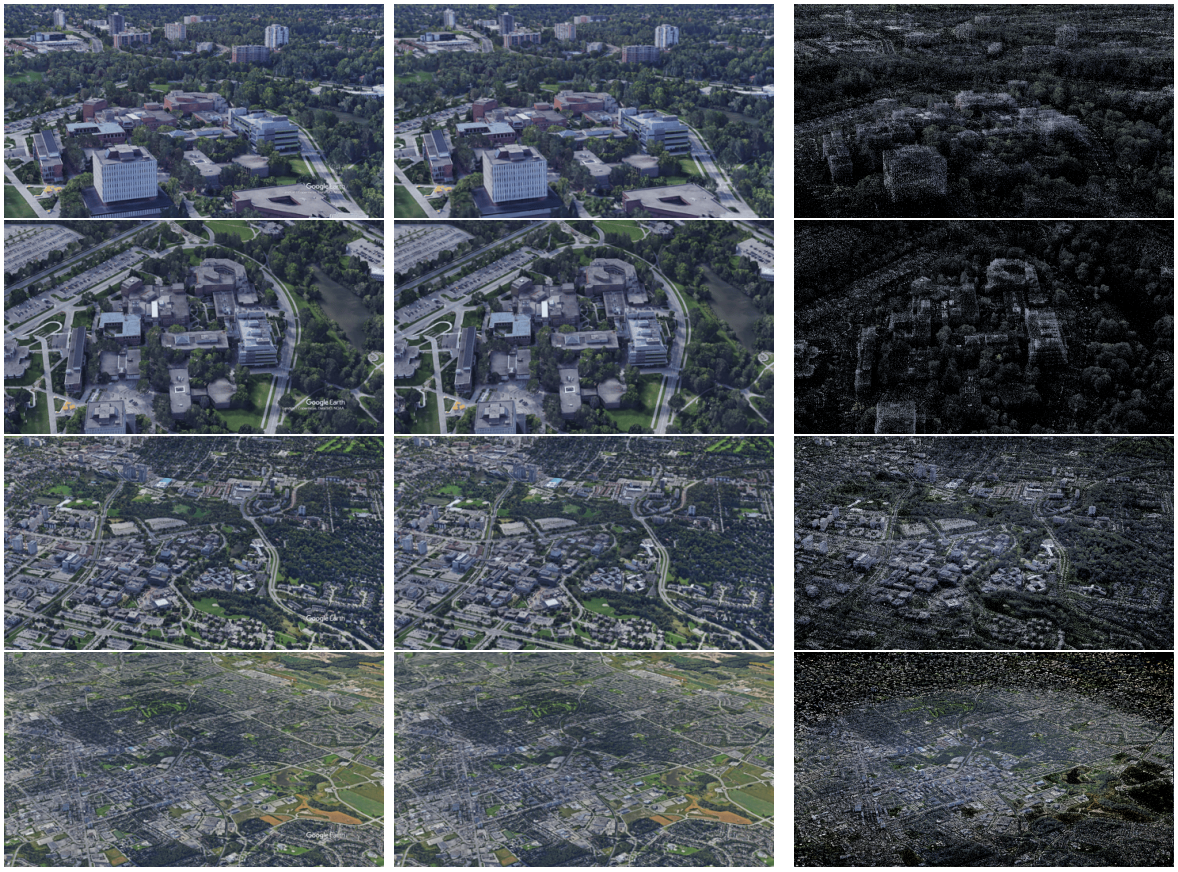


Figure 3.2: Ground truth, generated images, and visualization of Gaussian means of our Waterloo scene at different altitudes and orientations. **Left**: Waterloo scene ground truth; **Middle**: Waterloo scene 3DGS generated image; **Right** Waterloo scene visualization of location of each 3DGS Gaussian, i.e. 3D positional mean of each Gaussian. These points were then extracted as point clouds.

3.3.1 Novel View Synthesis in the Study Area

For the region of study, I used COLMAP SfM [15] preprocessing and extracted 3D points and camera poses from the 400 2D images. The experiments were performed with MipNeRF-360 [49] style training validation split: one in eight images ($\sim$ 12.5%) were reserved for testing purposes. The 1920 $\times$ 1080 pixels images were downsampled by a factor of 4 during training due to GPU memory constraints. I started densification at the 1000th iteration, and trained for 50000 iterations, densifying every 100 iterations. I used an initial positional learning rate of 3.2×10^{-5} and a scale learning rate of 2×10^{-3} based on preliminary experiments. The other training hyperparameters were kept as default.

The results are shown in Table 3.2 and Figure 3.2, in conjunction with further view synthesis experiments from the BungeeNeRF dataset. I achieve high view synthesis visual quality on both the training and test sets. From visual inspection, the rendered images are nearly indistinguishable from the ground truth images. This is also supported by the visual assessment metrics, with SSIM scores near 1, LPIPS scores near 0, which indicates almost perfect visual agreement between ground truth and generated images. The PSNR values of around 30dB also are indicative of good image quality and low level of noise. This is comparable to the PSNR of a compressed image with respect to a full-sized image with a good lossy compression algorithm [94], which is impressive considering the 3DGS model was trained at $\frac{1}{4}$ resolution.

3.3.2 3D Novel View Synthesis of Bungee-NeRF Scenes

For the BungeeNeRF scenes, the experiments were performed with MipNeRF-360 style training and validation split as previously described. The experimental settings were kept the same as the Waterloo scene experiments, except I reduced the total number of training

iterations to 30000. BungeeNeRF provided detailed benchmarks for the New York and San Francisco scenes, which were used for their main view-synthesis experiments. I performed detailed comparisons of Gaussian Splatting against BungeeNeRF, vanilla NeRF and Mip-NeRF for these two scenes. Additionally, I trained and evaluated Gaussian Splatting models for the remaining eight scenes whose camera paths were provided by BungeeNeRF.

As observed in Table 3.3, across both the New York and San Francisco scenes, I see a significant increase in view synthesis quality according to all three metrics. The visual quality improvement from 3DGS to BungeeNeRF is much larger than the quality improvement from BungeeNeRF to vanilla NeRF or any other benchmarked models. The large increase in view synthesis quality does not come at the cost of training time. In fact, the training of Gaussian Splatting models is three to four orders of magnitudes faster than implicit NeRF models such as vanilla NeRF [95], Mip-NeRF [96], and BungeeNeRF [96]. Gaussian Splatting models achieve higher view synthesis quality with faster training and rendering time at the cost of memory and storage requirement [97]. A comparison to a much more modern implementation of 3DGS is included in Table 3.3, trained using default hyperparameters, using the same number of training iterations as the presented 3DGS model. This implementation is dubbed 2DGS+ [84], which combines many recent advances in 3DGS. The visual quality of 2DGS+ did not exceed that of the vanilla 3DGS in these experiments, likely due 1) to fine-tuning of 3DGS parameters on my part, 2) the visual quality of the presented 3DGS results are nearly indistinguishable from ground truth, and 3) the improvements in 2DGS+ are not aimed at improving visual quality on these large scale scenes, but instead fix other issues with 3DGS.

As observed in the qualitative comparison in Figure 3.3, the images rendered using 3DGS are of high visual quality, and difficult to distinguish visually from ground truth images except for a Google Earth watermark noticeable at the bottom right of ground

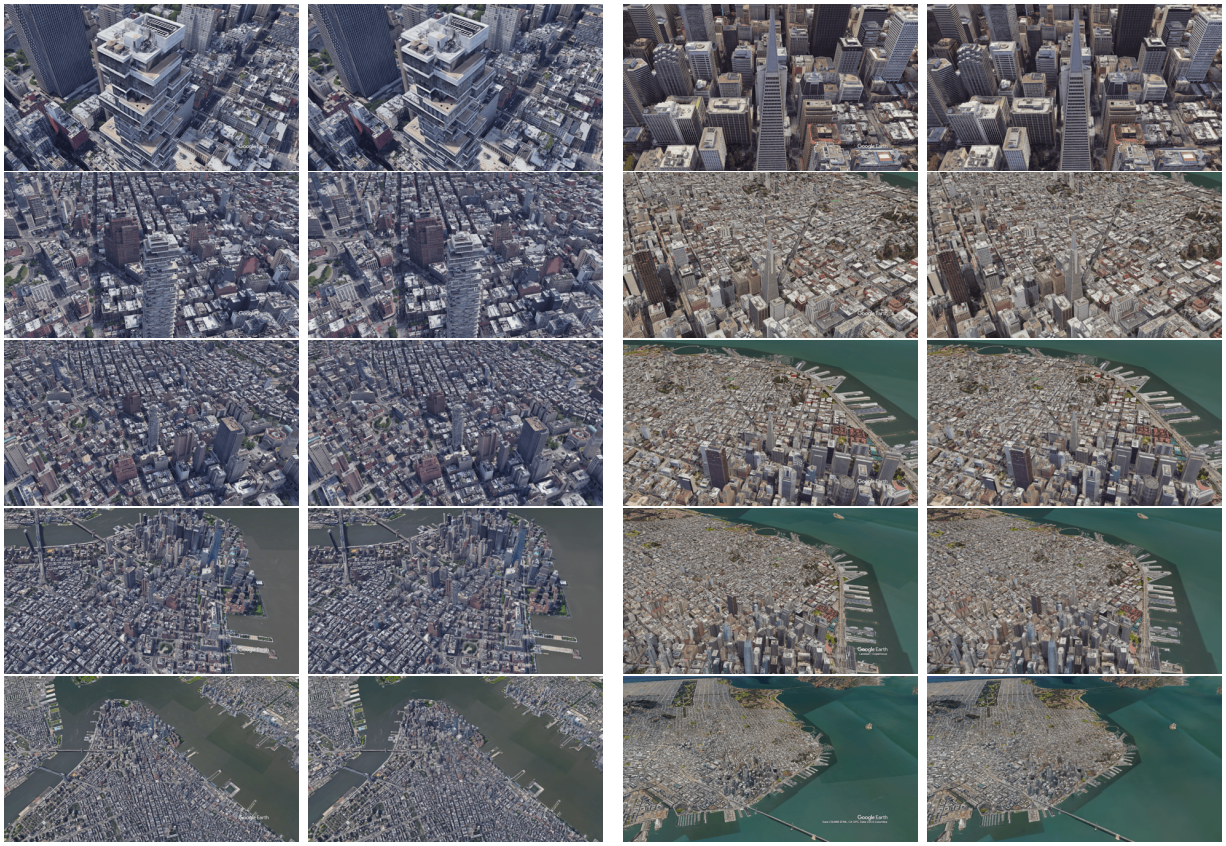


Figure 3.3: Ground truth vs rendered images of the New York and San Francisco scenes at different altitudes and orientations. **Left to right**: New York ground truth; New York render; San Francisco ground truth; San Francisco render.

truth images. Compared with ground truth images, the rendered images have slightly blurrier edges at the smallest scale ($\sim$ 300 m altitude) and certain street-level details are slightly less sharp at the largest scale ($\sim$ 3000 m altitude). Also noticeable in ground truth Google Earth images and rendered images is the piecing together of multiple data sources at the largest scale at the bottom row of Figure 3.3. There is a visible discontinuous and grid-like change in colouration of the water going from the San Francisco shoreline to the San Francisco Bay and Golden Gate area, likely indicating areas where different aerial



Figure 3.4: Novel view synthesis (rendered images) comparison of NeRF-based results on the San Francisco scene. **Left to right**: NeRF, Mip-NeRF, BungeeNeRF L1, BungeeNeRF L4.

and/or satellite images were stitched together. This effect was also learned by the 3DGS model, as is visible in the respective rendered images.

NeRF-based novel view synthesis images are included in Figure 3.4. These images can be compared with 3DGS and ground truth results at the top-left of Figure 3.3. Due to limitations in hardware, I used smaller NeRF models than suggested by the BungeeNeRF [52] authors, with a lower network depth at D = 6 and half the network width at 128. Despite this, the training time was significantly longer compared to 3DGS across all NeRF models by roughly a factor of 100, with BungeeNeRF L4 requiring around 400 times the training time due to four separate training stages. The leftmost three models of Figure 3.4 have likely not converged compared to the NeRF benchmarks in Table 3.3 provided in [52]

In addition, I also tested the performance of 3DGS on the other BungeeNeRF Google Earth Studio scenes. A 0.7 to 5.7 PSNR drop can be observed when moving from training set to test set across all scenes in Table 3.1, indicating a certain degree of overfitting across training views. The overfitting is more severe on the $\sim$ 200 image scenes compared to the $\sim$ 450 image New York and San Francisco scenes, with the 400 image Waterloo scene lying in the middle. The Bilbao scene, centred on the Guggenheim museum, has by far the worst performing 3DGS reconstructions. This is perhaps due to a combination of the complex building shape of the Guggenheim museum, a lack of sufficient training views at low altitude, and poorer quality Google Earth 3D models at off-nadir view angles at low

altitude which resulted in poor quality training images.

Table 3.2: Training set PSNR, and Test set PSNR/SSIM/LPIPS results across region of interest and various BungeeNeRF scenes.

Dataset	Train (PSNR↑)	Test (PSNR↑	SSIM↑	LPIPS↓)
Waterloo	32.3	30.5	0.953	0.053
New York	31.5	30.7	0.964	0.050
San Francisco	30.8	29.9	0.952	0.067
Amsterdam	32.3	29.7	0.948	0.053
Barcelona	31.2	28.1	0.937	0.066
Chicago	32.3	30.0	0.959	0.046
Los Angeles	32.0	28.6	0.914	0.094
Paris	31.6	28.5	0.953	0.051
Rome	32.7	27.0	0.861	0.112
Quebec	32.9	30.1	0.953	0.060
Bilbao	32.1	27.2	0.851	0.141

Table 3.3: PSNR, LPIPS, and SSIM benchmarks on BungeeNeRF New York and San Francisco scenes, NeRF results cited from [52].

	New York (56 Leonard)			San Francisco (Transamerica)		
Method	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑
NeRF (D=8, Skip=4) [95]	21.7	0.320	0.636	22.6	0.318	0.690
NeRF w/ WPE (D=8, Skip=4) [95]	21.6	0.365	0.633	22.4	0.331	0.680
Mip-NeRF-small (D=8, Skip=4) [96]	22.0	0.344	0.648	22.7	0.327	0.687
Mip-NeRF-large (D=10, Skip=4) [96]	22.2	0.318	0.666	22.5	0.330	0.686
Mip-NeRF-full (D=10, Skip=4,6,8) [96]	22.3	0.266	0.689	22.8	0.314	0.699
BungeeNeRF (same iter as baselines) [52]	23.5	0.235	0.739	23.6	0.265	0.749
BungeeNeRF (until convergence) [52]	24.5	0.160	0.815	24.4	0.192	0.801
3DGS	**30.7**	**0.050**	**0.964**	**29.9**	**0.067**	**0.952**
2DGS+	29.9	0.056	0.954	29.3	0.074	0.944

3.3.3 3D Reconstruction of the Study Area

For the 3D reconstruction experiments, due to computational constraints of multi-view stereo (MVS) densification, which is even more memory intensive than 3DGS, I extracted the sparse point cloud using the first 50 images along the first level camera path. Then,

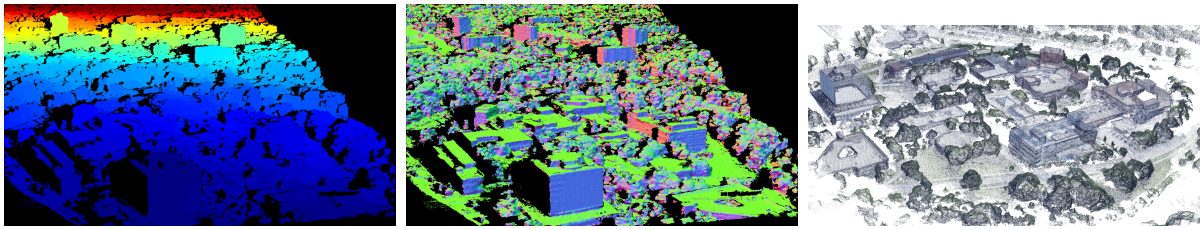


Figure 3.5: Multi-View-Stereo densification results visualization. **Left**: Example depth map. **Middle**: Example normal map. **Right**: MVS reconstructed dense coloured point cloud.

using [82], I generated depth and normal maps, which I used to generate a dense point cloud as the ground truth/reference 3D geometry of the EV1 neighbourhood. I then trained a 3D Gaussian Splatting model on these first 50 images. Afterwards, the Gaussian positional means were extracted as a new 3DGS densified 3D point cloud resulting in 1856968 points, starting from a sparse point cloud of 24740 points, a near 10x densification. The positional means I extracted as 3DGS densified point clouds are visualized in Figure 3.2(rasterized at Gaussian scale = 10^{-3}). In comparison, the MVS densified point cloud resulted in 2528969 points. The MVS densification results are visualized in Figure 3.6.

The initial sparse point cloud and the 3DGS densified point clouds are aligned with each other. However, the MVS densified point cloud which I take to be the reference point cloud was offset from both by a rotation, a translation, a scale factor and non-affine deformations far from the origin. This is visible in the last row of Figure 3.6, but even more obvious as the view is extended further. As such, I cropped all three point clouds and performed point cloud registration to align the three-point clouds, correcting for similarity transformations. I used the iterative closest point algorithm (ICP) [99] to register both the initial and the 3DGS densified point clouds to the MVS densified point cloud. This process aligned all three point clouds with translations and rotations, but there is a slight non-affine deformation as the distance from origin and height increases post-registration,

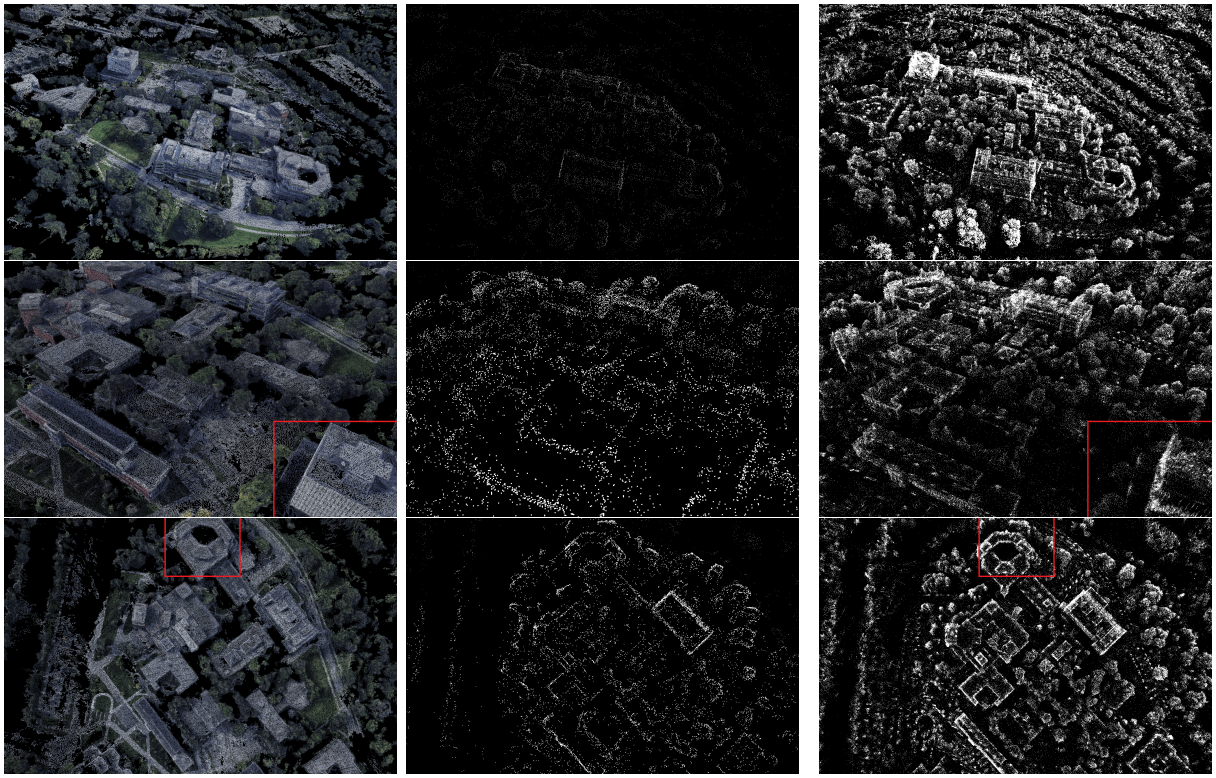


Figure 3.6: Comparison of 3DGS point cloud densification of the EV-1 neighbourhood at the University of Waterloo against initial and MVS densified point clouds. **Left**: Multi-view stereo depth and normal maps fusion as ground truth (coloured) dense point cloud. **Middle**: Initial sparse point cloud. **Right**: 3DGS densified point cloud. Areas with obvious discrepancies/misalignment are highlighted in red rectangles (aligned pixel-wise).

as can be seen in Figure 3.7.

The cropping resulted in 12,773, 244,849, and 1,270,820 points for the sparse, 3DGS densified, and MVS densified point clouds, respectively. The sparse and MVS densified point clouds were denser at the centre of the scene, whereas the 3DGS densified point cloud exhibited a more uniform density. Consequently, cropping reduced the points in the sparse and MVS densified clouds by roughly a factor of ~ 2, compared to ~ 7.5 for the 3DGS densified cloud. The generated point clouds were evaluated against the dense

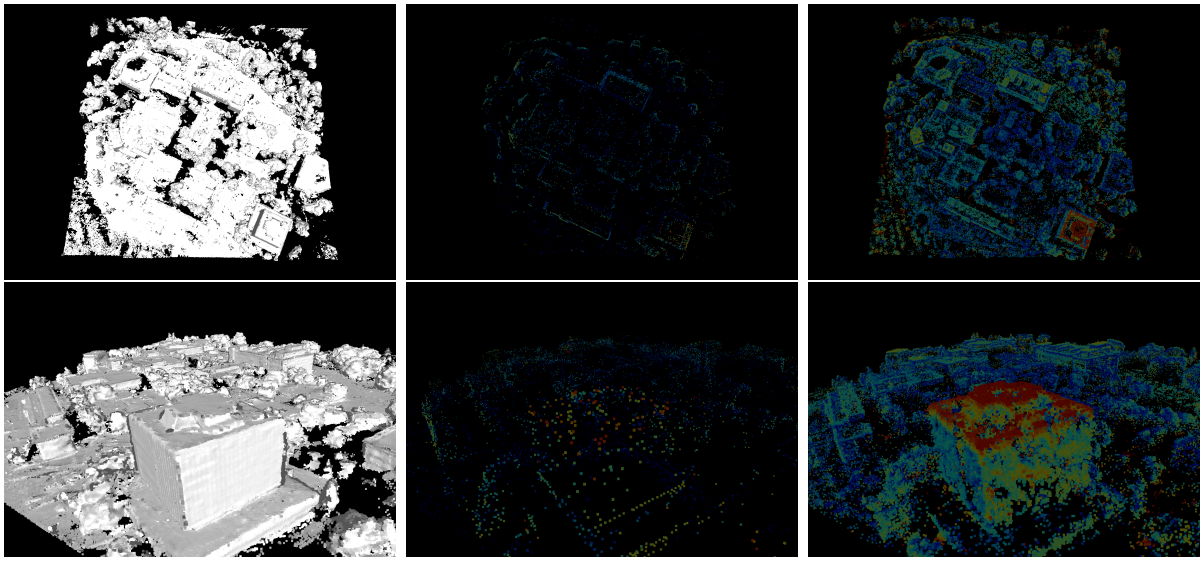


Figure 3.7: Post cropping and registration visualization of reference point cloud and deviation in local Hausdorff distance from reference. Blue = small deviation, red = large deviation. **Left**: Reference MVS densified point cloud, meshed using Poisson surface reconstruction [98]. **Middle**: Sparse initial point cloud coloured by Hausdorff distance from reference mesh. **Right**: 3DGS densified point cloud coloured by Hausdorff distance from reference mesh.

MVS reference using D1 (point-to-point) and D2 (point-to-surface) mean squared error (MSE), Hausdorff distance, and Chamfer distance, with results shown in Table 3.4. While the 3DGS densified cloud showed marginally higher D1 and D2 MSE than the sparse cloud, these metrics do not account for density differences, focusing instead on outliers and noise. Hausdorff and Chamfer distance better captured the variations in point distribution, revealing much closer agreement between the 3DGS densified and reference MVS clouds compared to the sparse cloud. This finding aligns with the visual inspection in Figure 3.6. Additionally, Figure 3.7 highlights local Hausdorff distances, showcasing non-affine distortions between the reference MVS cloud and the other point clouds.

Table 3.4: Comparison of initial sparse point cloud and 3DGS densified point cloud vs. MVS densified point cloud as reference.

	Sparse	3DGS Densified
Points	24740	1856968
Points post-cropping	12773	244849
D1 MSE↓	7.63×10^{-3}	8.15×10^{-3}
D2 MSE↓	6.88×10^{-3}	7.30×10^{-3}
Haussdorff distance↓	8.75×10^{-1}	3.75×10^{-1}
Chamfer distance↓	2.55×10^{-2}	1.62×10^{-2}

3.3.4 Experiment with Google Earth Studio's Simulated Illumination



Figure 3.8: Effect of sun illumination condition on novel view synthesis with high tilt angle on the test set. **Left**: Sunny clear sky at noon ground truth (**top**) vs. synthesized view (**bottom**). **Right**: Midnight illumination ground truth (**top**) vs. synthesized view (**bottom**).

Google Earth Studio allows for the collection of images under simulated illumination

changes from the sun's position (time of day) and the sky condition. To test the effect of changing sun direction and sunlight intensity I performed additional experiments in the region of study using a new camera path at a height of 450 m with a tilt of 78°, where 0° corresponds to top-down view, and 90° corresponds to a horizontal camera orientation. A large tilt angle is used to observe view angle-based effect of sun illumination. I trained a 3DGS model and benchmarked view synthesis and 3D point cloud reconstruction results for five different sky conditions and times of day. The novel view synthesis results are shown in Table 3.5.

Table 3.5: Performance Metrics for Different Sky and Sunlight Conditions

Sky/Sunlight	(Train) PSNR↑	Test (PSNR↑	SSIM↑	LPIPS↓)
Bright-Uniform	32.6	19.0	0.544	0.290
Sunny-Noon	32.6	19.5	0.624	**0.277**
Overcast-Noon	32.6	22.4	0.622	0.326
Midnight	33.7	22.6	0.522	0.525
24H Timelapse	32.4	**22.6**	**0.628**	0.326

COLMAP struggled under poor lighting conditions, and returned camera poses with larger errors under the Midnight lighting condition. The novel view synthesis results show that the training has likely converged. However, the rendered images from the testing views are uniformly lower quality than the previous experiments. This is likely caused by the large tilt angles resulting in the presence of floater Gaussians. Each illumination

Table 3.6: Point Cloud Distance Metrics Comparison Against Reference Point Cloud (Bright-Uniform).

Sky/Sunlight	D1 MSE	Chamfer Distance	Hausdorff Distance
Sunny-Noon	**.454**	**.410**	**3.12**
Overcast-Noon	.617	.673	14.4
Midnight	10.5	3.21	23.1
24H Timelapse	1.78	1.47	15.2

condition was tested with its own ground truth images for PSNR, SSIM, and LPIPS. Changing sky conditions to darker and more difficult illumination conditions (such as the *24-hour time-lapse*) negatively affects view synthesis results. However, the results are not immediately obvious according to the test scores and should be observed qualitatively. Poor lighting seems to hide visual artifacts, which strongly improves PSNR. SSIM and LPIPS are more reliable measures of visual quality. See Figure 3.8, which shows that the *Midnight* illumination has much poorer test set view synthesis visual quality compared to *Sunny Noon*, despite having a higher PSNR score.

Using the point cloud generated under the *Bright-Uniform* illumination, I performed various full-reference point cloud comparisons using the metrics of Section 3.3.3. I did not mesh the resulting point clouds and therefore did not calculate a point-to-surface mean squared error (D2 MSE). The mesh from the previous section is from COLMAP-MVS, a method which was not used for this set of tests. The results are shown in Table 3.6. There is a direct correlation between lighting similarity and point cloud geometric deviation. There is an especially large deviation in the *Midnight* lighting scene, which is likely due to the aforementioned camera pose errors in the COLMAP initialization, causing large geometric distortions near the edges of the scene.

3.3.5 Discussions

Surprisingly, 2DGS+ was not shown to achieve superior view-synthesis results compared to the more primitive 3DGS. This could be due to two factors. Firstly, the design choice to use 400+ images arranged in multi-level concentric circles provides extensive scene coverage for neighbourhoods of interest. This de-emphasizes particular strengths of 2DGS+ in floater reduction, better densification and pruning, and better few-

shot capabilities. Secondly, 2DGS+ may be "over-engineered" compared to 3DGS. By having a much higher number of hyperparameters, the method could need much more careful tuning to show advantages in scenarios where both methods perform well.

Google Earth Studio produces composite images and images rendered from 3D models, constructed using remote sensing images of a variety of governmental and commercial sources including Landsat, Copernicus, Airbus, NOAA, U.S. Navy, USGS, Maxar, during different flights. This can be both an advantage and a disadvantage. Low-altitude images using far-from-vertical/off-nadir point-of-views rely on Google Earth Engine's own 3D models, which are limited in detail compared to real remote sensing images. On the other hand, the variety of data sources benefits the robustness of the 3D Gaussian Splatting model, which has been trained on images from different sensors in different photometric and radiometric conditions. The disadvantages are also counterbalanced by the ease at which Google Earth Studio allows for the creation of multi-scale datasets with spiralling camera paths suited for large-scale 3D scenes centred around some neighbourhood of interest in a city. Some effects of image stitching can be observed in the images retrieved from Google Earth Studio in Figure 3.3. Without access to the initial data source prior to the stitching, standard methods for correcting these effects cannot be applied. However, using diffusion methods to correct these effects could be a potential future research direction. The simulated lighting condition experiment showed that there is an inverse correlation between the presence of floaters and the quality of lighting. This effect should also be investigated further.

When recovering 3D geometry (as 3D point cloud) from both the SfM preprocessing, the post-3DGS densified point cloud, and even the MVS densified dense point cloud, a mild to strong presence of noise can be noticed, which should be addressed in future 3DGS research. In this 3D reconstruction and densification experiments, I used the MVS densified

point cloud as ground truth, even though it was also constructed from 2D images. Despite recovering good quality dense 3D surfaces, the MVS densified point cloud was offset from both the initial sparse point cloud and the 3DGS densified point cloud with a non-linear transformation, which should be investigated further.

The lack of geometry ground truth in these experiments is worth further investigation. We can consider the Google back-end 3D models as ground truth given our data source, and these ground truths may have a bias compared to real-life measurements. In the future, for geometry recovery benchmarks, a scanned point cloud such as from a LiDAR source would be more accurate as ground truth. A future project could be to use a scanned point cloud as ground truth and to register and georeference both MVS and 3DGS densified point clouds to a scanned point cloud to properly study the geometry of these densifications, as well as enable further mapping and GIS applications. The memory requirement for the COLMAP MVS densification was larger than that of the 3DGS densification and was a reason behind performing the densification experiment on a smaller scale using fewer images. Despite these concerns and the fact that 3DGS was not built as a 3D geometry extraction tool, it is reasonably able to recover scene geometry through densification and optimization of the Gaussian positions.

Additionally, we should note that all Gaussian Splatting models are trained partially via supervising SSIM-difference. Simultaneously, SSIM is used as a quality assessment metric. This is generally considered poor practice but has been established as standard in the field.

Significant Figures

In this section and for the rest of this thesis, view synthesis and 3D geometry assessment results are treated with 3 significant figures. Both three and four significant figures are well-accepted in the field. Leading zeros after the decimal place are treated as significant zeros for LPIPS scores. Ideally, identical experiments would be run multiple times to obtain a sample standard deviation to help determine the number of significant digits in the measurements of the scores. By calculating the standard error of the mean, the last significant digit of the measurement of the mean can be matched with the first significant digit of the standard error of the mean. If we desire more rigour, we can even perform hypothesis testing for each suggested improvement to measure whether the score improvements with the proposed improvements are statistically significant. However, this is beyond the scope of these experiments and is against the field's conventions. There are also constraints of time and computational resources for repeated training and benchmarking.

3.4 Chapter Summary

In this chapter, the standard 3D Gaussian Splatting method and 2DGS+ were compared against NeRF methods on a large-scale urban scene reconstruction dataset spanning 10 cities as a benchmark. Then, an aerial off-nadir dataset of the region of study was captured by leveraging Google Earth imagery from Google Earth Studio. The scene was photorealistically rendered, and its geometry was captured using 3DGS. In the region of study, an investigation of the 3D reconstruction capability of 3DGS was performed via 3D point cloud densification and extraction, with the densification compared and visualized against Multi-View-Stereo dense reconstruction. Both a symmetric transformation mis-

alignment and a non-linear deformation were identified, with the misalignment removed through point cloud registration and the deformation quantified and visualized between the Multi-View-Stereo densified point cloud and the 3DGS densified point cloud. It was also shown that in terms of view synthesis quality, the basic 3DGS method is not necessarily inferior to highly engineered Gaussian Splatting methods (such as 2DGS+) if the training has already converged due to overly abundant scene coverage. Whether 3DGS, 2DGS, 2DGS+, or others, Gaussian Splatting methods show promise for replacing classical photogrammetry for large-scale urban modelling and could help contribute towards the next generation of Urban Digital Twins.

Chapter 4

Gaussian Building Mesh: Extract a Building's 3D Mesh with Google Earth and Gaussian Splatting

Focusing on creating precise 3D mesh models at the individual building scale for downstream digital twin systems use, I tested Gaussian Splatting-based methods for 3D mesh extraction. I then developed my own framework for 3D building mesh extraction, addressing the perceived shortcomings of existing methods, and improving accuracy, robustness and training speed.

This chapter is comprised of my research preprint [100] eponymous with the chapter title. Section 4.1 is the chapter's introduction which motivates and briefs the research. Section 4.2 presents additional background material for the chapter. Section 4.3 is the methodology section which describes the Gaussian Building Mesh framework. Section 4.5 presents and discusses the chapter's results. Section 4.6 is the chapter's conclusion.

4.1 Introduction

The extraction of 3D building models from remote sensing images has long been an active research topic, with applications ranging from urban planning, disaster management, environmental monitoring, telecommunications, construction, digital media, and many more. In remote sensing, the standard way to extract 3D information from 2D images is by using photogrammetry, which involves identifying key points in multi-view images of a scene and then triangulating and registering these key points into a cohesive 3D point cloud of the scene [36, 37]. Recent innovations in learning-based 3D rendering approaches, namely Gaussian Splatting, have opened up new possibilities in learning both accurate 3D lighting and 3D geometry from 2D images, attracting much research interest [6, 101]. Additionally, advances [102, 103] in deep learning image processing have greatly improved the capabilities of extracting individual objects from images.

Leveraging Google Earth Studio [31], and inspired by GS2Mesh [65], I propose a 3D building mesh extraction pipeline capable of extracting the 3D mesh of a building given its proper name, address, postal code, or geographical coordinates. The contributions are as follows:

- I combine Google Earth Studio, the Segment Anything Model, and a Gaussian Splatting mesh extraction model into a 3D building mesh extraction pipeline. This allows for the generation of the 3D mesh of a building from its name, address, postal code, or latitude/longitude coordinates.

- I leverage Segment Anything Model-2 [104] with GroundingDINO [103] for 3D consistent building masking with either text or click-based masking input modalities, improving on GS2Mesh-based mask extraction. I add mask re-prompting and refinement as contributions.

- I propose a mask refinement method based on morphological operations and the Ramer-Douglas-Peucker algorithm.

- I use an improved version of 2DGS as the Gaussian Splatting model and generate the 3D coloured mesh of a building of interest using masked Truncated Signed Distance Function (TSDF) Fusion/Integration. During the meshing process, I make many adjustments, such as filtering and smoothing the depth maps and tuning the hyperparameters during both Gaussian Splatting training and TSDF meshing.

GS2Mesh [65] combines out-of-the-box object segmentation and 3DGS to generate 3D meshes of objects regardless of the background. To our knowledge, it is the only method similar to the method presented in this chapter and is the inspiration for his method. Compared to GS2Mesh, I use a state-of-the-art (not-yet-published) fork of 2DGS [84] instead of the original 3DGS method for 3D representation learning. This allows us to obtain accurate depth maps without using deep learning-based stereo depth reconstruction, rendering GS2Mesh's DLNR stereo depth reconstruction module obsolete, which I remove. This also allows for another significant change: because I no longer need a background as context for DLNR, I move the automated building masking step to before Gaussian splatting training, i.e., I train Gaussian splatting on masked images instead of full images. This increases the training speed by a factor of $\sim$5, depending on the portion of the region masked as background. I also add mask re-prompting mask refinements based on morphological operations and contour simplification, which correct for SAM2 masking errors, as well as depth smoothing, which helps with TSDF meshing. These improvements greatly increased the mesh extraction accuracy and efficiency. The pipeline is shown in Figure 4.1 and detailed in Section 4.3.

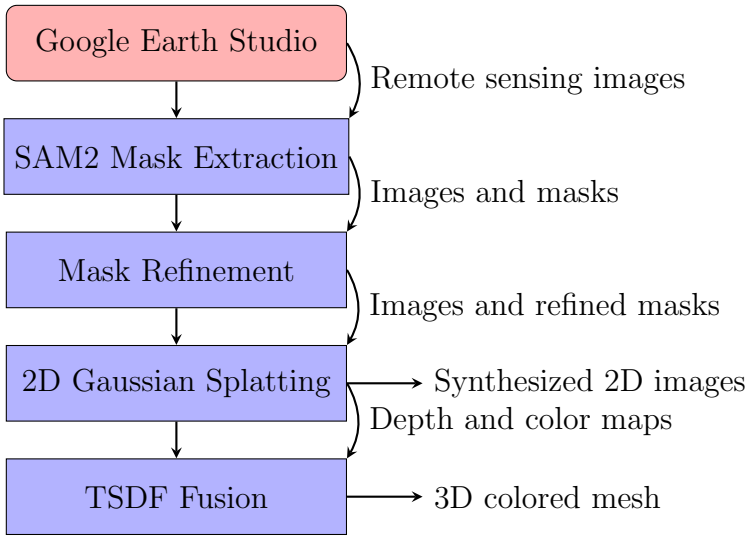


Figure 4.1: Flow chart of the Gaussian Building Mesh pipeline. The output data modality at each step is denoted on the right. Processes and modules are boxed in blue.

4.2 Additional Background

4.2.1 SAM2 and Grounding DINO

Segment Anything Model (SAM) [102] is an out-of-the-box image segmentation model pretrained on a massive billion-image dataset. It is capable of segmenting most objects given point-based or bounding box priors, without requiring further training from the user. Users can provide point-click or bounding box prompts to identify the object(s) of interest, and SAM will return segmentation masks and associated scores. However, a key limitation of SAM is that, when applied to a scene viewed from multiple images, the individual masks generated are not necessarily consistent with each other [104]. This inconsistency limits SAM's effectiveness in segmenting video data, which requires temporal consistency, and multi-view data, which requires 3D consistency.

SAM2 [104], released in August 2024, addresses this issue by introducing consistent

video segmentation that maintains 3D and temporal consistency through the use of memory attention. Since I aim to extract the mesh of an individual building rather than an entire neighbourhood, segmentation masks are crucial.

GroundingDINO [103] is a pretrained open-set object detector capable of extracting object bounding boxes in images from natural language prompts without requiring additional training from the user. GroundingDINO can be combined with SAM/SAM2 to enable text-based object segmentation. The process involves first generating a bounding box from the text description and then using the bounding box to prompt SAM/SAM2. This combination of GroundingDINO and SAM/SAM2 is known as the Grounded-SAM pipeline [105], which is available as an open-source library.

4.2.2 GS2Mesh

GS2Mesh [65] is a Gaussian Splatting-based 3D reconstruction pipeline, outperforming concurrent and competing methods such as SuGAR [106], 2DGS [26], and GOF [64] on the DTU dataset [107] benchmark.

- GS2Mesh learns and stores the scene in a standard 3DGS model.
- The trained 3DGS model is then used to generate a stereo pair for each input image. Each stereo pair is used to generate a depth image.
- Grounded-SAM is used to generate multi-view masks to mask out the background for mesh extraction.
- A pre-trained depth from stereo model DLNR is used to generate depth maps for each stereo pair.

- The entire ensemble of depth images are integrated into a mesh using the Truncated Signed Distance Function fusion (TSDF) algorithm [12] with the Marching-Cubes algorithm [14].

More specifically, a standard 3DGS model is trained from the input images. The 3DGS model is then used to generate a stereo pairs for each training image's camera pose. For each pair, the left-image is generated with the same camera pose as the training image, and the right-image is generated with a small shift $[b, 0, 0]$ to the right. Since Gaussian Splatting models perform best near training poses, this method ensures visual high quality in the generated stereo image pair.

SAM2 is used to generate segmentation masks for the objects or these stereo-image pairs. In the GS2Mesh paper, which was published before SAM2, the authors addressed the 3D consistency issue by projecting the initial mask onto other frames, sampling new points within the projected mask as SAM prompts, and creating a new SAM mask from these prompts for each frame. The GS2Mesh codebase has since been updated to use SAM2 for 3D geometry-consistent mask generation.

From these stereo pairs, DLNR [108], pretrained on the Middlebury dataset [109], is used for depth extraction from stereo images. To improve the quality of the reconstructions, multiple masks are applied to the stereo model's output to filter out regions visible to only one camera and to discard depth estimates outside the valid range.

Following depth extraction, a standard Open3D [110] implementation of the TSDF algorithm initializes and populates a voxel grid with the scene geometry. The voxel representation of the scene is populated with the signed distance to the nearest scene surface, integrated from the depth images generated by DLNR. The marching cubes algorithm then assigns each cube's vertices in the voxel grid to be inside or outside the nearest surface

based on the previously calculated TSDF values. Based on the 8-vertex configuration, a local surface is meshed for each cube, which is repeated across the entire voxel grid. This process generates a mesh from the voxel representation.

According to my preliminary tests, GS2Mesh often fails in remote sensing building extraction tasks due to issues in the mask generation module and the pretrained deep learning-based stereo depth map module. While GS2Mesh slightly outperforms 2DGS on the DTU benchmark, the two methods are fundamentally different.

4.3 Methodology

Leveraging Google Earth Studio and inspired by GS2Mesh, I created a 3D building mesh extraction pipeline capable of extracting the 3D mesh of a building given its location name, address, postal code, or geographic coordinates. As shown in Figure 4.1, the pipeline consists of the following steps:

1. **Multi-view remote-sensing image collection**: I leverage Google Earth Studio to collect multi-view images of a building of interest.

2. **Building mask extraction**: I extract multi-view consistent building masks for the building of interest in each of the images. I adapted existing GS2Mesh and SAM2 code to allow for text-based and click-based (point or box) user prompting to help SAM2 identify the building of interest in an initial image. I introduce re-prompting capabilities after mask propagation to address mask inconsistencies.

3. **Building mask refinement**: I refine SAM2 masks using morphological dilation, which slightly extends the mask outwards and fills in holes. I then simplify mask contours using the Ramer-Peuker-Douglas algorithm.

4. **Gaussian Splatting**: I train a 2DGS+ [84] model to learn building geometry and radiance.

5. **Mesh extraction**: I perform masked Truncated Signed Distance Function Fusion on smoothed depth maps to extract a 3D building mesh. I perform mesh refinement and mesh simplification using Open3D triangular mesh post-processing functionalities.

4.3.1 Google Earth Studio Dataset

I leverage Google Earth Studio to extract 12 scenes with buildings of interest. For each scene, I extract 31 frames/images over a 360° circular camera path centred around the building of interest. As part of Google Earth's functionality, I tested and used various methods to identify the buildings of interest, including addresses, postal codes, geographic coordinates, and building names.

4.3.2 Building Segmentation and Mesh Refinement

SAM2 provides a rough mask of the building. I noticed the following problems while inspecting the SAM2 masks and the 3D mesh extraction that resulted from directly using those masks.

1. **Poor geometry and holes**: I noticed that SAM2 had trouble with some buildings whose roof or wall colours might be confused with the background, resulting in incomplete masks with holes.

2. **Noise pixels and false positives**: SAM2 masks sometimes produced small false-positive pixels away from the object of interest. This occurred at times with multiple identical buildings close to each other.

3. **Poor mask boundary at building base**: SAM2 produced masks with poor geometry at the buildings' base, often resulting in incomplete masking and jagged mask boundaries. This resulted in 3D reconstructions with jagged bottoms, greatly affecting visual quality.

4. **Tight-fitting mask, sometimes incomplete**: SAM2 masks were often too tightly fitted onto buildings. This sometimes resulted in some outer parts of the buildings not being captured at certain viewpoints. This greatly affected 3DGS reconstruction quality.

To address these issues, I developed a mask refinement pipeline. I optionally morphologically erode the mask first, removing noise. Then, I grow the mask using morphological dilation to fill in any holes and expand incomplete masks. I then extract the mask's contour using Canny's algorithm and straighten/simplify the contour using the Ramer–Douglas–Peucker algorithm. The contour is then filled into a refined mask, addressing these issues.

User Re-Prompting

Like GS2Mesh, by using the Grounded-SAM pipeline with SAM2, I extract object masks without further training via user prompting with natural language text (when combined with GroundingDINO), bounding boxes, or point-clicks. However, preliminary testing showed that in a 360° video of a complex object, if the object self-occludes during rotation, the resulting masks can be inconsistent (see Figure 4.2). To solve this issue, I modified the GS2Mesh masking pipeline to add user re-prompting. I allow users to identify frames with errors and re-prompt the image at those frames. SAM2 re-propagates the user's new prompt throughout all frames, effectively solving these mask inconsistencies.

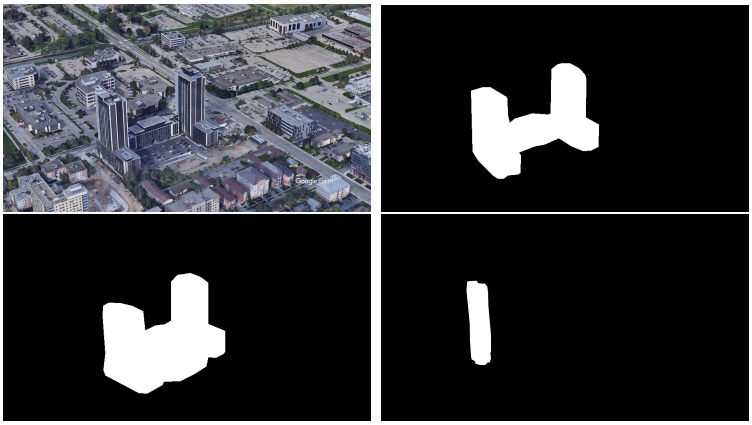


Figure 4.2: Example of SAM-2 mask inconsistencies. **Top Left**: 2D image of the building of interest. **Top right**: SAM-2 mask of the same frame. **Bottom-left**: SAM-2 mask in a later frame after some rotation with the parking lot masked in erroneously. **Bottom-right**: SAM-2 mask in a later frame, with the second tower missing.

Morphological Dilation and Erosion

The morphological dilation $\oplus$ of an image I by a filter F is defined by

$$I \oplus F = \bigcup_{p \in F} I_p, \tag{4.1}$$

where I_p denotes the image I shifted by pixel $p = (p_x, p_y)$. Roughly speaking, this grows the image by sliding the filter along the pixels of the original image. This operation can be performed iteratively to further extend the image. At each step, the output image of the previous iteration is dilated again using the same filter. I apply two iterations of morphological dilation with a square filter of size (10,10).

Before the morphological dilation, I add an optional morphological erosion step (with the same parameters), which is the inverse operation of morphological dilation. That is, morphological erosion shrinks the mask according to the filter. This step serves two pur-

poses. Firstly, depending on the outline of the building, we may not want the dilated mask to extend past the building's true boundaries. Performing the erosion-dilation steps allows the mask to be roughly as tight-fitting as the initial SAM2 mask, while still morphologically filling in holes in the mask. Secondly, at times, SAM2 masks introduce small false-positive pixels away from the central object in the form of noise pixels. This step allows the pipeline to erode away this type of noise.

Canny Edge Extraction

Canny edge extraction is a well-known classical algorithm for extracting lines using the image's gradient. I briefly outline it here and refer readers to the original paper for details [111]. I use a Gaussian filter of $\sigma = 2$, a low threshold of 0, and a high threshold of 1 to extract a contour which I refine with the Ramer-Douglas-Peucker algorithm.

Ramer–Douglas–Peucker Contour Refinement

The Ramer–Douglas–Peucker algorithm (RDP) [112] is an iterative contour simplification algorithm that recursively reduces the number of points in the contour. The algorithm is as follows:

Given:

A curve defined by a sequence of ordered points:

$$P = \{P_1, P_2, \ldots, P_n\}, \quad \text{where } P_i = (x_i, y_i) \tag{4.2}$$

and a threshold parameter:

$$\epsilon > 0 \tag{4.3}$$

Algorithm:

1. Start with endpoints P_1 and P_n. Define the line segment:

$$\overline{P_1 P_n}$$

2. For each point $P_i = (x_i, y_i)$ with $i \in [2, n-1]$, compute the perpendicular distance d_i to the line $\overline{P_1 P_n}$:

$$d_i = \frac{|(x_n - x_1)(y_1 - y_i) - (x_1 - x_i)(y_n - y_1)|}{\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}} \tag{4.4}$$

3. Determine the maximum distance:

$$d_{\max} = \max_{i=2,\ldots,n-1} d_i \tag{4.5}$$

4. If $d_{\max} > \varepsilon$, let P_k be the point at which the maximum distance occurs. Recursively apply the algorithm to the subsets:

$$\{P_1, P_2, \ldots, P_k\} \quad \text{and} \quad \{P_k, P_{k+1}, \ldots, P_n\} \tag{4.6}$$

5. If $d_{\max} \leq \varepsilon$, approximate the entire set by the line segment $\overline{P_1 P_n}$, and keep only P_1 and P_n.

Output:

A reduced set of points $P' \subseteq P$, forming a piecewise linear approximation within tolerance ϵ.

I used $\epsilon = \frac{p}{500}$, as the ratio of the perimeter p of the contour being simplified, allowing longer perimeter contours to retain more complexity. The resulting building contours became linear interpolations of approximately 20-40 points. The contour is then filled to create the refined mask which addresses the aforementioned issues.

4.3.3 2DGS+

I use a state-of-the-art implementation of Gaussian Splatting built on the foundations of 2DGS. This implementation, which is (unofficially) dubbed 2DGS+ [84] in this thesis, (detailed in Section 3.2.4), is an unpublished fork of the 2DGS repository which combines many recent advances in Gaussian Splatting, improving 2DGS with ideas from AbsGS [85], PixelGS [86], TrimGS [87], AtomGS [88], GaussianPro [90], and Taming-3DGS [91].

In these experiments, the Gaussian Splatting representation generated from 2DGS+ has fewer floaters (free-floating Gaussians in mid-air), is smoother, and produces better depth maps, resulting in significantly higher-quality meshes during TSDF fusion meshing (see Section 4.5.4). I tuned the hyperparameters for 3D mesh extraction quality (not novel view synthesis quality). I used the progressive training modality from GaussianPro, densifying the Gaussians until the 25,000th iteration. I used $\lambda_{depth} = 0.2$ and $\lambda_{normal} = 0.1$.

4.3.4 TSDF Fusion and Mesh Extraction

Like GS2Mesh and 2DGS, I used the Open3D implementation [110] of the TSDF Fusion algorithm with Marching Cubes to convert depth maps into coloured 3D meshes. I adapted the 2DGS mesh extraction module, adding a depth map smoothing step and tuning the hyperparameters. I used the 2DGS+ implementation [26] of 2DGS with its bounded mesh extraction module.

In early experiments, I noticed non-smoothness in the depth maps of certain complex buildings, which resulted in poor-quality mesh reconstructions. In these cases, applying a simple Gaussian blur filter to the depth maps before TSDF fusion greatly improved the mesh reconstruction quality.

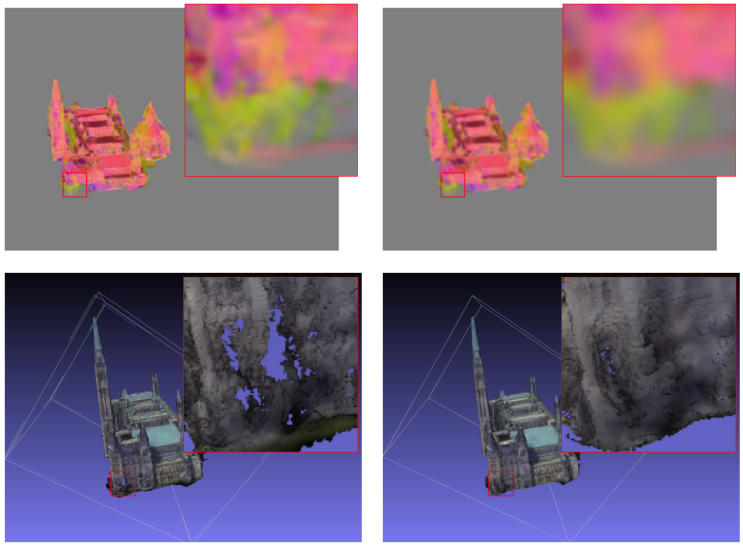


Figure 4.3: Example of the effects of depth blurring. **Top-left**: Raw depth map. **Top-right**: Smoothed depth map. **Bottom-left**: 3D Mesh from raw depth map. **Bottom-right**: 3D Mesh from the smoothed depth map.

After mesh extraction, like GS2Mesh and 2DGS, I used Open3D to clean the mesh by computing mesh clusters, removing unreferenced vertices, and degenerate triangles. Unlike GS2Mesh and 2DGS, I discarded all but the largest mesh cluster.

4.4 Metrics

The novel view synthesis metrics used in this chapter are introduced in Section 2.6.

In addition, a full reference 3D-SSIM metric [113] is used to compare a 360 ° view rendered video of a generated mesh vs. a ground truth video. The metric uses Eq. (2.52). There are other full-reference 3D geometry and visual quality metrics that compare 3D models to other 3D models. However, I lacked ground truth 3D models for the meshed buildings. I only had access to ground truth 2D images. This is the main reason for using the video-based 3D-SSIM for mesh quality comparison. Both the average 3D-SSIM across the entire video and the minimum 3D-SSIM across video frames are provided.

4.5 Experiments and Discussion

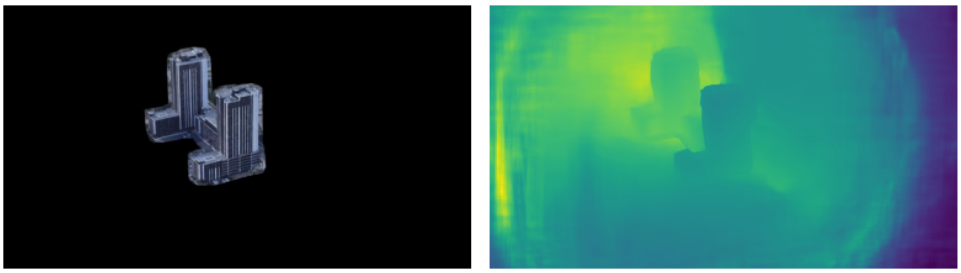


Figure 4.4: Example of DLNR failure case. **Left**: 2D image of the building of interest with background masked out. **Right**: DLNR stereo depth reconstruction from building with background masked out.

4.5.1 Preliminary Experiments

The preliminary experiments on 3D building mesh extraction from remote sensing images showed that the GS2Mesh 3D reconstruction pipeline, which uses DLNR [108], a pretrained neural-network-based stereo depth reconstruction and depth map fusion, only matched the quality of 2DGS depth map fusion in the best-case scenario. More often than not, it strug-

gled in the 3D reconstruction of buildings due to two key issues: poor mask quality and poor stereo depth estimation. DLNR occasionally struggled with stereo depth reconstruction, likely because it was not trained on a remote sensing image dataset and had difficulty interpreting large-scale aerial scenes. This issue becomes particularly pronounced when stereo-depth extraction is applied after masking, as the neural network loses the context of the background. In Figure 4.4, we observe that DLNR fails completely when performing depth extraction on a building with the background masked out. This motivated us to seek alternative solutions for the 3DGS-DLNR pipeline components in GS2Mesh.

Preliminary experiments with 2DGS+ showed that depth maps of certain complex buildings exhibited non-smoothness, resulting in poor-quality 3D meshes. I addressed this issue with the simple solution of smoothing the depth map using a Gaussian filter. An example is shown in Figure 4.3. After applying the filter, the holes in the zoomed-in region were almost completely filled. This particular issue arose due to poor visual coverage of the zoomed-in region in the initial training data.

4.5.2 Mask Refinement

SAM2 masks sometimes included false-positive pixels away from the object of interest. In Figure 4.5 (top row), this issue is visible in the left-most subfigure. This is likely caused by false-positive segmentation due to the presence of a neighbouring building's rooftop with a similar shape and colour in the original full image. The mask refinement algorithm successfully removed the false-positive pixels and refined the mask contour.

SAM2 also occasionally produced masks with holes or false negatives. We can observe that this occurred more frequently when part of the building exhibited significantly different colouration from the rest of the structure. An example is shown in Figure 4.5 (bottom

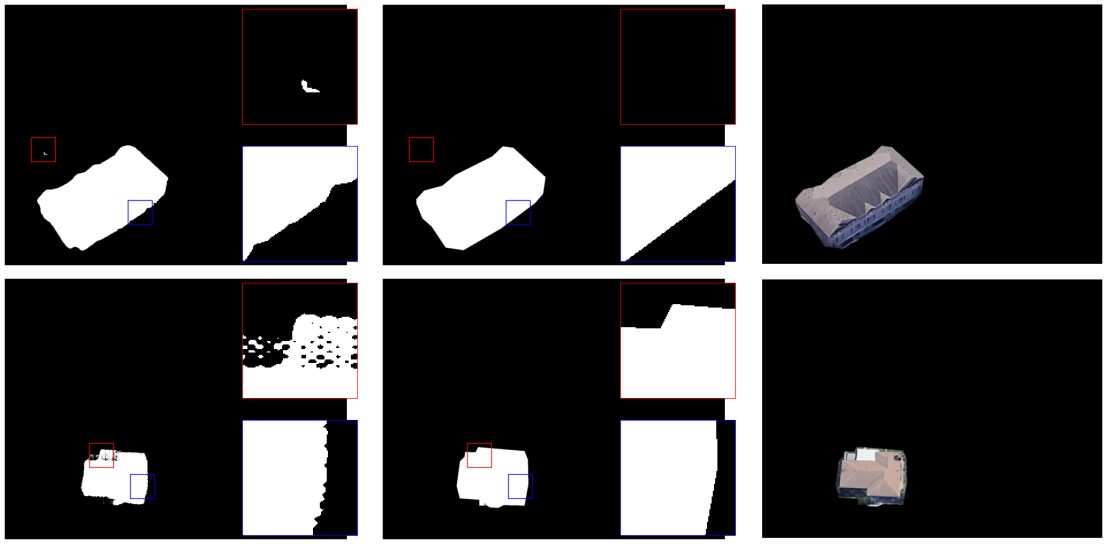


Figure 4.5: Example of mask refinement. Noise removal**(top)** and hole filling **(bottom)**: **Left**: Raw SAM-2 Mask. **Middle**: Refined Mask. **Right**: masked Image. Regions of interest have been highlighted. A red box indicates a region of false positives or false negative pixels, and a blue box indicates a region where the contour was refined and straightened.

row), where the garage roof has a different colour. The mask refinement algorithm was able to complete the mask and refine the mask contour.

4.5.3 View Synthesis

Compared to GS2Mesh, I replaced the Gaussian Splatting module in the pipeline, transitioning from the original implementation of 3DGS to 2DGS+. As such, I compare the training set novel view synthesis results of 3DGS and 2DGS+ across our 7 scenes. I did not use the standard MipNeRF-360 convention of a training/testing split (leaving out one out of every eight images). Instead, I trained on the entirety of the Google Earth Studio footage, as I wanted smooth coverage of the 360° camera rotation for 3D reconstruction

purposes. The novel view synthesis scores on the training set are provided in Table 4.1.

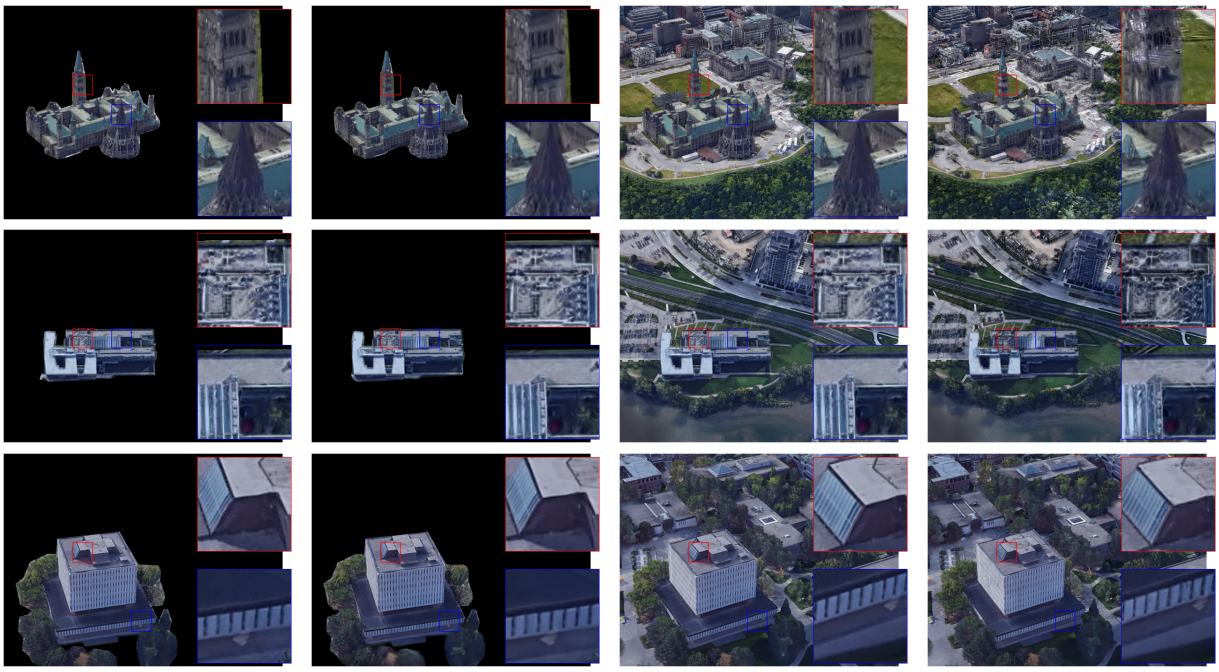


Figure 4.6: Visualization of 2D novel view synthesis results. **Left to Right: 2DGS+ Ground Truth; 2DGS+ Sythesized Image; 3DGS Ground Truth; 3DGS Synthesized Image**. For all cases including those not shown, the results were nearly visually indistinguishable from ground truth by human eyes, except for 3DGS trained on the Canada Parliament scene (**top right**).

For 2DGS+ in my pipeline, I trained the model on the masked building without the background. The model's synthesized images were then compared to images of the building with the background masked out. On the other hand, in the GS2Mesh pipeline, the 3DGS training occurs before masking and is conducted on the entire image, including the background. Its 3DGS module generates images with the entire background, and the results are compared with unmasked ground truth images.

On average, I found that 2DGS+ produced higher SSIM and LPIPS scores, whereas 3DGS achieved higher PSNR scores. SSIM and LPIPS are typically better indicators

of human visual perception than PSNR. Nonetheless, both models achieved novel view synthesis results that were nearly indistinguishable from ground truth images, with the exception of 3DGS struggling on the Canada Parliament scene, as shown in the top row of Figure 4.6 (see the central tower highlighted in the top-right zoom-in). This is also reflected in Table 4.1, where the 3DGS scores for the Canada Parliament scene are significantly lower than for other scenes.

Table 4.1: Comparison of 2D Novel View Synthesis Metrics Between Gaussian Splatting Modules

	2DGS+ (in Our Pipeline)			3DGS (in GS2Mesh)		
Building	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓
Icon	.994	37.6	.010	.987	37.2	.016
Canada Parliament	.989	35.1	.016	.932	29.8	.108
CN Tower	.998	40.0	.004	.995	42.6	.009
Laurel Heights	.995	38.1	.009	.990	38.0	.012
Perimeter Institute	.995	39.4	.008	.994	43.9	.015
Dana Porter Library	.975	32.2	.033	.993	38.4	.011
Townhouse	.995	36.8	.008	.992	39.7	.012
Buckingham	.992	32.7	.011	.971	34.9	.045
Whitehouse	.998	4.00	.003	.984	35.5	.035
Westminster	.987	33.8	.020	.967	33.3	.091
Pyramid	.997	40.5	.004	.986	39.6	.029
BathChancellor	.998	42.8	.0027	.981	35.3	.031
Average	**.993**	**37.4**	**.011**	.981	37.3	.035

4.5.4 3D Building Mesh Extraction

As shown in Table 4.2, my building extraction results are superior to GS2Mesh, with the exception of the Townhouse scene, where both methods scored similarly, within ∼0.0005 3D-SSIM. The results are somewhat skewed by the Canada Parliament scene (first column

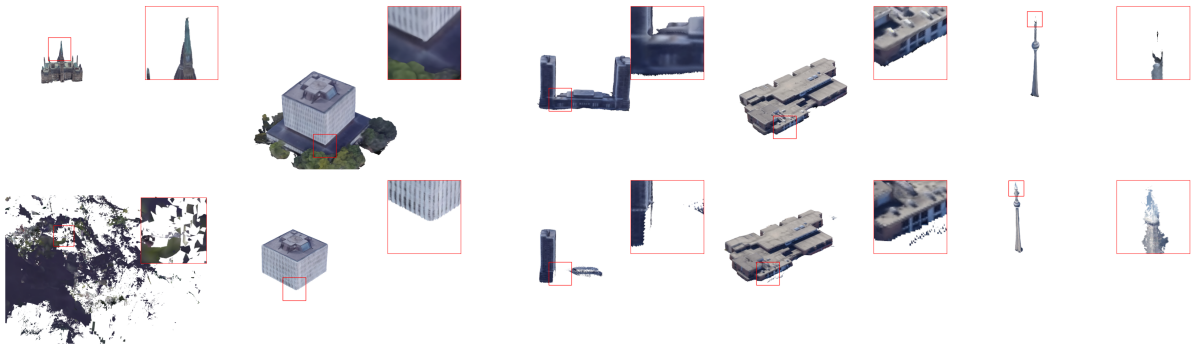


Figure 4.7: Visualization of 3D coloured mesh results. **Left to Right**: Canada Parliament; Dana Porter Library; ICON; Laurel Heights; CN Tower. **Top row** are our results, **Bottom row** are GS2Mesh results.

Table 4.2: Comparison of 3D-SSIM Scores Between our Pipeline and GS2Mesh

	GBM (Ours) 3D-SSIM ↑		GS2Mesh 3D-SSIM ↑	
Building	Average (31 frames)	Minimum	Average (31 frames)	Minimum
Icon	.929	.916	.836	.800
Canada Parliament	.854	.846	.146	.070
CN Tower	.984	.977	.940	.928
Laurel Heights	.957	.953	.943	.941
Perimeter Institute	.975	.973	.948	.943
Data Porter Library	.821	.808	.705	.686
Townhouse	.978	.975	.979	.976
Buckingham	.914	.910	.836	.817
Whitehouse	.910	.907	.430	.362
Westminster	.869	.866	.879	.861
Pyramid	.956	.949	.784	.690
BathChancellor	.973	.971	.807	.761
Average	**.927**	**.921**	.769	.736

of Figure 4.7), where GS2Mesh 3D mesh reconstruction failed completely due to the significant presence of floaters during the training of its 3DGS module. This failure caused its DLNR depth reconstruction module to produce completely inaccurate depth maps. The poor mesh quality resulted in extremely low 3D-SSIM scores, as reflected in Table 4.2.

The resulting mesh was completely unrecognizable. The poor performance of the 3DGS module of GS2Mesh for this scene is also reflected in its PSNR score in Table 4.1, which is approximately 10 dB lower than for other scenes. GS2Mesh also partially failed on the ICON scene and Dana Porter Library scene, where the lack of user re-prompting caused the masking to be inconsistent across the initial training images, resulting in missing geometric features and incomplete meshing.

My method produced less noisy meshes overall and cleaner mesh boundaries. This improvement is reflected in the 3D-SSIM scores and can be seen in the fourth column of Figure 4.7. However, both GS2Mesh and my method struggled with sharp structural elements, such as the top of the CN Tower, as shown in the fifth column of Figure 4.7.

4.5.5 Ablation Studies

Table 4.3: Ablation Study on 2DGS+

	Train			Test			3DSSIM↑		Training Time↓
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	Mean	Min	
2DGS	34.0	.987	.021	28.5	.954	.045	.876	.865	0:11:24
2DGS+Trim	36.4	.992	.011	28.5	.955	.042	.882	.877	0:13:05
2DGS+Trim+Prog-Norm	36.7	.992	.013	28.4	.952	.043	.881	.876	0:12:03
2DGS+Trim+Prog+Norm	36.8	.992	.013	28.3	.951	.044	.884	.870	0:11:49
2DGS+Trim+Prog+EANorm*	36.6	.992	.013	28.4	.952	.042	.891	.887	0:15:10
2DGS+Trim+Fast+Norm	36.8	.993	.011	28.7	.955	.041	.882	.877	0:14:34
2DGS+Trim+Prog+EANorm+Appearance	36.7	.992	.013	28.5	.952	.043	.884	.879	0:15:21
2DGS+Trim+Prog+EANorm+Appearance+60k	38.2	.994	.010	28.2	.949	.045	.867	.863	0:30:26

In this section, I perform an ablation study on the Westminster Palace scene, comparing the improvements of 2DGS+ to 2DGS as a baseline. A mip-NeRF [38] style training/testing split was used. Both view synthesis and 3D mesh quality assessments are used. Additionally, training time was measured. Trim-Gaussian's [87] and PixelGS's [86] contribution-based Gaussian pruning is denoted as Trim. Progressive propagation from GaussianPro [90] is denoted as Prog. The standard 2DGS normal-based supervision is

denoted as Norm. The Edge-Aware Normal/Geometry Supervision from AtomGS [88] is denoted EANorm. The Appearance Network from GOF [64] is denoted as Appearance. Fast-SSIM from Taming-GS [91] is denoted as Fast. Doubled training iteration is denoted as 60k.

4.5.6 Discussion

In terms of experiments, for the Dana Porter Library scene (second column of Figure 4.7), without user re-prompting, the base of the building was masked inconsistently due to the presence of occluding trees, which resulted in a messy GS2Mesh reconstruction. Instead, for GS2Mesh training on this scene, I found that prompting for the large cubical structure without the first floor and the trees resulted in better reconstruction. My pipeline was able to adequately capture the building base along with the surrounding trees. For the ICON scene (third column of Figure 4.7), without re-prompting, the mask for the second tower and part of the connected structure was missing in some frames, resulting in an incomplete reconstruction for GS2Mesh. Both methods struggled with sharp, needle-like building structure elements, such as the top of the CN Tower. This was likely due to a combination of the difficulty in masking needle-like objects and the challenge of producing depth maps for these objects.

The GS2Mesh failure on the Canada Parliament scene was unexpected. The failure likely occurred in the 3DGS module, as a significant amount of floaters can be noticed (Gaussians floating in empty space). I did not attempt further fine-tuning of the 3DGS training, although this likely could have significantly helped with the scene reconstruction for GS2Mesh. These floaters were absent during the 2DGS+ stage in my pipeline, even without additional fine-tuning.

I believe further improvements can be made to the mesh boundary at the building base. Although my method's mesh boundaries improved, we can still notice irregularities and poor-quality meshing at the base of buildings. I suspect that this issue, along with the problem of meshing sharp structural elements, can be addressed by further improving building masks—a potential extension to this research. My improvements to SAM2 masking can also be used independently of the meshing pipeline in other applications of SAM2/GroundedSAM. This topic could constitute a separate line of research apart from Gaussian Splatting-based mesh extraction since SAM2 finds many other applications as a pretrained foundational segmentation model with text-prompting, SAM2 improvements can be tested for other downstream tasks in remote sensing, autonomous vehicle vision systems, and general instance segmentation from video. Future improvements can also be made in the Gaussian Splatting process, reducing noise and floaters drastically improve TSDF meshing results, and further improvements can still be made in this area.

As can be observed in the ablation studies, the initial TrimGS improvement on contribution-based Gaussian trimming results in a significant improvement over 2DGS. However, the other 2DGS+ improvements only provide marginal improvements, and at times even decreased certain scores. Doubling training iterations improved training view synthesis scores but decreased the testing scores. I denote 2DGS + Trim + Pro + EANorm as the standard 2DGS+ method, which is used throughout this thesis. This is due to its high scores and the lack of needing an additional small neural network. In a future experiment, the cause and mitigation efforts for the specific failure case of GS2Mesh in the Parliament of Canada scene should be investigated.

4.6 Chapter Summary

In this chapter, I proposed a 3D mesh extraction pipeline for buildings. By utilizing Google Earth Studio and integrating SAM2/GroundingDINO, 2DGS+, and my mask refinement algorithm, I developed a robust pipeline capable of extracting a coloured 3D mesh of a building using its address, postal code, geographic coordinates, or location name. This was benchmarked against GS2Mesh, which is to the best of my knowledge, the only comparable segmentation-based Gaussian Splatting meshing algorithm publicly available. The results demonstrate that my approach produces significantly higher-quality meshes and is significantly more robust and less prone to failure in certain complex scenes than GS2Mesh. The GBM framework can be further improved by adopting newer and newer improvements to Gaussian Splatting.

Chapter 5

Digital Twin Buildings: 3D Modelling, GIS Integration, and Visual Descriptions Using Gaussian Splatting

In this chapter, I build a prototype digital twin system for individual buildings, working towards the next generation of Urban Digital Twins. For this, I integrate fast and accurate 3D mesh reconstruction with (potentially automated) cloud mapping platform data retrieval. This is combined with automated user-accessible large language model-based multi-modal data analytics. This results in a digital twin system named Digital Twin Building that is fast, efficient, and user-accessible. The proposed digital twin system allows users with no prior coding knowledge to retrieve and analyze data using natural language.

This chapter is comprised of my research preprint eponymous with the chapter title [114]. Section 5.1 is the chapter's introduction which motivates and briefs the research. Section 5.2 introduces additional background material for the chapter. Section 5.3 is the methodology section which describes the Digital Twin Buildings framework. Section 5.4 presents and discusses the chapter's results. Section 5.5 presents a case study that is comprised of an independent preprint manuscript [115]. Section 5.6 is the chapter's conclusion section.

5.1 Introduction

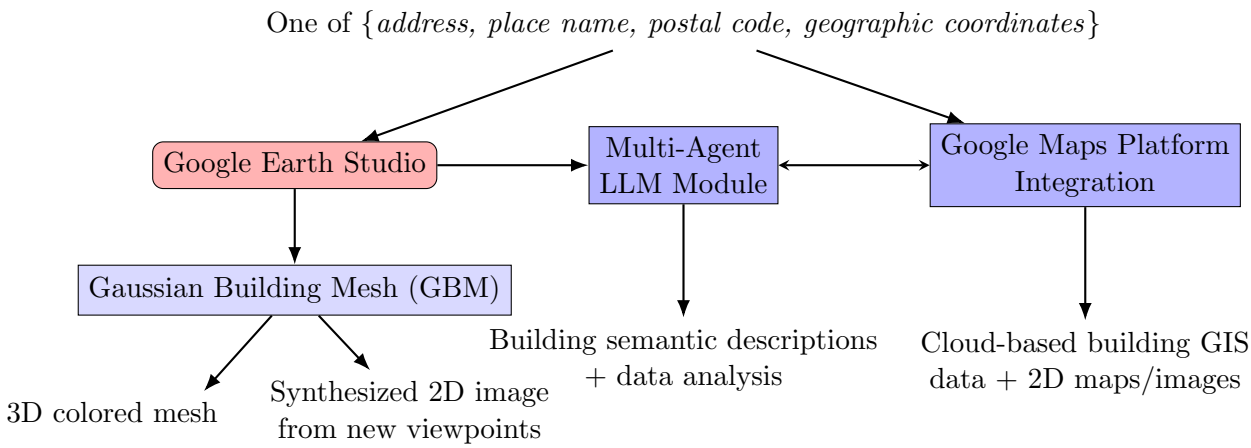


Figure 5.1: Diagram of the Digital Twin Building framework. External modules are boxed in red. Internal tools/modules are boxed in blue. **The modules specifically developed in this chapter are in dark blue**. Inputs and outputs are drawn in plain text.

Urban Digital Twins, as digital urban models with internet-connected data streams and integrated data analysis capabilities have been a key concept in geographic information systems and sciences since 2018, with general digital twin systems preceeding this concept by two decades [9]. However, modern urban digital twins face challenges in data

interoperability, data complexity, and interpretability among many others [10]. Recent advances in Large Language Models are particularly suited for solving these problems and are poised to propel advances in digital twin systems such as urban digital twins. Their adoption in these systems is still in an early stage [5, 116, 117], and I contribute to this research in this chapter.

In this chapter, the GBM framework from the previous chapter is combined with Cloud Mapping Service Integration and Multi-Agent Large Language Models (LLM) for data analysis to work towards urban digital twin systems, creating the Digital Twin Building Framework. The framework retrieves Gaussian Splatting and 3D mesh models with the GBM framework, fundamental geocoding and mapping information and 2D images, and is used to perform visual analysis on the 2D images using the Multi-Agent LLM module. This is shown in Figure 5.1.

Depending on need, the Google Maps Platform Integration can also retrieve local elevation maps, real-time traffic data, air quality data and access other data sources and services, which can then be analyzed.

The contributions are as follows.

- I introduce Digital Twin Building (DTB), a framework for extracting 3D mesh models of buildings. I integrate Cloud Mapping Services for retrieving geocoding, mapping information, and 2D images.
- I designed a novel Multi-Agent Large Language Models (LLM) module for user-accessible automated multi-modal data analysis, allowing for LLM-based code generation from user prompts.
- I performed extensive visual analysis experiments of multi-view/multi-scale images

of the building of interest using the LLM module. I assessed the performance of the popular ChatGPT(4o/mini) and Deepseek-V3/R1 models.

5.2 Additional Background

5.2.1 ChatGPT/Deepseek Application Programming Interface

Unlike traditional local deep learning, which necessitates GPUs for both training and inference, Application Programming Interface(API)–based Large-Language-Model querying requires minimal local hardware. The computation is offloaded to the server side, meaning a device with basic connectivity, such as a mobile phone, suffices for interacting with OpenAI or Deepseek servers to query the powerful ChatGPT and Deepseek LLMs. The number and frequency of the query is also not limited by client hardware. As such a simple mobile device can be used to process a large amount of data.

Table 5.1: Table of Commonly Used Deepseek and OpenAI LLMs, compiled on 2025-01-31.

Model Name	Model Class	Model Type	Image Processing	Parameters	USD /1M Input Tokens	USD / 1M Output Tokens
chatgpt4o-latest	GPT4o	Autoregressive	Analysis	$\sim$ 1000+B	2.5	10
gpt-4o-mini	GPT4o Mini	Autoregressive	Analysis	$\sim$10's of B	0.15	0.6
deepseek-chat	Deepseek V3	Autoregressive	OCR	617B	*0.14/0.7 $\times$ 0.1*	0.28
deepseek-reasoner	Deepseek R1 (V3-base)	Reasoning	OCR	617B	0.14	2.19
gpt-o1[1]	GPT-o1 (GPT4-base)	Reasoning	None	$\sim$175B	15	60

OpenAI models are not open-sourced, their model sizes (parameters) are estimated (B = billions, M = millions). [1]gpt-o1 was not included in the experiments due to cost, but its specifications are included for comparison. *The Deepseek V3 API call price is discounted by 50% until 2025-02-08, and by a further 90% if input caching is used for repeated identical prompting.*

5.2.2 Google Maps Platform

Google Map Platform is a cloud-based mapping service and a part of Google Cloud. Its APIs allow the client device to connect to various cloud-based GIS, mapping, and remote sensing services hosted on the Google Cloud servers.

The services utilized in this research include remote sensing image retrieval, map retrieval, elevation data retrieval, geocoding/reverse geocoding, and building polygon retrieval. However, Google Maps Platform also offers other APIs for urban and environmental research, including real-time traffic data, solar potential, air quality, and plant pollen data, in addition to the full suite of commonly used Google Maps navigation and mapping tools.

Although less known in the remote sensing and GIS community than its sister application Google Earth Engine, Google Map Platform has been used in a variety of GIS research including navigation, object tracking, city modelling, image and map retrieval, geospatial data analysis for commercial and industrial applications [118–122]. It is also used as part of many commercial software for cloud-based mapping integration.

5.2.3 Vision Language Models

Certain large language models support image-text dual-modality. Models such as DeepseekV3/R1 and their variants [76, 77], support Optical Character Recognition (OCR, i.e., reading text from images). Models like "ChatGPT-4o" [72] and its variants, and LLaMA 3.2+ [78], support complete interpretation and analysis of image content. This can be achieved by using image encoders with large-scale image-text pretraining which allows for embedding of text and image to a joint latent space.

The CLIP-based framework trains an image encoder and a text encoder together. It aligns their outputs in a shared embedding space. The goal is to bring matching image-text pairs closer and separate non-matching ones using contrastive loss, as follows.

$$\mathcal{L} = \frac{1}{2}\left[-\frac{1}{N}\sum_{i=1}^{N}\log p_{ii} - \frac{1}{N}\sum_{i=1}^{N}\log p'_{ii}\right] \tag{5.1}$$

where p_{ii} is the probability that image i matches its text, and p'_{ii} is the probability that text i matches its image. They are computed using softmax over cosine similarities of their embeddings.

$$p_{ij} = \frac{\exp\left(s_{ij}/\tau\right)}{\sum_{k=1}^{N}\exp\left(s_{ik}/\tau\right)}. \tag{5.2}$$

where s_{ij} is the cosine similarity of image i and text j given by (Eq. 5.5) and τ is a scaling hyperparameter.

Vision Language Models [123] is the term used to describe large language models with image processing capabilities. These models typically consist of an LLM backbone, an image encoder, and a tokenizer/projector that transforms the embedding into input tokens for the LLM. This is shown in Figure 5.2.

ChatGPT-4o [72], Llama 3.2 [78], QWen-VL [124], the Gemini series [125], and Chameleon [126] are well-known examples of VLMs which are either CLIP-based or mixed-token-based. These vision language models offer new approaches to image processing and are especially suited for zero-shot or few-shot tasks.

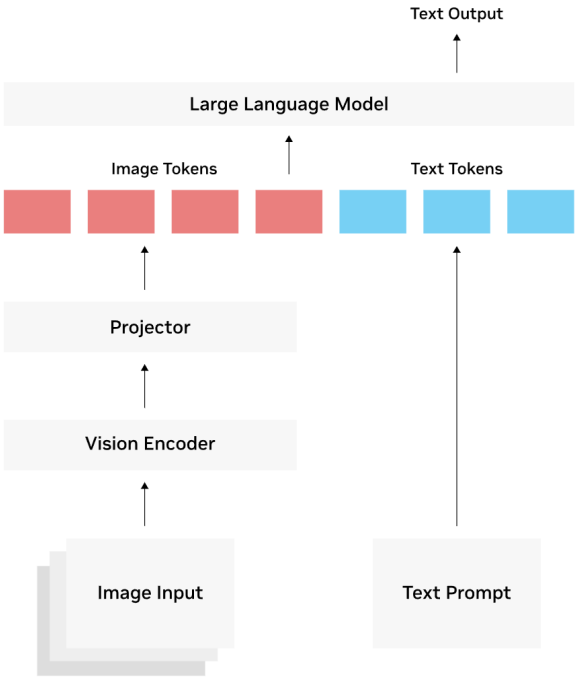


Figure 5.2: Diagram of a typical VLM framework, sourced from [123]. A typical VLM that is built on top of an existing LLM encodes and projects input images to tokens which can be input into the pretrained LLM.

5.3 Methodology

5.3.1 GBM Mesh Extraction

Gaussian Building Mesh (GBM) [100], introduced in Chapter 4, is an end-to-end 3D building leveraging Google Earth Studio [31], Segment Anything Model-2 (SAM2) [104] and GroundingDINO [103], and a modified [84] Gaussian Splatting [20] model. The pipeline enables the extraction of a 3D building mesh from inputs such as the building's name, address, postal code, or geographical coordinates. Since the GBM uses Gaussian Splatting (GS) as 3D representation, it also allows for the synthesis of new photorealistic 2D images

of the building under different viewpoints. The GBM pipeline is as follows.

- Google Earth Studio is used to retrieve multi-view remote sensing images from an address, postal code, place name, or geographic coordinates.
- SAM2 and GroundingDINO are used for consistent building mask extraction using text or click-based inputs. This module extracts the building from its background in 2D images.
- A mask refinement method was developed, incorporating morphological operations and the Ramer-Douglas-Peucker algorithm to improve 2D mask accuracy.
- An improved version of the 2DGS Gaussian Splatting [84] model is utilized to learn a 3D representation from 2D multi-view masked remote sensing images.
- 3D coloured building meshes are produced through masked Truncated Signed Distance Function (TSDF) fusion.

5.3.2 Google Maps Platform Integration

Python client binding for Google Maps Platform Services APIs is used to create an integration tool to automatically retrieve the GIS and mapping information of a building of interest. For these image analysis experiments, the data is retrieved with four API calls. The first is a Google Maps Platform Geocoding/Reverse Geocoding API call which retrieves the complete address information including geographic coordinates, entrance(s) coordinates, and building polygon mask vertex coordinates. Then, a Google Maps Platform Elevation API call is used to retrieve the ground elevation using the building's coordinates as input. Finally, two API calls are made using the Google Maps PlaFtform Static Maps

API to retrieve map(s) and satellite/aerial image(s) at the desired zoom level. This process is illustrated in Figure 5.3. The aerial/satellite image(s) is then used as one of the inputs to the Multi-Agent LLM Module.

Google Map Platform Integration can easily be modified to retrieve additional data from the cloud-based mapping service by adding parallel API calls below the Geocoding/Reverse Geocoding API call (e.g. Air Quality, Weather, Solar Data, Traffic, etc...). For example, if we wish to analyze real-time traffic data, we can simply perform API calls to the Google Maps Platform's Traffic API.

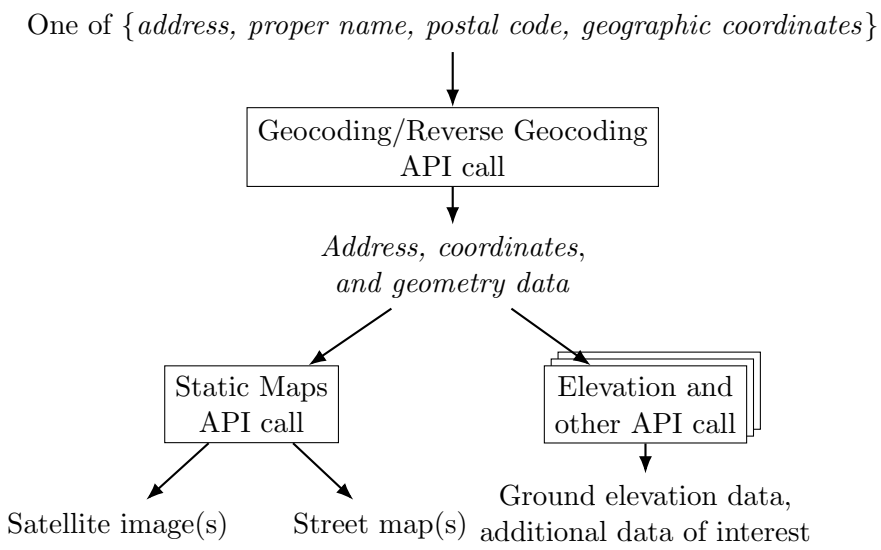


Figure 5.3: Diagram of our Google Map Platform API Services Integration Tool.

5.3.3 Multi-Agent LLM Analysis of Multi-View/Scale Images

The motivation of this module is to create a multi-agent LLM system to analyze the multi-modal data retrieved from Google Cloud Platform Services integration. An additional goal is to enable sufficient automation for users with no coding knowledge to perform analysis

and data retrieval with natural language. For the most of this chapter, I restrict the scope of this LLM analysis to the multi-agent multi-modal content analysis of multi-view/scale images. I also added an additional case study extending the scope to air quality data analysis.

For this analysis, the primary goal is to retrieve and store keywords describing the architectural style, function, landscape, and architectural elements of the building using keywords. With the idea that an accurate set of keywords will allow an agent to reconstruct a text-based description of the image without actually seeing the image, I also retrieve the caption using keywords as a secondary objective. This process is illustrated in Figure 5.4.

From a building's address, place name, postal code, or geographic coordinates, I retrieve multi-view off-nadir images of the building of interest using Google Earth Studio (or use the ones previously retrieved in the GBM module). I also retrieve top-down view aerial/satellite image(s) at different scales using the building using Google Map Platform Integration. For each image, I initiate a GPT4o/GPT4o-mini agent and prompt it to analyze the image and retrieve a set of keywords for each image. I then initiate two agents, one to aggregate the keywords from all the images of the building, and one to turn the aggregated keywords into a human-readable caption description.

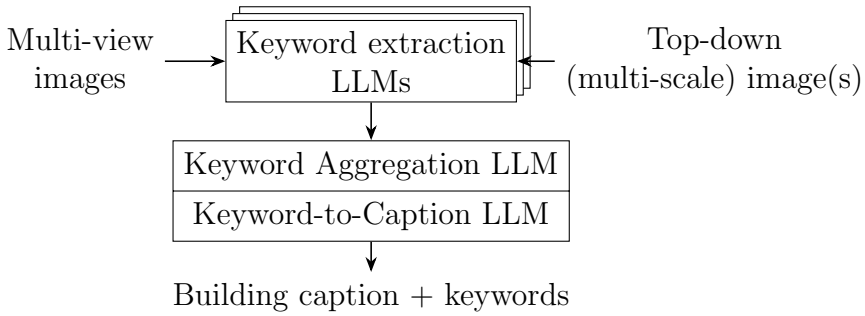


Figure 5.4: Diagram of Multi-Agent LLM processing multi-view-multi-scale images.

5.3.4 Automated Multi-Agent Code Generation and Execution

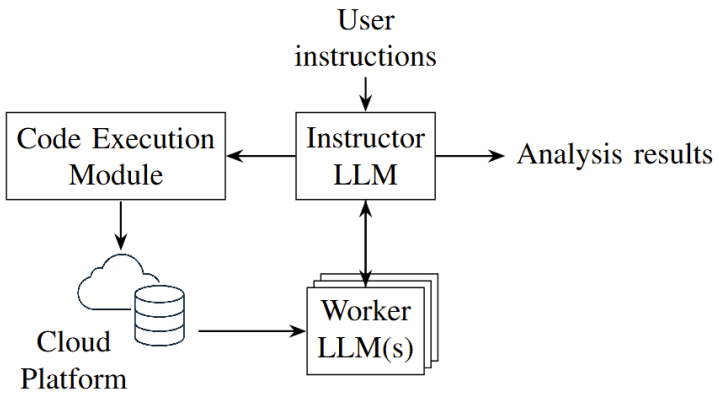


Figure 5.5: Instructor-Worker System for automated cloud-based dataset and data stream retrieval.

Further automation can be achieved by adding a code execution module and carefully designing the LLM agents' system prompts and prompt passing. The correct Google Maps Platform API calls for data retrieval (or calls to other cloud-based data sources) require domain-specific knowledge and are syntax-sensitive. For better user accessibility, I designed an automation framework that automates this step, allowing users to perform data analysis retrieval and analysis using natural language.

This automation framework is illustrated in Figure 5.5. The Instructor LLM is a Chain-of-Thought reasoning model such as Deepseek-R1/GPT-O1 with a higher problem-solving capability, but a more expensive API call price and typically slower response time. The Worker is an autoregressive (chat-style) model that is inexpensive to use and has a faster response time than reasoning models.

The Instructor LLM is responsible for providing the query to the cloud mapping API in a one-shot manner. I.e. The user provides to the Instructor a single example for the format of the desired API call from our Python wrapper introduced in Digital Twin Building's

Google Maps Platform integration [114]. The user can also provide code documentation to the Instructor. The Instructor's response string is processed through our Code Execution Module which is comprised of two code compliance checks and a string-to-code execution function. For the first check, the module initializes another LLM agent and asks it to examine the response string for any malicious codes. For the second check, string formatting is used to check whether the response string complies with the one-shot example's code format. If both checks are passed, the API call from the Instructor LLM is executed as code using Python's built-in string-to-code execution function *exec()*. Although the Code Execution Module was only used for the Google Maps Platform API call, it can also be used to automate code generation for post-processing or the final analysis, e.g. to generate plots during analysis, as long as code documentation or an example is provided. **These one-shot coding examples (either the documentations or an example function calls) can also be pre-declared and stored as a system prompt allowing for a user with no coding knowledge to use the system to perform data retrieval and analysis.** I.e., the one-shot examples can be pre-defined as opposed to user-provided. Although only the Google Maps Platform API call was tested for code generation-based automation, other code generation-based automation can easily be achieved, such as for data post-processing, plotting, and numerical analysis.

This modified framework can be used to automate the retrieval of data from cloud platforms such as Google Maps Platform, greatly improving user accessibility. This modified automated framework is tested in the case study in Section 5.5.

5.3.5 Natural Language Processing Metrics

The evaluation of image analysis results in the absence of ground truth was an issue encountered in this research. Although metrics such as BLEU (Bilingual Evaluation Understudy) and CIDEr (Consensus-based Image Description Evaluation) are commonly used to evaluate captioning performance, they require supervised datasets with ground truth captions [127]. However, the images lack ground truth captions. As a no-reference metric for evaluating caption generation on an unsupervised dataset, I use the CLIP [69], BLIP [128], and PAC scores [129], which are commonly used no-reference image captioning metrics. Readers are referred to the original papers for detailed explanations. These three scores are all based on the cosine similarity between co-embedded image and text features using a pretrained encoder.

$$\text{CLIP-BLIP-PAC Score } (\%) = 100\frac{\mathbf{t}\cdot\mathbf{i}}{\|\mathbf{t}\|\|\mathbf{i}\|}. \tag{5.3}$$

where the text embedding of the caption $\mathbf{t}$, and the image embedding of the corresponding image $\mathbf{i}$ are from the respective CLIP, BLIP, PAC encoders. Additionally, the PAC score is truncated at zero and can be rescaled (the original PAC paper scales up the score by an additional factor of two).

I also use perplexity to roughly assess Large-Language-Model image-to-keyword extraction confidence. Perplexity is a measure of the model's confidence in its response and is given by

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(x_i)\right) \tag{5.4}$$

where $\log P(x_i)$ is the log-probability of the token x_i as assessed by the model during text-token autoregression.

The BERTScores (BERT Precision, Recall, and F1 Score) [127] are full-reference metrics used to assess text generation results in the presence of a reference text, and are used in Section 5.5. They correlate better with human judgment than BLEU and CIDEr scores and capture semantically meaningful similarities. The BERTScores use a pre-trained BERT model [66] to embed both an input text $(c_1, c_2 ... c_m)$ and a ground truth text $(r_1, r_2, ... r_n)$, $\mathbf{e}_{c_i} = \text{BERT}(c_i), \quad \mathbf{e}_{r_j} = \text{BERT}(r_j)$. Then, a cosine similarity matrix is calculated. BERTScore precision, recall, and F1 are then computed based on similarity matching.

The cosine similarity is given by,

$$s_{ij} = \frac{\mathbf{e}_{c_i} \cdot \mathbf{e}_{r_j}}{\|\mathbf{e}_{c_i}\| \|\mathbf{e}_{r_j}\|} \tag{5.5}$$

where i indexes candidate text (model output) and j indexes reference text (ground truth).

The BERTScore Precision is obtained by matching each candidate token with the most similar reference token and averaging their mutual cosine similarities.

$$P_{BERT} = \frac{1}{m} \sum_{i=1}^{m} \max_j s_{ij}. \tag{5.6}$$

The BERTScore Recall is obtained by matching each reference token with the most similar candidate token and averaging their mutual cosine similarities.

$$R_{BERT} = \frac{1}{n} \sum_{j=1}^{n} \max_i s_{ij}. \tag{5.7}$$

The BERTScore F1 is calculated using the harmonic mean of BERTScores Precision and

Recall.

$$F_{1BERT} = 2 \cdot \frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}}. \tag{5.8}$$

5.4 Experiments and Discussion

I chose seven different buildings to test my framework for multi-modal LLM-based data analysis. These include well-known landmarks, commercial, residential, and institutional buildings. I extract 31 multi-view images in a 360° view pose around the building of interest, which I then use in conjunction with the GBM module to create the 3D coloured mesh of the building. Then I subsample six images, one every 70°, as inputs to the Multi-Agent LLM module. I also use the Google Map Platform integration to retrieve two aerial/satellite image(s), one at Google Maps zoom level 18, and one at Google Maps zoom level 19 as inputs to the Multi-Agent LLM module.

In preliminary experiments, I noticed a relatively large variation in the final CLIP-score across different attempts even while using the same model and prompt, since the LLMs' outputs are not deterministic (even when using *LLM Temperature* = 0). As such, I perform two experiments to understand the performance of the module when using different models as LLM agents, and to understand the distribution of scores across different attempts for both the keyword extraction step and the captioning step.

5.4.1 Keyword Extraction

For multi-agent image-to-keyword extraction, both *chatgpt-4o-latest, and gpt-4o-mini* are suitable. Additionally, both high-resolution and low-resolution image analysis are available. I tested all 4 combinations for all 7 scenes, for 10 iterations, with 8 images and LLM agents,

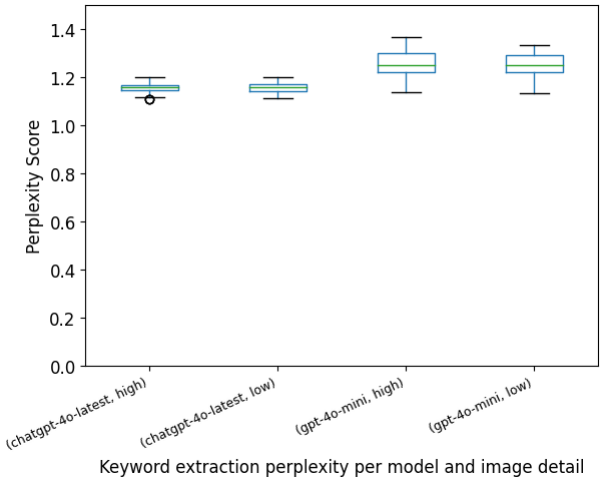


Figure 5.6: Box plot of image-to-keyword perplexity distribution per model and level of image detail (2240 samples total).

resulting in a total of 2240 API calls. I recorded the LLM responses and the perplexity scores. The results are shown in Figure 5.4.

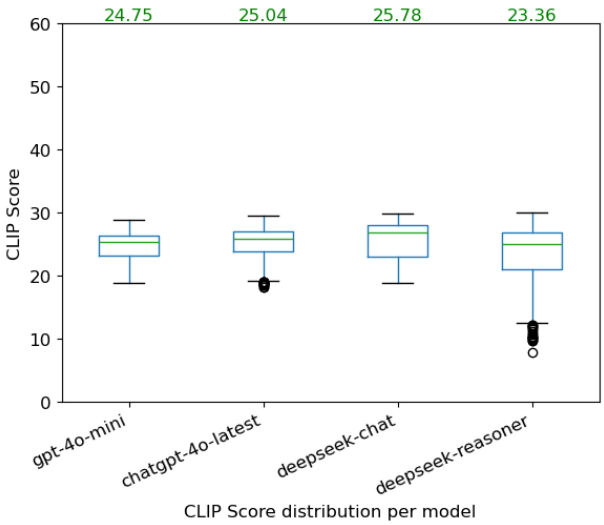


Figure 5.7: Box plot of CLIP Scores per model (mean in green).

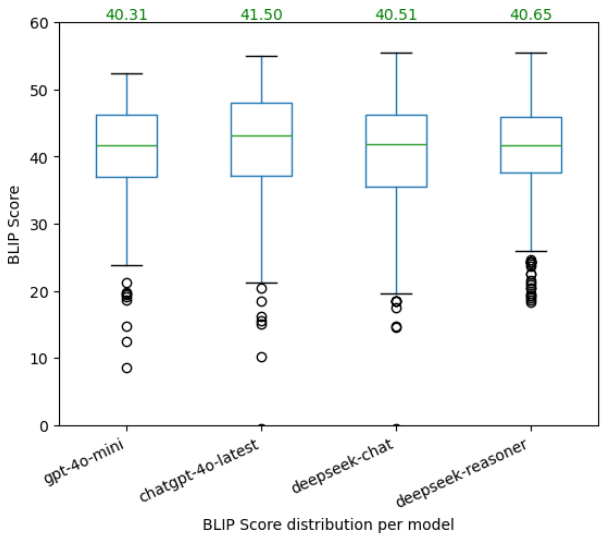


Figure 5.8: Box plot of BLIP Scores per model (mean in green).

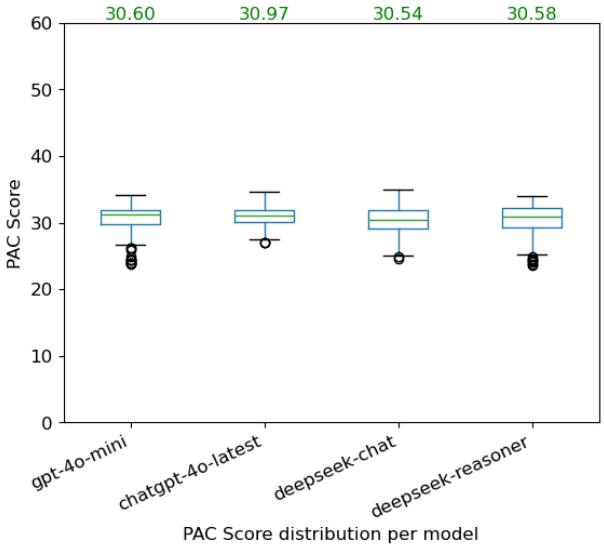


Figure 5.9: Box plot of PAC Scores per model (mean in green).

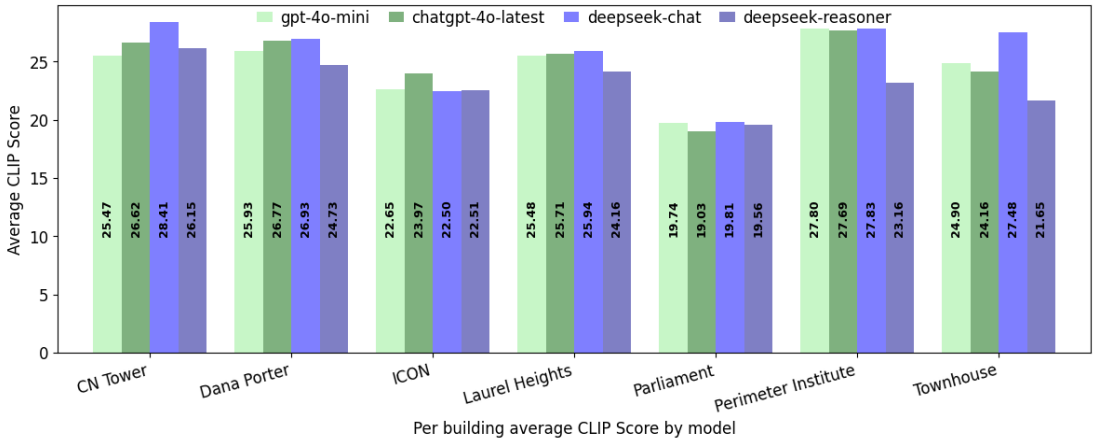


Figure 5.10: Average CLIP score of each building image captioning model for each different scene.

5.4.2 Captioning

To evaluate the multi-image captioning capabilities, I run the image captioning pipeline end-to-end using *gpt-4o high image resolution* API calls for keyword extraction. For each of the 7 buildings, we test 5 iterations of keyword-aggregation-caption for each of the four models: *gpt-4o-mini, chatgpt-4o-latest, deepseek-chat, deepseek-reasoner*. Each test requires 10 API calls (eight calls for image keyword extraction, one for aggregation, and one for caption generation), totalling 1400 API calls. I calculate CLIP, BLIP, and PAC scores for every single one of the input images. This results in $7 \cdot 5 \cdot 4 \cdot 8 = 1120$ triplets of scores, or 280 triplets of scores per model. The image captioning score distributions are visualized in Figures 5.7, 5.8, and 5.9.

5.4.3 Visualization

I present a visualization of the extracted 3D model, caption, keywords, and Google Maps Platform-based information for the Perimeter Institute (PI) building scene in Figure 5.11. The Perimeter Institute for Theoretical Physics is an independent research centre located at 31 Caroline St. N, Waterloo, Ontario, Canada. I show the 3D mesh and depth maps extracted from the scene, the 2D map, and the aerial image with the building's polygon at Google Maps zoom level 18, retrieved via the Google Maps Platform Static Maps API. Additionally, I plot the keywords extracted from a single view and the caption generated by the Multi-Agent LLM module.

5.4.4 Discussion of Digital Twin Buildings

At each testing iteration, the generated caption differs slightly. There is some inherent variation to the captioning scores across iterations. As such, I used repeated testing with box plot visualization to capture this behaviour better. Because I test the captions with respect to both top-down images at different resolutions and angled images, there is an even larger variability in the scores than can be accounted for by differences in the generated captions. The BLIP Score and PAC Score were more sensitive to outliers. By examining these outliers, we were able to confirm that outliers occurred when combining an overly descriptive view with a viewpoint which does not contain the described image features. I.e. outliers are not model-specific as the CLIP Score in Fig 5.7 would lead us to believe.

The BLIP Score shows a much larger interquartile range than CLIP and PAC Scores, which demonstrates that it is less robust (more volatile to non-important captioning and image differences). PAC score proved to be both robust (with its small interquartile range) and sensitive (with its ability to detect meaningful outliers). This is consistent with the

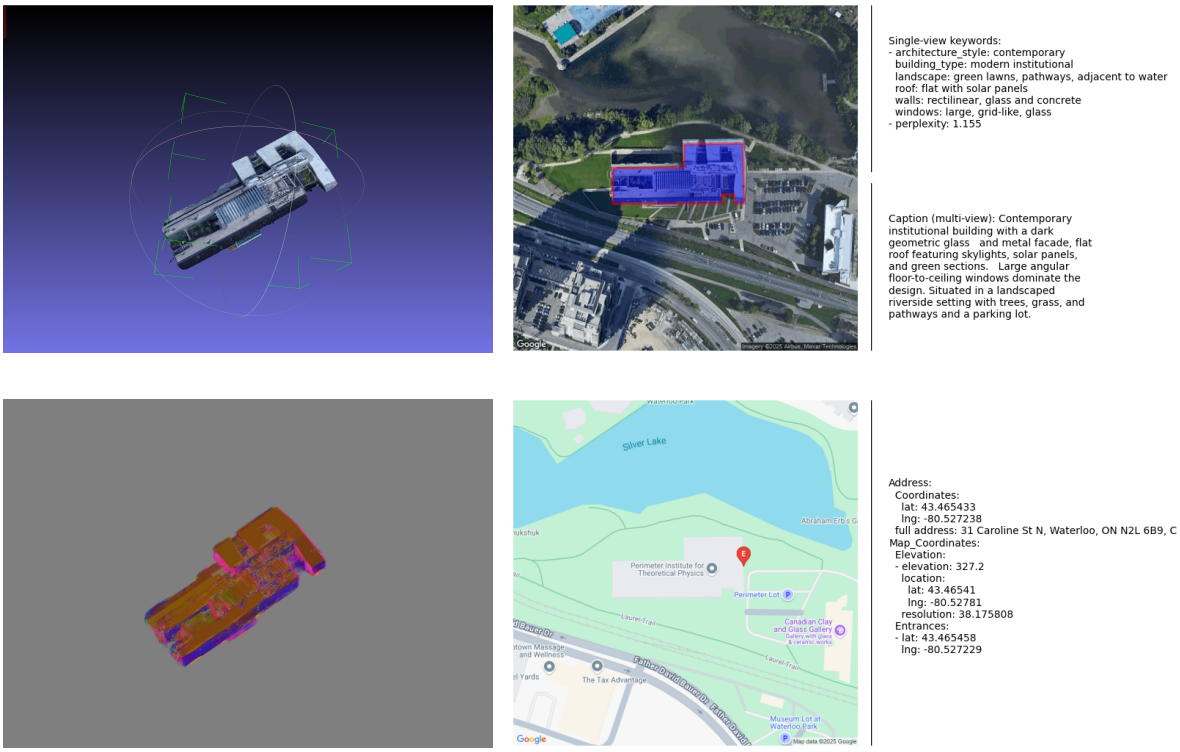


Figure 5.11: Visualization of results. **Top Left**: Colored 3D mesh. **Bottom Left**: Depth map. **Top Right**: Aerial image with keywords and captions and retrieved polygon mask. **Bottom Right**: Retrieved map with map information (the entrance is labelled with a red place marker).

experiments in [129] strongly correlating it with human assessment of captioning.

5.5 Application: Air Quality Analysis

5.5.1 Google Maps Platform Air Quality API

Google Maps Platform offers air quality data in the form of current air quality, 720-hour history at 500 m resolution, and current air quality tile maps. This data is gathered from

various local air quality monitoring stations and satellite sensors from local providers and interpolated when necessary.

Table 5.2: Google Maps Platform Air Quality Pollutant List, Abbreviation, and Unit

Pollutant	Abbreviation	Unit
Carbon monoxide	CO	Parts per billion (PPB)
Nitrogen dioxide	NO2	Parts per billion (PPB)
Ozone	O3	Parts per billion (PPB)
Inhalable particulate matter (<10µm)	PM10	Micrograms per cubic meter (µg/m^3)
Fine particulate matter (<2.5µm)	PM2.5	Micrograms per cubic meter (µg/m^3)
Sulfur dioxide	SO2	Parts per billion (PPB)

Google Maps Platform offers air quality measurements for six pollutants: carbon monoxide, nitrogen dioxide, ozone, particulate matter $< 10\mu m$, fine particulate matter $< 2.5\mu m$, and sulphur dioxide. These are listed in Table 5.2. The API also provides information on potential pollutant sources, health hazards, local air quality indices, and health advisory warnings.

5.5.2 Los Angeles Wildfire

The Pacific Palisades Region of Los Angeles, California was affected by a severe wildfire in January 2025, starting on January 7th and rapidly spreading over the next few days. It was contained after 24 days on January 31st. It was one of the two major wildfires that devastated Los Angeles in January 2025, causing more than 250 billion dollars in damage and burning an area of around 155 km^2 [130] [131].

Large wildfires are known to produce inhalable particulate matter at a level significantly above the baseline of a region. Two common measures of these particulates are PM2.5 level and PM10 level, which stand for particulates of 2.5 microns or less and particulates of 10 microns or less. [130]

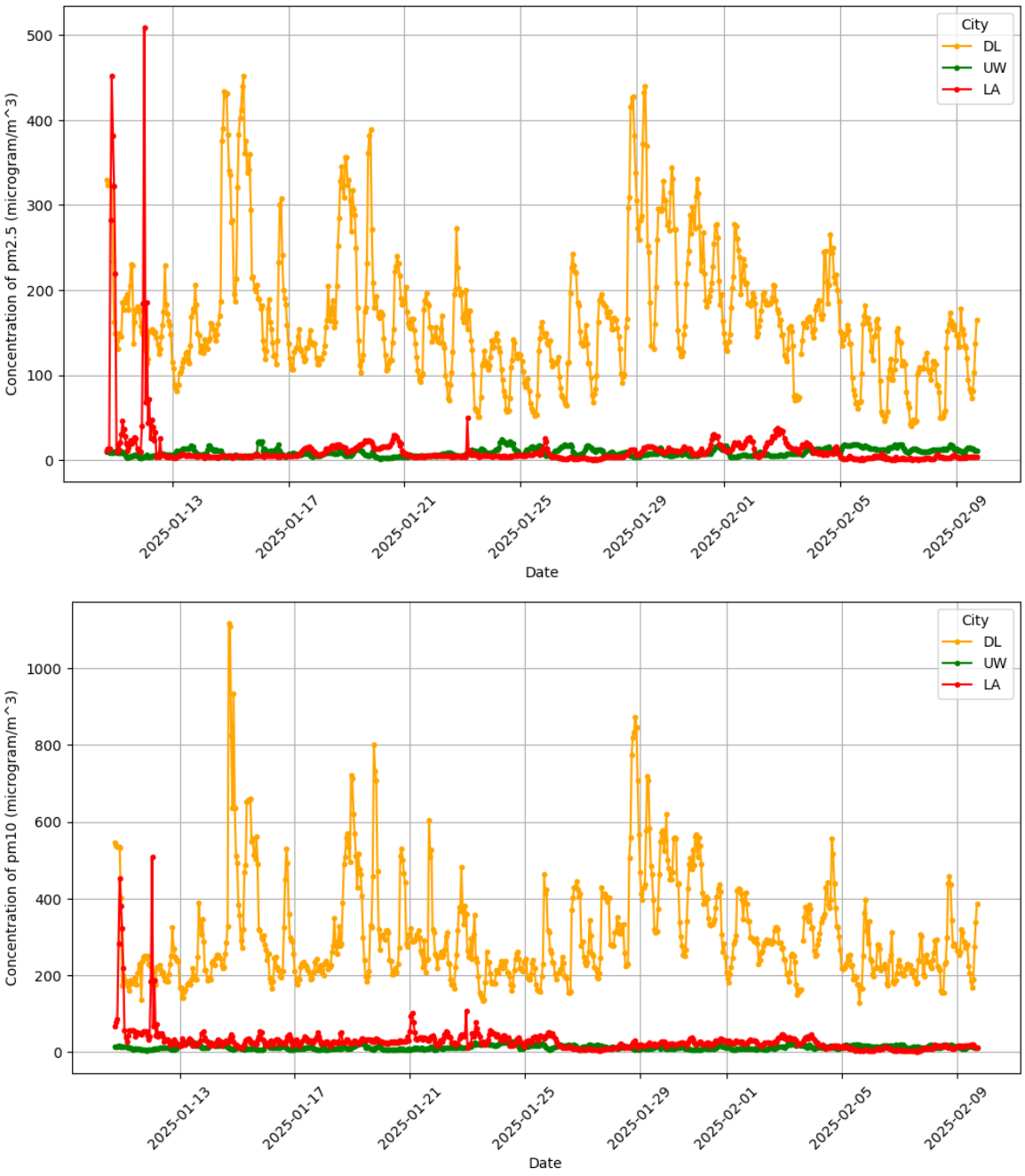


Figure 5.12: PM2.5 (**Top**) and PM10 (**Bottom**) levels from January 9th to February 9th in select neighbourhoods in Los Angeles (LA), Waterloo (UW), Delhi (DL).

Table 5.3: Air Quality Retrieval Neighbourhoods and Coordinates

City	Coordinates	Neighbourhood
Los Angeles, USA	34.0725° N, 118.5445° W	Topanga State Park
Waterloo, Canada	34.07022° N, 80.5449° W	University of Waterloo
Delhi, India	28.7041° N, 77.1025° E	Abhinav Public School

To analyze the impact of the wildfires on the Pacific Palisades' PM2.5 and PM10 levels, I use the Google Maps Platform integration to retrieve the 720 hour air quality history of the Topanga State Park neighbourhood, the suspected starting point of the Palisades wildfire. The data begins on January 9th 2025 due to the 720 hours limit at the time of data retrieval. I also retrieved the air quality at the University of Waterloo in Waterloo, Canada, as a low pollutant baseline and a Delhi neighbourhood in Delhi, India, as a high pollutant baseline. The PM2.5 and PM10 levels of the respective neighbourhoods (Table 5.3) are plotted in Figure 5.12.

When examining the Los Angeles Palisade PM2.5/10 levels, two major spikes can be observed, roughly on January 9-10th and 11-12th. There are also a multitude of smaller spikes. The January 9-10th and 11-12th major spikes have amplitudes that are significantly above the Waterloo baseline for clean air but are comparable to the baseline of Delhi. Surprisingly, the LA PM2.5 and PM10 pollutant levels are fairly comparable to Waterloo's levels outside of the spike events, despite the ongoing wildfires. This suggested a certain degree of localness to the pollution caused by wildfires.

Fig. 5.12 shows two major spikes in particulate matter in the first three to four days of the time series, as well as a few minor spikes later on. The particulate matter levels returned to baseline fairly quickly. The baseline air quality near the measurement site is close to clean city baseline levels despite the ongoing fires that lasted for the entire month. The data seem to show that PM2.5 and PM10 levels are highly localized, perhaps

with spikes occurring only with nearby active wildfires close to the sensor site. This effect appears to not be mentioned in the literature and should be investigated further.

5.5.3 Multi-Agent Analysis of Air Quality

Below is an example of the framework in use when the user asks for an analysis of the PM2.5 and PM10 levels of the LA neighbourhood during the time period. The Instructor LLM queries the cloud platform to retrieve the data and instructs the Worker LLMs to analyze the data.

Response 1: Example Instructions for Worker LLMs.

1. **Data Extraction**: For each day in the dataset, calculate the daily average levels of PM2.5 and PM10 using the hourly data.

2. **Outlier Detection**: Identify outlier events by flagging days where PM2.5 or PM10 levels exceed 1.5 times the interquartile range (IQR) above the 75th percentile or below the 25th percentile.

3. **Summarization**: Compress the data into a structured format: - Daily averages: {date: YYYY-MM-DD, PM2.5_avg: value, PM10_avg: value} - Outlier events: {start_date: YYYY-MM-DD, end_date: YYYY-MM-DD, pollutant: PM2.5/PM10, max_level: value}

4. **Output**: Return only the structured summary without additional commentary or explanations.

Then, Worker LLMs respond with their summary for their respective chunk. A response for a chunk is provided below as an example:

```
{
  "daily_averages": [
    {"date": "2025-02-04", "pm25_avg": 8.94, "pm10_avg": 13.23},
    {"date": "2025-02-05", "pm25_avg": 2.48, "pm10_avg": 7.34},
    {"date": "2025-02-06", "pm25_avg": 2.63, "pm10_avg": 7.37},
    {"date": "2025-02-07", "pm25_avg": 1.56, "pm10_avg": 5.63},
    {"date": "2025-02-08", "pm25_avg": 3.12, "pm10_avg": 13.12},
    {"date": "2025-02-09", "pm25_avg": 3.34, "pm10_avg": 15.12}
  ],
  "outlier_events": [
    {"start_date": "2025-02-04", "end_date": "2025-02-04",
    "peak_pm25": 16.2, "peak_pm10": 18.44},
    {"start_date": "2025-02-08", "end_date": "2025-02-09",
    "peak_pm25": 5.99, "peak_pm10": 19.79}
  ]
}
```


The summaries for all chunks are aggregated and passed back to the Instructor LLM, which receives further instruction. For example, when asked to analyze the time period for outliers and possible causes, the Instructor LLM responds with the following.

Response 2: Key Pollutant Spike Events & High-Pollution Days Analysis

1. Extreme Pollution Events (Significantly Above Regional Averages)

- **January 10, 2025**: - **PM2.5**: Peaked at **382.03** (extreme spike). - **PM10**: Peaked at **452.24** (extreme spike). - **Likely Event**: Severe short-term pollution event (e.g., industrial accident, wildfire, or dust storm).

- **January 12, 2025**: - **PM2.5**: Peaked at **509.07** (highest recorded spike). - **PM10**: Peaked at **509.07** (unusually high co-spike). - **Likely Event**: Prolonged combustion event

(e.g., large-scale biomass burning or industrial emissions).

- **January 21–22, 2025**: - **PM10**: Spiked to **92.45** (Jan 21) and **108.05** (Jan 22). - **Likely Event**: Localized dust storms or construction activity.

- **February 2, 2025**: - **PM2.5**: Peaked at **37.86**; **PM10** at **45.10**. - **Likely Event**: Moderate industrial/vehicular pollution surge.

2. Days with Sustained High Pollution

- **January 10–16, 2025**: - **PM2.5**: Consistently elevated (avg **166.91** on Jan 10, dropping to **4.92** by Jan 13). - **PM10**: Similarly high (avg **174.79** on Jan 10, stabilizing at $\sim$ **27.92** post.

This LLM system was correctly able to identify outlier time periods and provide a probable cause for the level of pollutants, as well as assess that the pollutant levels stabilized at around 10% of the maximum level. Both PM10 and PM2.5 levels peaked at **509.07** $\mu g/m^3$. This number was not hallucinated by the LLMs and was double-checked against the raw data from the Google Maps Platform. As it is unlikely for two measurements to peak at the same value up to five significant digits, this number could be significant for other reasons (e.g. sensor limitations). The smaller spikes of PM10 and PM2.5 pollutants after stabilization confused the LLM system (e.g., January 21–22). It was not able to deduce that the region was devastated by large-scale wildfires throughout the entire month. Despite the fires moving away from the sensor after the initial few days, the major cause of the pollutants was still the wildfire, not dust storms or construction.

5.5.4 Multi-Agent Health Recommendation and Benchmark

Google Maps Platform Air Quality API provides health recommendations for healthy people, people with lung diseases, heart diseases, and pregnancies, including the elderly, and children. These recommendations can be retrieved together with the air quality history. To compare the effectiveness of the Instructor-Worker framework for policy recommendation, as final instruction after receiving the Worker summaries, I prompted the Instructor to generate health recommendations for healthy people and people with lung diseases for three days in the Topanga State Park neighbourhood in Los Angeles. January 10 (low air quality), January 22 (moderate air quality), and February 9th (good air quality). Example responses for three different iterations are shown below. (Iteration 4 had the lowest semantic alignment according to BERTScore)

Response 3: January 10 Health Recommendation for People with Lung Disease:

- **Iteration 1**: Air quality is unhealthy and may aggravate lung conditions. - Advised to avoid all outdoor physical activities. - Stay indoors in a well-ventilated space. - Consult a healthcare provider if symptoms arise.

- **Iteration 2**: Air quality is hazardous. - Remain indoors and keep activity levels low. - Follow medical advice and consider using air purifiers.

...

- **Iteration 4**: Air quality is very unhealthy. - Avoid all outdoor activities. - Stay indoors with windows closed. - Use air purifiers if available. - Follow doctor's recommendations and have quick-relief medication on hand.

...

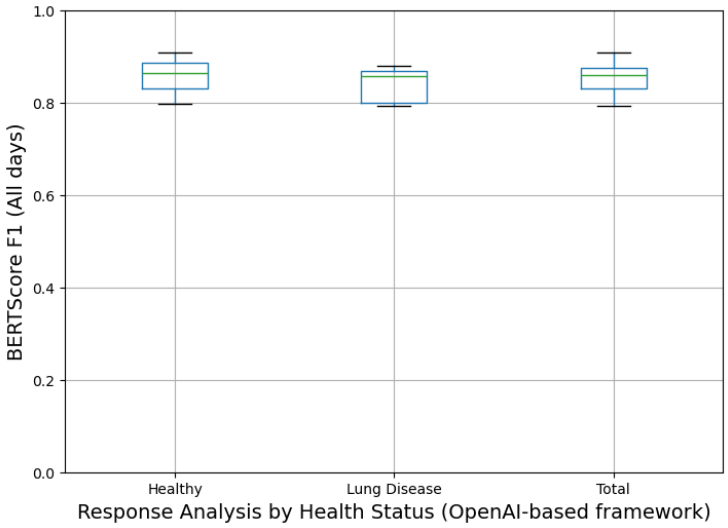
- **Ground Truth**: - Reduce the intensity of outdoor activities. - Keep relevant med-

ications available and consult a doctor if needed. - Limit exposure to busy roads, open fires, and smoke sources. - Reduce time near industrial emission stacks. - Staying indoors with an activated air filtration system is recommended for long-term health.

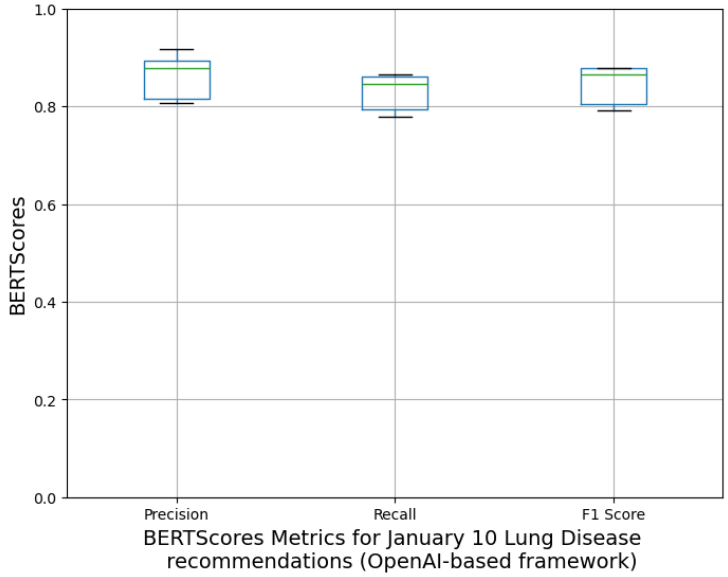
Despite using an LLM temperature of zero, the framework produced semantically similar, but structurally different sentences. I ran 10 iterations of the framework with identical initial instructions to assess the variance in the responses. The responses were assessed against ground truth recommendations from the Air Quality API using the BERTScores Eqs. (5.6,5.7,5.8).

Figure 5.13b shows a relatively large within-(sub)class variance, comparable to the total variance (for people with lung disease). When plotting the BERTScores for health recommendations for people with lung disease on January 10th in Sub-Figure (b), an extensive interquartile range of around 0.6 for all three scores can be observed. This is comparable to the interquartile range for people with lung disease across all days and is larger than the total interquartile range. This suggests that the variance is primarily driven by iteration-to-iteration differences in text generation rather than by differences between high- and low-pollution days or between people with lung disease and healthy individuals.

Moreover, the lack of outliers (which would have been plotted as circles) in Sub-Figure (a) of Figure 5.13b indicates that the recommendations' semantic alignments were consistently maintained with the ground truths provided by Google's Air Quality API. In fact, the model is fairly consistent in generating good scores, as shown in Figure 5.14. This is also observed by examining the health recommendation with minimum BERTScores (P = 80.8 %, R = 77.8 %, F1 = 79.3 %) which occurs at the fourth iteration for January 10 for people with lung disease, shown in Response 3. The scores still show a positive semantic alignment, which is corroborated by comparing the response to the ground truth.



(a) Interclass BERTScore F1 comparison for health recommendation comparison for healthy people vs. people with lung disease.



(b) BERTScore F1, Precision, Recall for January 10 recommendation for people with lung disease.

Figure 5.13: BERTScore distribution comparison for health recommendations.

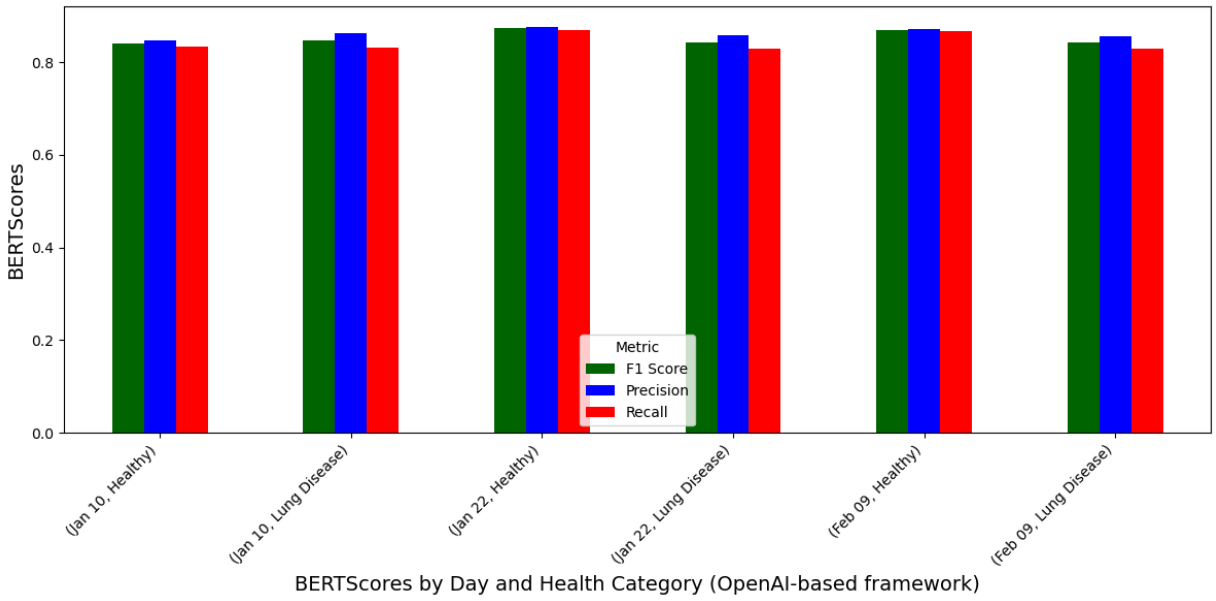


Figure 5.14: Bar graph of BERTScores for health recommendations by day (air quality) and health class.

By examining example responses (3), a substantial difference in writing style between the LLM system's responses (more concise) and the ground truth writing style (more elaborate) can be noticed. Aligning the writing style more closely with the ground truth would be trivially easy and would likely yield higher average BERT scores. However, this increase in score would not necessarily indicate better recommendations in practice. Since the absence of outliers is a better indicator of recommendation success than perfect textual similarity.

The reasoning model-based Instructor LLM can also generate more complex and comprehensive reports and recommendations with cross-references to data. Preliminary tests of this scenario were performed by prompting the system to generate a short report on the air quality. Without a ground truth policy to compare against, a numerical benchmark cannot be provided. However, visual inspection indicates that the example recommended

policy report is reasonable. An example of a short policy recommendation report based on air quality data is shown below. The Instructor LLM was prompted with *The area was affected by wildfires. Generate a short report with policy recommendations based on the air quality.*

Response 4: Air Quality Report and Policy Recommendation

1. **Severe Pollution Episodes**: Extreme PM2.5/PM10 spikes coincided with wildfires, far exceeding WHO guidelines. - Example: Peaks of 452–509 µg/m^3 on 01/10–01/12 (WHO 24-hour PM2.5 limit: 15 µg/m^3).

2. **Prolonged Exposure**: Elevated PM2.5/PM10 levels persisted for weeks, indicating chronic air quality degradation. - Example: PM10 >70 µg/m^3 on 01/21; PM2.5 >25 µg/m^3 recurrently.

3. **Recurring Outliers**: Hourly PM2.5/PM10 peaks suggest ongoing intermittent wildfire activity or secondary dust dispersion. - Example: 30.1 µg/m^3 PM2.5 on 01/31; 108.05 µg/m^3 PM10 on 01/23.

4. **Policy Recommendations**:

- **Emergency Response Framework**: - Deploy real-time SMS/APP alerts prioritizing vulnerable groups. - Distribute N95 masks and designate HEPA-filtered emergency shelters.

- **Wildfire Mitigation**: - Implement pre-fire season controlled burns to reduce fuel loads. - Invest in satellite/ground sensors for early wildfire detection.

- **Health Infrastructure**: - Stockpile respiratory medications and expand ICU capacity. - Launch public awareness campaigns on PM2.5 health risks.

- **Long-Term Air Quality Management**: - Regulate agricultural/industrial burning

during dry seasons. - Establish green buffers with fire-resistant vegetation.

- **Climate Resilience**: - Promote drought management policies to reduce wildfire risk. - Strengthen cross-regional collaboration for air quality monitoring.

5.5.5 Discussion of the Application-based Case Study

This research showed the viability of LLM systems for data analysis and policy recommendations. The recommended policies were always aligned with ground truth policies. Unlike supervised models, this framework can be directly applied to new situations without ground truth.

Although our framework was built and tested for OpenAI and Deepseek APIs, only benchmarked OpenAI models for repeated BERTscore assessments were benchmarked. Although Deepseek is a less expensive LLM service provider, as of February 2025, the platform is less reliable. When attempting to make continuous API calls to Deepseek, service refusals occurred. To allow for more LLM model and provider choices, LLM APIs from other LLM service providers such as Google's Gemini API, Anthropic's Claude API, and xAI's Grok API should be integrated and benchmarked.

However, the high variance in responses generated by LLMs and LLM systems indicates that benchmarking LLMs should be done with great care. When few samples are used for score assessment, it is possible that the variance in the response out-scales any contributions from proposed innovations. A new metric should be developed for policy recommendation since full-reference text assessment metrics are too influenced by writing style alignment, whereas a measure of the semantic anti-alignment is more important for practical purposes (I.e. measure the number and degree of false recommendations).

Additionally, a more rigorous way to perform statistical analysis using LLMs should

be developed. Although the analysis generated by the LLM system in Section 5.3.4 can suffice for some purposes, it does not have scientific rigour. My preliminary experiments in February 2025 showed that these models still struggled with hypothesis testing and calculating non-elementary mathematical operations and functions. An LLM system that can perform rigorous hypothesis testing and provide p-value estimates should be designed, perhaps by designing LLM systems that can call upon appropriate external statistical libraries. This will greatly improve data analysis efficiency experiments in many fields of science.

Although we used LLM models from LLM providers, who already took into consideration LLM-safety alignment, we wish to host and fine-tune local models in future experiments. In these cases, LLM-safety alignment should be of concern during deployment. This can be done via a combination of output filtering, system prompt engineering, and training data filtering.

5.6 Chapter Summary

In this chapter, I presented Digital Twin Buildings, a prototype digital twin system framework. The framework can extract the 3D mesh of a building, connect the building to Google Maps Platform APIs, and perform automated Multi-Agent Large Language model data analytics. I demonstrated this by extracting visual description keywords and captions of the building from multi-view-multi-scale images of the building. The framework can also be used to process different data modalities sourced from Google Cloud Services. This approach enables seamless integration with geospatial data with LLM-based data analysis, and enhanced interaction with real-world structures, paving the way for advanced applications in urban analytics, navigation, and virtual environments.

I also tested a potential real-life data analysis and policy recommendation scenario using a modified version of the multi-agent LLM system. This was done using the air quality history data of the devastating Los Angeles wildfires of January 2025, retrieved using the framework's Google Maps Platform integration using the LLM system's generated query in a one-shot manner. The generated recommendations were all positively aligned with ground truth with a minimum BERTScores of (P = 0.808, R = 0.778, F1 = 0.793). This case study showed the potential of LLM-based systems for aiding GIS analysis and integration into Urban Digital Twin frameworks, greatly improving user accessibility and large-scale data analysis potential.

Chapter 6

Conclusions and Recommendations for Future Research

6.1 Conclusions

This thesis advances research in urban digital twins by contributing to innovations in efficient, accurate, and user-accessible urban 3D modelling, data integration, and LLM-based automated data analytics. This thesis is comprised of three distinct research projects and a case study, each contributing novel advancements in their respective domains, as detailed in the corresponding chapters. These are summarized as follows.

Chapter 3 introduces an approach to large-scale 3D urban scene reconstruction and point cloud densification using Gaussian Splatting and Google Earth imagery. It is the first work to propose Google Earth Studio as a data source for remote sensing-based Gaussian Splatting training, exploring its performance across various scales and conditions in urban environments. It was shown to outperform non-Gaussian Splatting methods used in

traditional Urban Digital Twins in terms in terms of training speed, 3D accuracy, and 2D photorealism.

Focusing on precise 3D reconstruction at the local scale, Chapter 4 presents the Gaussian Building Mesh (GBM) pipeline, a method for extracting the 3D mesh of a building from its place name, address, postal code, or geographic coordinates. Key contributions include the integration of SAM2 with GroundingDINO for building mask extraction, improvements to the masking module for training on masked images, and a novel mask refinement procedure that enhances mask quality and geometry. This method generates an accurate meshed-based 3D representation more suited for numerical computations and simulations than 3D point clouds, outperforming previous methods. It is particularly suited for digital twin systems modelling at the local scale.

Working towards a prototype next-generation urban digital twin systems model, Chapter 5 introduces the Digital Twin Buildings framework, a user-accessible digital twin system that allows the user to use natural language to query data and perform data analysis (as opposed to computer language). The framework integrates 3D building modelling, cloud mapping geographic information systems, and multi-agent LLM-based automated data analytics. Leveraging the GBM pipeline from Chapter 4, the framework extracts 3D models from addresses, coordinates, or place names and integrates them with the Google Maps Platform and a newly designed LLM system built using LLM provider APIs like OpenAI and Deepseek. The framework's capabilities are demonstrated through experiments analyzing multi-view, multi-scale imagery using multi-agent LLMs. As an example of practical application, the LLM system is also used in a case study to analyze air quality data from Los Angeles during the January 2025 wildfires in Section 5.5.

Together, these projects constitute a fast, efficient, and accessible framework for urban digital systems modelling, contributing to the development of Urban Digital Twins. We

are in an exciting era of rapid AI development driven by recent advances in computer vision and natural language processing. I hope that the research in this thesis and the proposed research directions benefit future scholars by providing a blueprint for the next generation of efficient, accurate, and accessible Urban Digital Twins.

6.2 Recommendations for Future Research

As discussed in the respective chapters, many challenges remain to be overcome. However, these challenges also present new research opportunities in 3D computer vision, natural language processing and geographic information systems/science, as well as further improvement opportunities for the next generation of Urban Digital Twins.

6.2.1 3D Urban Reconstruction

Using memory efficient implementations of 3DGS such as [132–135] will allow for better reconstruction across multiple neighbourhoods by allowing for more Gaussian primitives. Better camera pose arrangement for large-scale scenes can also be investigated. For drone or aerial image footage, the camera pose arrangement must follow the platform's movement. What type of camera pose arrangement is best suited for 3D reconstruction via Gaussian Splatting is an important question to answer. Google Earth Studio, can be used to test different camera paths, even unrealistic ones such as multiple spirals arranged hierarchically, centred around each neighbourhood of interest, or approximate space-filling curves filling out a camera path with dense coverage across the entire large-scale scene could be tested. These methods can be combined with Gaussian-Splatting methods built for large-scale scenes such as [58]. Alternatively, a large-scale 3D reconstruction scheme which pieces

together multiple local models following the Mega-NeRF [53] framework can be considered. Another future research direction is remote sensing-based large-scale semantics-based 3D reconstruction and semantic synthesis for semantically aware 3D maps. For urban scenes, this research area is expected to find applications in urban digital twin creation, urban monitoring, and urban/land-use planning. By integrating Gaussian Splatting methods with object semantics understanding, this research direction can also more generally extend land-use/land-cover segmentation to three dimensions, which has a multitude of research and commercial applications.

For generating the 3D mesh of individual buildings, the TSDF–Marching Cubes pipeline is the main weakness. Gaussian Splatting already generates a point cloud-based 3D representation. However, TSDF meshing requires the rendered depth maps from Gaussian Splatting models as input. The rendered depth maps and the TSDF mesh are highly sensitive to noise points and floaters, resulting in difficult-to-denoise mesh distortions and mesh noise. The generation of 3D mesh from Gaussian Splatting models should be further investigated; whether in the form of novel meshing pipelines, improvements to the current TSDF–Marching Cubes pipeline, or novel primitives representations.

The improvements to the foundational computer vision models for open-set segmentation and object detection can also greatly improve important aspects of Gaussian Splatting research, especially the research areas focusing on 3D segmentation. By being usable out-of-the-box on any new dataset, and by not suffering from out-of-distribution generalizability (at least compared to supervised models), these models are much more useful in real-life applications than the thousands of supervised models which have been benchmarked on commonly used datasets but have never been used in practical applications. Although SAM2 and Grounding-DINO have proven to be of great use in the GBM research in Chapter 4, Further performance improvement of these 2D foundational models

should be investigated.

No-reference (unsupervised) quality assessment metrics for 3D point clouds, 3D meshes, and text generation warrant further research. I personally believe machine learning and deep learning research have relied too heavily on supervised datasets for benchmarking. In all three research projects, many models/frameworks required performance quantification without supervised datasets, leading me to use less common metrics and unconventional benchmarking pipelines (e.g., assessing 3D mesh quality with 3D-SSIM comparing against 2D images). While these solutions could distinguish between "good" and "bad" models, I find them unsatisfactory due to weak theoretical justification and insufficient empirical evidence linking assessment scores to true performance.

6.2.2 LLM-based Geospatial Data Analysis

A future research direction is to leverage multi-agent LLMs to perform geospatial data analysis using different data sources from different Google Cloud Platform services including various Google Maps Platform and Google Earth Engine APIs, as well as social media data streams (e.g. connecting to Twitter/X or other social media APIs). Using LLM-based data analytics for decision-making or policy-making is particularly worth exploring. LLMs can quickly parse and summarize a large amount of data and produce human-readable summaries and actionable recommendations.

Another compelling research direction is benchmarking and fine-tuning the reasoning capabilities of advanced reasoning-based large language models, such as GPT-o1/o3 and Deepseek-R1, for remote sensing and GIS tasks. These models have demonstrated graduate-level proficiency in solving problems across science, technology, engineering, and mathematics. Testing and improving the spatial reasoning capabilities of these advanced

models could be particularly interesting.

The ability to perform rigorous statistical tests using LLMs via natural language prompting is extremely useful. As of my current experiments, although LLMs can find trends and outliers, they still struggle to perform correct rigorous statistical tests and provide p-values. Designing hybrid systems that allow LLMs to automatically call upon statistical libraries to perform proper hypothesis testing would be of great help to scientific research.

Better LLM metrics for text sentiment assessment should be developed. Although there has been a wide variety of research on LLM problem-solving capabilities and assessment thereof, simple text generation metrics are still too influenced by the writing style. For many applications where the tone and style are unimportant, a pure semantic alignment-based metric is desirable.

And finally, LLM systems are easily adaptable to different tasks. Building LLM systems for the next generation of user-accessible digital twin systems in multiple domains should be explored.

References

[1] H. Lehner and L. Dorffner, "Digital geoTwin Vienna: Towards a digital twin city as geodata hub," 2020.

[2] G. Schrotter and C. Hürzeler, "The digital twin of the city of Zurich for urban planning," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 88, no. 1, pp. 99–112, 2020.

[3] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert systems with applications*, vol. 165, p. 113816, 2021.

[4] F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin, "Applications of 3D city models: State of the art review," *ISPRS International Journal of Geo-Information*, vol. 4, no. 4, pp. 2842–2889, 2015.

[5] Z. Li and H. Ning, "Autonomous GIS: The next-generation AI-powered GIS," *Int. J. Digit. Earth*, vol. 16, no. 2, pp. 4668–4686, 2023.

[6] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3D Gaussian splatting as new era: A survey," *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[7] K. Carolan, L. Fennelly, and A. F. Smeaton, "A review of multi-modal large language and vision models," *arXiv preprint arXiv:2404.01322*, 2024.

[8] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "Nerf: Neural radiance field in 3d vision, a comprehensive review," *arXiv preprint arXiv:2210.00379*, 2022.

[9] J. Ferré-Bigorra, M. Casals, and M. Gangolells, "The adoption of urban digital twins," *Cities*, vol. 131, p. 103905, 2022.

[10] B. Lei, P. Janssen, J. Stoter, and F. Biljecki, "Challenges of urban digital twins: A systematic review and a delphi expert survey," *Automation in Construction*, vol. 147, p. 104716, 2023.

[11] U. Ramakarl, "File:Mesh overview.svg - Wikimedia Commons — commons.wikimedia.org," https://commons.wikimedia.org/wiki/File:Mesh_overview.svg, 2009, [Accessed 25-02-2025].

[12] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[13] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[14] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.

[15] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Springer, 2000, pp. 298–372.

[19] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[20] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d Gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[21] M. Zwicker, H. Pfister, J. Van Baar, and M. Gross, "EWA splatting," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 223–238, 2002.

[22] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510.

[25] "M4: Spherical Harmonics — chem.libretexts.org," https://chem.libretexts.org/Ancillary_Materials/Reference/Reference_Tables/Mathematical_Functions/M4%3A_Spherical_Harmonics, [Accessed 16-02-2025].

[26] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d Gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 Conference Papers*, 2024, pp. 1–11.

[27] A. Waswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[28] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, and S. R. Bowman, "Gpqa: A graduate-level google-proof q&a benchmark," *arXiv preprint arXiv:2311.12022*, 2023.

[29] Z. Liu, Y. Chen, M. Shoeybi, B. Catanzaro, and W. Ping, "AceMath: Advancing Frontier Math Reasoning with Post-Training and Reward Modeling," *arXiv preprint arXiv:2412.15084*, 2024.

[30] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[31] Alphabet Inc., "Google Earth Studio," 2015-2024. [Online]. Available: https://www.google.com/earth/studio/

[32] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer, "A survey of structure from motion*." *Acta Numerica*, vol. 26, pp. 305–364, 2017.

[33] N. Snavely, "Bundler: Structure from motion (sfm) for unordered image collections," 2008. [Online]. Available: http://phototour.cs.washington.edu/bundler/

[34] M. Adorjan, "Opensfm: A collaborative structure-from-motion system," Ph.D. dissertation, Wien, 2016.

[35] C. Reality, *RealityCapture Software*, 2024, bratislava, Slovakia. Available: https://www.capturingreality.com.

[36] H. G. Solutions, *ENVI Software Suite*, 2024, boulder, CO, USA. Available: https://www.l3harrisgeospatial.com/Software-Technology/ENVI.

[37] Esri, *ArcGIS Software Suite*, 2024, redlands, CA, USA. Available: https://www.esri.com/arcgis.

[38] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5855–5864.

[39] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.

[40] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM transactions on graphics*, vol. 41, no. 4, pp. 1–15, 2022.

[41] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *Proceedings of the European Conference on Computer Vision*. Springer, 2022, pp. 333–350.

[42] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[43] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *arXiv preprint arXiv:2106.10689*, 2021.

[44] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu, "Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3295–3306.

[45] Q. Fu, Q. Xu, Y. S. Ong, and W. Tao, "Geo-neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3403–3416, 2022.

[46] Y. Wang, I. Skorokhodov, and P. Wonka, "Hf-neus: Improved surface reconstruction using high-frequency details," *Advances in Neural Information Processing Systems*, vol. 35, pp. 1966–1978, 2022.

[47] M. Oechsle, S. Peng, and A. Geiger, "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5589–5599.

[48] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

[49] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5470–5479.

[50] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[51] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, "Urban radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 932–12 942.

[52] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, "Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering," in *Proceedings of the European Conference on Computer Vision*. Springer, 2022, pp. 106–122.

[53] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 922–12 931.

[54] D. Derksen and D. Izzo, "Shadow neural radiance fields for multi-view satellite photogrammetry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1152–1161.

[55] R. Marí, G. Facciolo, and T. Ehret, "Sat-nerf: Learning multi-view satellite photogrammetry with transient objects and shadow modeling using rpc cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 1311–1321.

[56] H. Zhou, J. Shao, L. Xu, D. Bai, W. Qiu, B. Liu, Y. Wang, A. Geiger, and Y. Liao, "Hugs: Holistic urban 3d scene understanding via Gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 336–21 345.

[57] Z. Li, Y. Zhang, C. Wu, J. Zhu, and L. Zhang, "Ho-Gaussian: Hybrid optimization of 3d Gaussian splatting for urban scenes," in *Proceedings of the European Conference on Computer Vision*. Springer, 2025, pp. 19–36.

[58] B. Kerbl, A. Meuleman, G. Kopanas, M. Wimmer, A. Lanvin, and G. Drettakis, "A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets," *ACM Transactions on Graphics*, vol. 44, no. 3, 2024.

[59] Y. Wang, X. Tang, J. Ma, X. Zhang, C. Zhu, F. Liu, and L. Jiao, "Pseudo-Viewpoint Regularized 3D Gaussian Splatting For Remote Sensing Few-Shot Novel View Synthesis," in *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2024, pp. 8660–8663.

[60] M. Chen, D. Lal, Z. Yu, J. Xu, A. Feng, S. You, A. Nurunnabi, and Y. Shi, "Large-Scale 3D Terrain Reconstruction Using 3D Gaussian Splatting for Visualization and Simulation," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 48, pp. 49–54, 2024.

[61] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[62] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. Sajjadi, A. Geiger, and N. Radwan, "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5480–5490.

[63] M. Chen, Q. Hu, Z. Yu, H. Thomas, A. Feng, Y. Hou, K. McCullough, F. Ren, and L. Soibelman, "Stpls3d: A large-scale synthetic and real aerial photogrammetry 3d point cloud dataset," *arXiv preprint arXiv:2203.09065*, 2022.

[64] Z. Yu, T. Sattler, and A. Geiger, "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–13, 2024.

[65] Y. Wolf, A. Bracha, and R. Kimmel, "GS2Mesh: Surface Reconstruction from Gaussian Splatting via Novel Stereo Views," in *Proceedings of the European Conference on Computer Vision*, 2024.

[66] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, no. 2. Minneapolis, Minnesota, 2019.

[67] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[68] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[69] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[70] A. Radford, "Improving language understanding by generative pre-training," 2018.

[71] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[72] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[73] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[74] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[75] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[76] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "Deepseek-V3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.

[77] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," *arXiv preprint arXiv:2501.12948*, 2025.

[78] Meta, "Llama 3.2: Revolutionizing edge AI and vision with open, customizable models — ai.meta.com," https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/, [Accessed 16-02-2025].

[79] K. Gao, D. Lu, H. He, L. Xu, J. Li, and Z. Gong, "Enhanced 3d urban scene reconstruction and point cloud densification using gaussian splatting and google earth imagery," *IEEE Transactions on Geoscience and Remote Sensing*, 2025.

[80] Statistics Canada, "2021 Census of Population," *Statistics Canada Catalogue*, 2023. [Online]. Available: https://www12.statcan.gc.ca/census-recensement/2021/dp-pd/prof/index.cfm?Lang=E

[81] Y. Xiangli, L. Xu, X. Pan, N. Zhao, A. Rao, C. Theobalt, B. Dai, and D. Lin, "Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering," in *Proceedings of the European Conference on Computer Vision*, 2022, pp. 106–122.

[82] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Proceedings of the European Conference on Computer Vision*. Springer, 2016, pp. 501–518.

[83] E. Zheng, E. Dunn, V. Jojic, and J.-M. Frahm, "Patchmatch based joint view selection and depthmap estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1510–1517.

[84] C. Ye, "2d-Gaussian-splatting-great-again," https://github.com/hugoycj/2d-Gaussian-splatting-great-again/tree/main, 2024.

[85] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou, "Absgs: Recovering fine details in 3d Gaussian splatting," in *Proceedings of the 32nd ACM International Conference on Multimedia*, 2024, pp. 1053–1061.

[86] Z. Zhang, W. Hu, Y. Lao, T. He, and H. Zhao, "Pixel-gs: Density control with pixel-aware gradient for 3d Gaussian splatting," *arXiv preprint arXiv:2403.15530*, 2024.

[87] L. Fan, Y. Yang, M. Li, H. Li, and Z. Zhang, "Trim 3D Gaussian Splatting for Accurate Geometry Representation," *arXiv preprint arXiv:2406.07499*, 2024.

[88] R. Liu, R. Xu, Y. Hu, M. Chen, and A. Feng, "AtomGS: Atomizing Gaussian Splatting for High-Fidelity Radiance Field," *arXiv preprint arXiv:2405.12369*, 2024.

[89] C. Ye, L. Qiu, X. Gu, Q. Zuo, Y. Wu, Z. Dong, L. Bo, Y. Xiu, and X. Han, "Stablenormal: Reducing diffusion variance for stable and sharp normal," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–18, 2024.

[90] K. Cheng, X. Long, K. Yang, Y. Yao, W. Yin, Y. Ma, W. Wang, and X. Chen, "Gaussianpro: 3d Gaussian splatting with progressive propagation," in *International Conference on Machine Learning*, 2024.

[91] S. S. Mallick, R. Goel, B. Kerbl, M. Steinberger, F. V. Carrasco, and F. De La Torre, "Taming 3dgs: High-quality radiance fields with limited resources," in *SIGGRAPH Asia 2024 Conference Papers*, 2024, pp. 1–11.

[92] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, "Dust3r: Geometric 3d vision made easy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 697–20 709.

[93] Y. ZHOU, Z. ZHANG, W. SUN, X. MIN, and G. ZHAI, "Perceptual quality assessment for point clouds: A survey," *ZTE Communications*, vol. 21, no. 4, p. 3, 2023.

[94] A. N. Netravali, *Digital pictures: representation, compression, and standards*. Springer, 2013.

[95] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[96] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5855–5864.

[97] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "Nerf: Neural radiance field in 3d vision, a comprehensive review," *arXiv preprint arXiv:2210.00379*, 2022.

[98] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, pp. 1–13, 2013.

[99] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[100] K. Gao, L. Li, H. He, D. Lu, L. Xu, and J. Li, "Gaussian Building Mesh (GBM): Extract a Building's 3D Mesh with Google Earth and Gaussian Splatting," *arXiv preprint arXiv:2501.00625*, 2024.

[101] G. Chen and W. Wang, "A survey on 3d Gaussian splatting," *arXiv preprint arXiv:2401.03890*, 2024.

[102] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[103] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *Proceedings of the European Conference on Computer Vision*. Springer, 2025, pp. 38–55.

[104] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[105] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded SAM: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.

[106] A. Guédon and V. Lepetit, "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*, 2024, pp. 5354–5363.

[107] T. Sølund, A. G. Buch, N. Krüger, and H. Aanæs, "A large-scale 3D object recognition dataset," in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 2016, pp. 73–82.

[108] H. Zhao, H. Zhou, Y. Zhang, J. Chen, Y. Yang, and Y. Zhao, "High-frequency stereo matching network," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 1327–1336.

[109] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, pp. 7–42, 2002.

[110] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[111] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[112] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer graphics and image processing*, vol. 1, no. 3, pp. 244–256, 1972.

[113] K. Zeng and Z. Wang, "3D-SSIM for video quality assessment," in *2012 19th IEEE international conference on image processing*. IEEE, 2012, pp. 621–624.

[114] K. Gao, D. Lu, L. Li, N. Chen, H. He, L. Xu, and J. Li, "Digital twin buildings: 3d modeling, gis integration, and visual descriptions using gaussian splatting, chatgpt/deepseek, and google maps platforms," *arXiv preprint arXiv:2502.05769*, 2025.

[115] ——, "Instructor-worker large language model system for policy recommendation: a case study on air quality analysis of the january 2025 los angeles wildfires," *arXiv preprint arXiv:2503.00566*, 2025.

[116] O. Topsakal and T. C. Akinci, "Creating large language model applications utilizing LangChain: A primer on developing LLM apps fast," in *Proc. Int. Conf. Appl. Eng. Nat. Sci.*, vol. 1, no. 1, 2023, pp. 1050–1056.

[117] Y. Chen, W. Wang, S. Lobry, and C. Kurtz, "An LLM agent for automatic geospatial data analysis," *arXiv preprint arXiv:2410.18792*, 2024.

[118] A. M. Luthfi, N. Karna, and R. Mayasari, "Google maps API implementation on IOT platform for tracking an object using GPS," in *2019 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*. IEEE, 2019, pp. 126–131.

[119] A. Bhandari and R. Noone, "Support local: Google Maps' local guides platform, spatial power and constructions of "the local"," *Communication, Culture & Critique*, vol. 16, no. 3, pp. 198–207, 2023.

[120] H. Li and B. Hecht, "3 Stars on Yelp, 4 stars on google maps: a cross-platform examination of restaurant ratings," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW3, pp. 1–25, 2021.

[121] P. Fuquan, S. Jian, W. Wenyong, and W. Zebing, "A city modeling and simulation platform based on google map api," in *Proceedings of the 2011, International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19–20, 2011, Melbourne, Australia: Volume 2: Information Systems and Computer Engineering*. Springer, 2012, pp. 513–520.

[122] T. Ganor, "An integrated spatial search engine for maps and aerial photographs on a google maps API platform," *Journal of Map & Geography Libraries*, vol. 13, no. 2, pp. 175–197, 2017.

[123] F. Bordes, R. Y. Pang, A. Ajay, A. C. Li, A. Bardes, S. Petryk, O. Mañas, Z. Lin, A. Mahmoud, B. Jayaraman *et al.*, "An introduction to vision-language modeling," *arXiv preprint arXiv:2405.17247*, 2024.

[124] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, "Qwen2. 5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.

[125] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[126] C. Team, "Chameleon: Mixed-modal early-fusion foundation models," *arXiv preprint arXiv:2405.09818*, 2024.

[127] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with BERT," *arXiv preprint arXiv:1904.09675*, 2019.

[128] J. Li, D. Li, C. Xiong, and S. Hoi, "BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*. PMLR, 2022, pp. 12 888–12 900.

[129] S. Sarto, M. Barraco, M. Cornia, L. Baraldi, and R. Cucchiara, "Positive-augmented contrastive learning for image and video captioning evaluation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 6914–6924.

[130] T. D. of Forestry and F. Protection, "Palisades Fire — — fire.ca.gov," https://www.fire.ca.gov/incidents/2025/1/7/palisades-fire, [Accessed 10-02-2025].

[131] https://www.theguardian.com/profile/victoria bekiempis, "Los Angeles' Palisades and Eaton wildfires are now fully contained — theguardian.com," https://www.theguardian.com/us-news/2025/feb/01/los-angeles-palisades-eaton-fires-contained, [Accessed 10-02-2025].

[132] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis, "Reducing the Memory Footprint of 3D Gaussian Splatting," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 7, no. 1, pp. 1–17, 2024.

[133] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3d Gaussian representation for radiance field," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 719–21 728.

[134] K. Katsumata, D. M. Vo, and H. Nakayama, "A compact dynamic 3d Gaussian representation for real-time dynamic view synthesis," in *Proceedings of the European Conference on Computer Vision*. Springer, 2025, pp. 394–412.

[135] G. Song, C. Cheng, and H. Wang, "GVKF: Gaussian Voxel Kernel Functions for Highly Efficient Surface Reconstruction in Open Scenes," *arXiv preprint arXiv:2411.01853*, 2024.