Western Graduate&PostdoctoralStudies

Western University
Scholarship@Western

Electronic Thesis and Dissertation Repository

3-17-2023 10:30 AM

Autonomous 3D Urban and Complex Terrain Geometry Generation and Micro-Climate Modelling Using CFD and Deep Learning

Tewodros F. Alemayehu, *The University of Western Ontario*

Supervisor: Bitsuamlak, Girma T., *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Civil and Environmental Engineering
© Tewodros F. Alemayehu 2023

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Civil and Environmental Engineering Commons, Computational Engineering Commons, and the Computer Engineering Commons

Recommended Citation

Alemayehu, Tewodros F., "Autonomous 3D Urban and Complex Terrain Geometry Generation and Micro-Climate Modelling Using CFD and Deep Learning" (2023). *Electronic Thesis and Dissertation Repository*. 9273.
https://ir.lib.uwo.ca/etd/9273

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

Sustainable building design requires a clear understanding and realistic modelling of the complex interaction between climate and built environment to create safe and comfortable outdoor and indoor spaces. This necessitates unprecedented urban climate modelling at high temporal and spatial resolution. The interaction between complex urban geometries and the microclimate is characterized by complex transport mechanisms. The challenge to generate geometric and physics boundary conditions in an automated manner is hindering the progress of computational methods in urban design. Thus, the challenge of modelling realistic and pragmatic numerical urban micro-climate for wind engineering, environmental, and building energy simulation applications should address the complexity of the geometry and the variability of surface types involved in urban exposures. The original contribution to knowledge in this research is the proposed end-to-end workflow that employs a cutting-edge deep learning model for image segmentation to generate building footprint polygons autonomously and combining those polygons with LiDAR data to generate level of detail three (LOD3) 3D building models to tackle the geometry modelling issue in climate modelling and solar power potential assessment. Urban and topography geometric modelling is a challenging task when undertaking climate model assessment. This thesis describes a deep learning technique that is based on U-Net architecture to automate 3D building model generation by combining satellite imagery with LiDAR data. The deep learning model registered an overall accuracy of 98%. The extracted building polygons were extruded using height information from corresponding LiDAR data. The building roof structures were also modelled from the same point cloud data. The method used has the potential to automate the task of generating urban scale 3D building models and can be used for city-wide applications. The advantage of applying a deep learning model in an image processing task is that it can be applied to a new set of input image data to extract building footprint polygons for autonomous application once it has been trained. In addition, the model can be improved over time with minimum adjustments when an improved quality dataset is available, and the trained parameters can be improved further building on previously learned features. Application examples for pedestrian level wind in urban geometry and solar energy availability assessment as well as modelling wind flow over complex terrain are presented.

Keywords: Autonomous, Deep learning, LiDAR, Building footprint, Urban geometry, Complex terrain, satellite image, pedestrian level wind, solar energy, speed-up

Summary for Lay Audience

The three-dimensional (3D) geometry modelling process of cities and landscape is one of the major challenges in micro-climate studies. This research developed an autonomous workflow that can create 3D models using data collected satellite and laser devices. The workflow applied artificial intelligence principles and has shown a potential to shorten redundant tasks. The resulting models were used for solar power potential assessment in residential buildings, pedestrian level comfort assessment in urban area and for complex terrain analysis.

Autonomous 3D building modelling is a technology that uses computers and advanced algorithms to create digital 3D models of buildings automatically, without human intervention. This thesis used a deep learning model in the workflow to achieve autonomous 3D modelling. Traditionally, architects and engineers would create these 3D models manually, which can be time-consuming and expensive. With autonomous 3D building modelling, computers can use data from a variety of sources, such as satellite imagery, LiDAR (laser-based) scans, and photographs, to automatically generate highly detailed 3D models of buildings.

To estimate the solar power potential of a residential building, several factors are considered. One of the most important factors is the location of the building. Different regions receive different amounts of sunlight, which affects the amount of energy that can be generated from solar panels. Other factors that are considered include the orientation and tilt of the roof, the shading from trees or other buildings, and the size of the roof area that can be used for solar panels. All of these factors are taken into account to determine the optimal placement and number of solar panels that can be installed on the building.

Pedestrian level wind comfort assessment using Computational Fluid Dynamics (CFD) is a technique that is used to determine how comfortable or uncomfortable it is for people to walk

or stand in a particular outdoor space, based on the flow of air around them. CFD is a computer simulation technique that uses complex mathematical equations to model and analyze the flow of fluids.

Co-Authorship Statement

This thesis has been prepared in accordance with the regulations for an Integrated Article paper format stipulated by the School of Graduate and Postdoctoral Studies at the University of Western Ontario and has been co-authored as:

From **Chapter 3**, **“Autonomous urban topology generation for urban flow modelling”** is published in *Journal of Sustainable Cities and Society* under the co-authorship of Tewodros F. Alemayehu and Girma T. Bitsuamlak.

Acknowledgement

I would like to thank God for guiding my work with his blessings. And I would also like to thank Dr. Girma Bitsuamlak for his valuable contributions and support. I am also grateful for all the members of my research team for their advice and support throughout the years. I would like to also thank Professor Martha for her valuable advice. My friend Tesfaye, thanks for all the help. I am also grateful to my wife Mahi for all the support and love during the research. I would like to also give a special thanks to all my family, my father Fantahun, my mother Welela, my brothers Ena, Eyaya, Nuna, Awet and Emet for their support and encouragement during my study.

Table of Contents

Abstract ii

Keywords iii

Summary for Lay Audience iv

Co-Authorship Statement vi

Acknowledgement vii

Table of Contents viii

List of Tables xii

List of Figures xii

List of Abbreviations xviii

Chapter 1 1

Introduction 1

- 1.1 Background and Motivation 1
- 1.2 Research Gap 5
- 1.3 Objectives 7
  - 1.3.1 Specific Objectives 7
  - 1.3.2 Significance of the Study 8
- 1.4 Organization of thesis 10
  - 1.4.1 A deep learning model-based building footprint polygon extraction for a GIS-based Solar power potential estimation 11
  - 1.4.2 Autonomous urban topology generation for urban flow modelling 11
  - 1.4.3 Modelling wind flow over complex terrain generated by using LiDAR and SRTM 11
- Guiding framework 11

References 12

Chapter 2 16

A deep learning algorithm building footprint polygon extraction for a GIS-based Solar power potential estimation ... 16

Abstract ... 16

Keywords ... 16

2.1 Solar power potential ... 17

2.2. Automated building footprint polygon generation ... 22

2.3 Methodology ... 25

- 2.3.1 Morphological filter ... 28
  - 2.3.2 Morphological filter results ... 31
- 2.3.3 Machine learning approach ... 35
  - *I. Support Vector Machine classifier* ... 39
  - *II. Random Trees* ... 41
  - *III. Maximumlikelihood* ... 42
- 2.3.4 Machine learning results ... 44
- 2.3.5 Autonomous Extraction of Building Footprint polygons from high-resolution Satellite Image Data using U-Net based deep learning model ... 45
- 2.3.6 Deep learning model architecture ... 47
- 2.3.7 Training data preparation and analysis ... 51
- 2.3.8 Results for the U-Net-based model ... 56
- 2.3.9 Validation model ... 59

2.4. Solar power potential analysis for residential building ... 60

2.5 Conclusion ... 67

2.6 Reference ... 68

Chapter 3 ... 74

Autonomous urban topology generation for urban flow modelling ... 74

Abstract ... 74

3.1 Introduction 75

3.2 Methodology 90

3.3 Deep learning model for building footprint extraction 92

3.3.1 Training Data Preparation 92

3.3.2 Deep learning model architecture 96

3.4 3D building model generation by integrating building footprint and 3D Lidar point cloud data 101

3.5 Pedestrian Level Wind Assessment on Autonomously Generated 3D Building Models 103

3.5.1 Governing equations and boundary conditions 103

3.5.2 Computational domain 113

3.6 Results and discussion 113

3.6.1 Building Footprint extraction result 113

3.6.2 3D building model generation result 117

3.6.3 CFD modelling results 119

3.6.4 Validation 119

Conclusion 120

References 122

Chapter 4 128

Simulating wind flow over complex terrain created by using LiDAR and SRTM 128

Abstract 128

4.1 Introduction 129

4.2 Wind speed-up factors 134

4.3 Numerical simulation methodology 139

4.3.1 Topography CAD modelling 140

4.3.2 Grid sensitivity analysis 143

4.3.3 Boundary conditions ... 144

4.3.4 Solver setup ... 145

4.3.5 Assessing ABL homogeneity in an empty Computational Domain (CD) . 145

4.4 Results and discussion ... 146

Conclusion ... 156

Reference ... 157

Chapter 5 ... 163

Conclusions and Recommendations ... 163

5.1 Overview ... 163

5.2 Chapter 2: A deep learning model-based building footprint polygon extraction for a GIS-based Solar power potential estimation ... 163

5.3 Chapter 3: Autonomous urban topology generation for urban flow modelling ... 164

5.4 Chapter 4: Modelling wind flow over complex terrain generated by using LiDAR and SRTM ... 166

5.5 Limitations of the thesis ... 166

5.6 Future Research Directions ... 167

Appendix ... 170

Curriculum Vitae ... 203

Publications ... 203

List of Tables

Table 2. 1 Accuracy assessment of the SVM method 170

Table 2. 2 Accuracy assessment of the random trees method 170

Table 2. 3 Accuracy assessment of the maximum likelihood method..................................170

Table 3. 1 Inlet boundary conditions (Richards and Hoxey, 1993, Richards and Norris, 2011). 85

Table 3. 2 Accuracies of the segmentation results..92

Table 3. 3 Pedestrian wind comfort criteria (Adamek et al., 2017).......................................99

Table 4. 1 Parameters for maximum speed-up over hills and escarpments (NBCC, 2005) ..135

List of Figures

Figure 1. 1 LiDAR data visualizations 4

Figure 1. 2 General guiding framework of the research 12

Figure 2.1 Solar geometry parameters 20

Figure 2. 2 (a) General workflow (b) Building footprint polygon extraction methods evaluation 26

Figure 2. 3 Process of the progressive morphological filter to identify building measurements 31

Figure 2. 4 Morphological filter results for extracting building footprints 32

Figure 2. 5 3D view of the morphological filter result 34

Figure 2. 6 Morphological filter applied to satellite imagery 34

Figure 2. 7 The testing area in downtown London, Ontario 38

Figure 2. 8 SVM, random trees and Maximum likelihood segmentation results 45

Figure 2. 9 (a) The building classification workflow, (b) Deep learning model architecture . 46

Figure 2. 10 Training image labelling 52

Figure 2. 11 Training data masks 52

Figure 2. 12 Data augmentation sample results 55

Figure 2. 13 (a) Neural network training (b) Final accuracy (c) Polygons extraction 57

Figure 2. 14 Residential building footprints extracted by deep learning model 58

Figure 2. 15 Workflow of the rooftop solar power estimation 60

Figure 2. 16 Solar energy mapping 66

Figure 2. 17 Rooftop suitability identification 67

Figure 2. 18 Solar power calculation 67

Figure 3. 1 The Alan G. Davenport wind loading chain (Isyumov, 2012). 77

Figure 3. 2 Workflow for 3D modelling of buildings. 77

Figure 3. 3 Sample of training images used to train the deep learning network. 77

Figure 3. 4 Training image labelling. 78

Figure 3. 5 Sample training image with its corresponding mask. 78

Figure 3. 6 Data augmentation sample results. 79

Figure 3. 7 RGB and grayscale images with their respective dimensions. 79

Figure 3. 8 U-Net-based deep learning model architecture. 80

Figure 3. 9 3D LiDAR data used for model generation. 81

Figure 3. 10 Point clouds on the tallest building can help to filter out outliers and the lowest points are found on rivers usually. 81

Figure 3. 11 (a) normalized Digital Surface Model (nDSM), (b) Digital Surface Model (DSM) and (c) Digital Terrain Model (DTM) values of the LiDAR data. 82

Figure 3. 12 Summary of workflow for urban geometry generation and climate modelling.84

Figure 3. 13 Computational domain that used the automated urban topology. 87

Figure 3. 14 Prediction results from the deep learning model training. 88

Figure 3. 15 The predicted building footprints of downtown area in London, ON. 88

Figure 3. 16 A series of images showing the 3D model generation workflow transformations. 91

Figure 3. 17 3D building models generated using Deep learning technique. 93

Figure 3. 18 Grid discretization of the 3D building models 94

Figure 3. 19 RANS CFD simulation at 0- and 45-degree wind angle of attacks. 98

Figure 3. 20 RANS CFD simulation at 90- and 135-degrees wind angle of attacks. 105

Figure 3. 21 RANS CFD simulation 180- and 225-degrees wind angle of attacks. 106

Figure 3. 22 RANS CFD simulation at 270- and 315-degrees wind angle of attacks. 107

Figure 3. 23 Turbulent kinetic energy simulation at 0- and 45-degree wind angle of attacks. 108

Figure 3. 24 Turbulent kinetic energy simulation at 90- and 135-degrees wind angle of attacks. 109

Figure 3. 25 Turbulent kinetic energy simulation 180- and 225-degrees wind angle of attacks. 110

Figure 3. 26 Turbulent kinetic energy simulation at 270- and 315-degrees wind angle of attacks. 111

Figure 3. 27 Comparison of 3D models 112

Figure 3. 28 Comparison of velocity contour generated using predicted and actual models. 118

Figure 4. 1 Wind flow over simple and complex terrains 133

Figure 4. 2 Wind speed-up over hills and escarpments (NBCC, 2005) 136

Figure 4. 3 The overall workflow 140

Figure 4. 4 Elevation contours used to generate the computational domain for the CFD simulation. 141

Figure 4. 5 Computational domain with high-resolution grid on the bottom surface of Area 1-LiDAR. 141

Figure 4. 6 Computational domain with high-resolution grid on the bottom surface of Area 1-SRTM. 142

Figure 4. 7 Computational domain with high-resolution grid on the bottom surface of Area 2-LiDAR. 142

Figure 4. 8 Computational domain with high-resolution grid on the bottom surface of Area 2-SRTM. 143

Figure 4. 9 Empty computational domain 144

Figure 4. 10 Empty computational domain 146

Figure 4. 11 Area1 velocity contour of LiDAR image for a wind direction of 0^0 147

Figure 4. 12 Area1 velocity contour of SRTM image for a wind direction of 0^0 148

Figure 4. 13 Area1 speedup contour at 30 m from the ground for a wind direction of 0^0 148

Figure 4. 14 Area1 speedup and turbulence intensity comparison between LiDAR and SRTM for wind direction of 0^0 149

Figure 4. 15 Area1 velocity contour of LiDAR image for a wind direction of 90^0 150

Figure 4. 16 Area1 velocity contour of SRTM image for a wind direction of 90^0 **150**

Figure 4. 17 Area1 speedup contour at 30 m from the ground for a wind direction of 90^0 151

Figure 4. 18 Area1 speedup and turbulent intensity comparison between LiDAR and SRTM for wind direction of 90^0 151

Figure 4. 19 Area2 velocity contour of LiDAR image for a wind direction of 0^0 152

Figure 4. 20 Area2 velocity contour of SRTM image for a wind direction of 0^0 153

Figure 4. 21 Area2 speedup contour at 30 m from ground for a wind direction of 0^0 153

Figure 4. 22 Area2 speedup and turbulent intensity comparison between LiDAR and SRTM for wind direction of 0^0 154

Figure 4. 23 Area2 velocity contour of LiDAR image for a wind direction of 90^0 154

Figure 4. 24 Area2 velocity contour of SRTM image for a wind direction of 90^0 155

Figure 4. 25 Area2 speedup contour at 30 m from the ground for a wind direction of 90^0 155

Figure 4. 26 Area2 speedup and turbulence intensity comparison between LiDAR and SRTM for wind direction of 90^0 156

List of Appendices

Appendix A: LiDAR and image processing…………………………………………………...153

APPENDIX B: Machine learning accuracy results………………………………………168

APPENDIX C: Deep learning algorithm and solar potential analysis miscellaneous details…………………………………………………………………………………..170

List of Abbreviations

LiDAR – Light Detection and Ranging

DGPS – Differential Global Positioning System

IMU – Inertial Measurement Unit

LAS – LASer

ASPRS – American Society for Photogrammetry and Remote Sensing

CFD – Computational Fluid Dynamics

PLW – Pedestrian Level Wind

3D – Three Dimensional

CAD – Computer Aided Design

GPS – Global Positioning System

DSM – Digital Surface Model

GIS – Geographic Information Systems

PV – Photovoltaic

TIN – Triangulated Irregular Network

DTM – Digital Terrain Model

LIO – Land Information Ontario

SVM – Support Vector Machine

RGB – Red Green Blue

CNN – Convolutional Neural Networks

ReLU – Rectified Linear Unit

RMSProp – Root mean square propagation

VIA – VGG Image Annotator

JSON – JavaScript Object Notation

RANS - Reynolds-Averaged Navier Stokes

nDSM – normalized digital surface model

LOD – Level of Detail

GeoTIFF – Geographic Tagged Image File Format

SRTM – Shuttle Radar Topography Mission

Chapter 1

Introduction

1.1 Background and Motivation

With approximately 70% of the world's population expected to live in urban areas by 2050, and cities being one of the largest energy consumer groups and emitters of greenhouse gases, urban areas offer a large potential for energy efficiency improvement (Sola et al., 2020). Embracing sustainability and maintaining the resiliency of Canada's built environment against natural hazards is necessary to sustain the wellbeing and prosperity of our communities. Sustainable building design in Canada, the second-largest country in the world with diverse geography characterized by climate extremes, mostly revolves around energy efficiency and resiliency (i.e. ability to withstand the climate loads and re-bounce quickly from extreme climate interruptions). Recent changes in ecosystems have had a negative impact on the liveability of outdoor built environments. The collective effects of these changes in urban outdoor spaces challenge effective urban planning which aims to create successful and usable outdoor spaces and affect efficacy of indoor spaces. Among the determinants of outdoor environment quality, a high priority is given to wind and thermal environment (Shooshtarian et al., 2020).

For example, buildings in Canada consume 30% of the total energy and 50% of the electricity on an annual basis. Through optimal resilient and sustainable design, however, there is an opportunity to reduce a building's energy consumption as high as 80%. Sustainable design entails consideration and integration of climate responsiveness and resilient design/retrofit of building(s). Therefore, a clear understanding and realistic modelling of the complex interaction between the climate and the built environment is a prerequisite. This necessitates unprecedented urban microclimate modelling at high temporal and spatial resolution. Transient climate parameters are required at a particular building location, ventilation inlet, window/door opening, or a street level over a long duration. Outdoor thermal comfort could significantly affect the usage and success of urban places. Accordingly, it is recommended to be considered in both urban design and planning projects. Urbanisation has been recognised as a major factor in elevated daily temperature values in cities (Shooshtarian et al., 2020).

One of the major challenges in realistic and pragmatic numerical urban micro-climate modelling for wind engineering, environmental, and building energy simulation applications is the complexity of the geometry (topology) and the variability of surface types involved in urban exposures. Each building form and surface classification is individually and manually entered into a Computer Aided Design (CAD) model through on-site-observations and publicly available information, which are often very time consuming and less precise. If the study site is in complex terrain, the modelling of the terrain also present additional challenge. Accurate site and building-specific information are required to assess climate stressors such as wind, for example, during structural or environmental design.

The main motivation of this study is, therefore, developing a computational workflow with an automated, site-specific 3D urban topology and accurate terrain orography for micro-climate modelling. The microclimate modelling includes various climate parameters such wind, temperature, solar, humidity etc. Here we will focus on wind flow modelling and solar resources. Workflows were developed to model wind flows in urban topology (e.g. pedestrian level wind assessment) and complex terrains, the use of the automated urban building footprint polygons generation for solar energy resource assessment in residential neighborhoods will also investigated.

One of the deciding parameters for city design is a pedestrian assessment of discomfort due to accelerated wind or lack of wind caused by local climate interaction with features like terrain, buildings, and vegetation (Chen and Mak, 2021). Surface roughness is a critical parameter for determining these aerodynamic impacts on people and buildings (Fan et al., 2022). Conveniently, remote sensing technology allows an effective way to estimate the surface roughness for a large area (Mosadegh and Nolin, 2022). Two types of remote sensing data can be used. The satellite images and Light Detection and Ranging (LiDAR) data. LiDAR remote sensing method provides direct measurements of the horizontal coordinates and vertical elevations of the objects on the surface of the Earth (Bui et al., 2022). It measures distances (ranges) based on the time between transmitting and receiving laser signals (Rutzinger, 2009). A LiDAR dataset includes measurements for all earth surface features scanned by a laser sensor.

A typical airborne LiDAR system is composed of a laser scanner; a ranging unit; control, monitoring, and recording units; a differential global positioning system (DGPS); and an inertial measurement unit (IMU) (Roriz et al., 2021). An integrated DGPS/IMU system is also called a position and orientation system that generates accurate position (longitude, latitude, and altitude) and orientation (roll, pitch, and heading) information. The collected LiDAR data is originally in LASer (LAS) format which is an industry-standard binary format for storing airborne LiDAR data. LiDAR has the following unique advantages that make it suitable for generating urban topology or complex terrain: LiDAR provides an efficient and reliable way to survey large-scale urban scenes (Li et al., 2019), its measurements are not influenced by sun shadow and relief displacement (Lee et al., 2006), and it provides georeferenced data that can be applied directly for most applications. LiDAR has also the following limitations: A basic characteristic of aerial LiDAR data is that information on roof structures of buildings is present in the data, but wall information is incomplete or missing, and Airborne LiDAR usually exhibits noise as well as non-uniform point densities (Espineira et al., 2021).

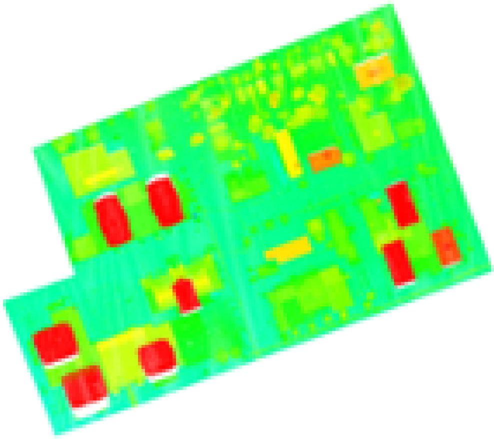


Figure 1. 1 LiDAR data visualizations

LiDAR data is a collection of points in three-dimensional space. The red points in Figure 1.1 show higher elevation points, mostly buildings while the yellowish-coloured points show medium-height buildings and trees, and the green-coloured points indicate ground points.

The LiDAR's laser pulses can be measured by radar(range) equation (Wu et al., 2021).

$$\boldsymbol{P_r = \frac{P_E G}{4\pi R^2}\frac{\sigma}{4\pi R^2}\frac{\pi D^2}{4}\eta_{sys}\eta_{atm}} \qquad \text{Eq.1.1}$$

where P_r is the received laser energy, P_E is the transmitted laser energy, G is the gain factor of the antenna, R is the range between target and sensor, σ is the effective target cross-section, D

is the receiver aperture diameter, η_{sys} is the system transmission factor, and η_{atm} is the atmospheric transmission factor.

Based on the American Society for Photogrammetry and Remote Sensing (ASPRS), LiDAR data is commonly used as a LASer (LAS) file format. So far, the previous paragraphs explained what LiDAR is and what it can be used for. The point cloud data is used with images to generate 3D building models and complex terrain in this thesis. Then, used to define a Computational Fluid Dynamics (CFD) domain for Pedestrian Level Wind (PLW) assessment or wind flow over a complex terrain. The roof geometry was also used to assess solar energy resources assessments in residential neighborhoods.

1.2 Research Gap

Climate studies in general and aerodynamic studies in particular depend on the wind climate, upstream terrain, study building shape among parameters. These varies from one study and to another and from one wind direction to another. Assessment methods such as wind tunnel or CFD that utilize specific exposure and study building shape are suitable to capture the effect all these geometric (roughness or aerodynamics). However, producing detail geometrical building of a study site in specific manner are challenging and engineering time intensive. These coupled with the challenges associated with turbulence modelling makes computational approaches challenging (Dagnew and Bitsuamlak, 2014). Realistic 3D building models required for CFD studies are not always readily available (Wang et al., 2018). Manual CAD modelling approach may achieve the required level of accuracy but it is an engineering time-consuming procedure (Dawes et al., 2001). The time-consuming process escalates when the study area is large. Even for a single-building model, the work may be tedious due to the complexity of building geometry. Importing and fixing CAD models to make them watertight takes a longer time and guaranteeing surface integrity during mesh generation is difficult (Dawes et al., 2001). Also, the difficulty in generating an optimal grid around a complex geometry hinders the practical use of the CFD in engineering analysis and design (Nakahashi et al., 2003). In turbulent flows budgeting the grids is of paramount consideration (Tsinuel and Bitsuamlak, 2022).

In methodologies that use satellite imagery, the most obvious shortcoming of this method is that the image only displays the top part of buildings and some critical side details including height information are usually missing (Sowmya and Trinder, 2000). Additionally, the resolution may not be satisfactory for specific purposes that require higher-resolution data. Unlike LiDAR, which directly collects an accurately georeferenced set of dense point clouds that can be immediately applied to basic operations, traditional photogrammetric methods do not offer this option (Chen et al., 2004).

The airborne LiDAR methodology approach gives a more precise representation of building side structures compared to any other input data. However, if the height position from which the point cloud data obtained is limited, building details will be missing in parts where a higher resolution is required (Holmgren et al., 2003). LiDAR data may incur horizontal errors, where points may be recorded off their original location due to Global Positioning System (GPS) and navigation unit operation anomalies (Park and Guldmann, 2019). Airborne LiDAR data is also more prone to suffer from missing data due to the absorption or reflection of laser energy (Minato et al., 1998). The interaction between complex urban geometries and the microclimate is characterized by complex transport mechanisms. The difficulty to generate geometric and physics boundary conditions in an automated manner is hindering the progress of computational methods in urban design. LiDAR may contain: noise that will affect the final output classification feature extraction phase (Gao et al., 2013) and usually operate at a monochromatic wavelength measuring the range and the strength of the reflected energy (intensity) from objects limiting the recording of a diversity of spectral reflectance (Morsy et al., 2017). In addition, using unclassified point clouds with building footprints to estimate building heights may yield erroneous results due to potential errors and anomalies in both datasets and their integration. Some of the points within footprints may often reflect irrelevant objects other than roofs, leading to biases in height estimation, and few studies have developed systematic methods to filter them out (Park and Guldmann, 2019). In consultation, there seem to be shortcomings in automated workflows for urban topology and complex terrain geometric modelling suitable for microclimate modelling. The use of numerical micro-climate studies in practice is limited. Many of the urban flow and complex terrain flows are done on oversimplified geometries and by using over simplified numerical turbulence models. (Abdi Bitsuamlak, 2019, Bitsuamlak et al. 2004).

1.3 Objectives

The short-term main objective of this research is to develop automated site-specific 3D urban and complex terrain topology and micro-climate modelling for wind engineering and building science applications.

1.3.1 Specific Objectives

To achieve the main objective the following specific objectives are identified:

- Automate two dimensional (2D) building footprint polygons generation process by utilizing deep learning methods

- Automate urban-scale 3D building model and complex terrain generation process based on data obtained from LiDAR data and digital images (airborne and satellite)
- Novel methods for residential buildings solar power potential estimation
- Novel high performance computing (HPC) based multi-scale urban climate model development to assess pedestrian wind level assessment
- Novel HPC based multi-scale wind flow model for complex terrain

1.3.2 Significance of the Study

Improved 3D building model generation method: This research provides an alternative low-cost method to generate 3D building models. Traditionally, the most direct route to obtain 3D building models relies on manual land surveying. Manual delineation of a city-scale scene is a labour-intensive and time-consuming task, and it is thus expensive for large-scale modelling.

Improved PLW comfort assessment process: The comfort and safety of pedestrians is increasingly being affected by wind due to the rise of super-tall buildings in cities, which in turn is the result of increasing population size. Thus, this research addresses the wind impact on pedestrians because of the interactions between buildings and the environment. The 3D building models will help to demonstrate how tall buildings can alter wind flow through automated CFD models. The 3D visualization meshed with the computational means will point out the relationship between wind, built environment and pedestrians while enabling stakeholders to make sustainable building designs in the process.

Improved wind flow over complex terrain: This research enables modelling of wind flow over complex flows for cases that are not covered in building codes and standards. By accurately modelling the terrain geometry through the state-of-the-art LiDAR cloud data, complex flows such as funneling, 3D dimensional and steep slope effects are assessed.

LiDAR technology provides many advantages over satellite images that can justify its use even though it is more expensive. Here are a few reasons why:

Higher resolution and accuracy: LiDAR produces highly accurate and detailed 3D point cloud data that can capture features as small as a few centimeters. In contrast, satellite images have limitations in terms of their spatial resolution, which can be affected by cloud cover and atmospheric conditions. Therefore, LiDAR can provide more precise information about the terrain, structures, and other objects.

Ability to penetrate through vegetation and structures: LiDAR can penetrate through vegetation and structures to provide accurate data about the ground surface and underlying features. In contrast, satellite images may be obstructed by vegetation or buildings, making it difficult to see the underlying features.

Flexibility in data acquisition: LiDAR can be mounted on various platforms such as aircraft, drones, and ground vehicles to collect data in different environments and terrain types. This flexibility in data acquisition allows for tailored data collection and the ability to collect data in areas that may be difficult to access with satellites.

Wide range of applications: LiDAR data can be used for a wide range of applications, such as topographic mapping, floodplain modelling, forest inventory, urban planning, and infrastructure assessment. The high resolution and accuracy of LiDAR data can provide critical information for decision-making processes in these fields.

In summary, while LiDAR technology may be more expensive than satellite images, its advantages in terms of resolution, accuracy, ability to penetrate through vegetation and structures, flexibility in data acquisition, and range of applications can justify its use in certain scenarios where high-quality data is required.

Additional contributions: This proposed research is beneficial for solar power potential assessment, air pollution dispersion studies, viewshed assessment, building inventory/population distribution development, and point cloud documentation for smart city applications. Furthermore, stakeholders can use this research to predict optimized building model shapes that can be obtained from the geometry generation algorithm for the construction of future buildings.

1.4 Organization of thesis

The major contribution to knowledge of this thesis is an autonomous workflow that employs a cutting-edge deep learning model for image segmentation to generate building footprint polygons autonomously and combining those polygons with LiDAR data to generate level of detail three (LOD3) 3D building models to tackle the geometry modelling issue in climate modelling and solar power potential assessment has been developed. Also a novel residential buildings solar power potential estimation method has been developed. In addition, accurate CFD models were generated by using high resolution LiDAR data and publicly available Shuttle Radar Topography Mission (SRTM) data in Geographic Tagged Image File Format (GEOTIFF) format for wind flow over complex terrain analysis.

This thesis is prepared based on the “Integrated-Article” format. The thesis contains a compilation of papers under review or published in peer-reviewed journals. Each paper addresses one of the research objectives identified in the previous section. The research is pursued in three prominent themes.

1.4.1 A deep learning model-based building footprint polygon extraction for a GIS-based Solar power potential estimation

Chapter 2: This part of the research explains how the deep learning model was utilized to extract building footprint polygons from satellite imagery. These building footprints were combined with LiDAR data to generate elevation models for solar power potential estimation.

1.4.2 Autonomous urban topology generation for urban flow modelling

Chapter 3: In this part of the research, the deep learning model is applied to generate urban building footprint polygons. Then, those footprints and corresponding LiDAR data were used to generate 3D models of buildings that defined the computational domain for computational fluid dynamics analysis of pedestrian level wind.

1.4.3 Modelling wind flow over complex terrain generated by using LiDAR and SRTM

Chapter 4: In this study accurate CFD models by using high resolution LiDAR data and publicly available SRTM data is developed. Impact of geometric modelling accuracy is discussed. Wind speed up values are estimated for complex terrain cases that are not covered in building codes and standards.

Chapter 5: Here summary, conclusion and future studies are discussed.

Guiding framework

This research used LiDAR and satellite images as an input for achieving the objectives of the study (Fig. 1.2). Morphological filter was initially used on LiDAR data to extract features. Machine learning and deep learning techniques were also tested for building footprint feature extraction. The deep learning model was considered for further analysis due to its high efficiency. The resulting building footprint polygons were used to generate digital elevation model by combining them with point cloud data for solar power potential assessment. LiDAR was also combined with 2D polygons to create 3D building models for PLW CFD analysis. Also, a comparison of a satellite image-based model and LiDAR based complex terrain analysis was carried out.

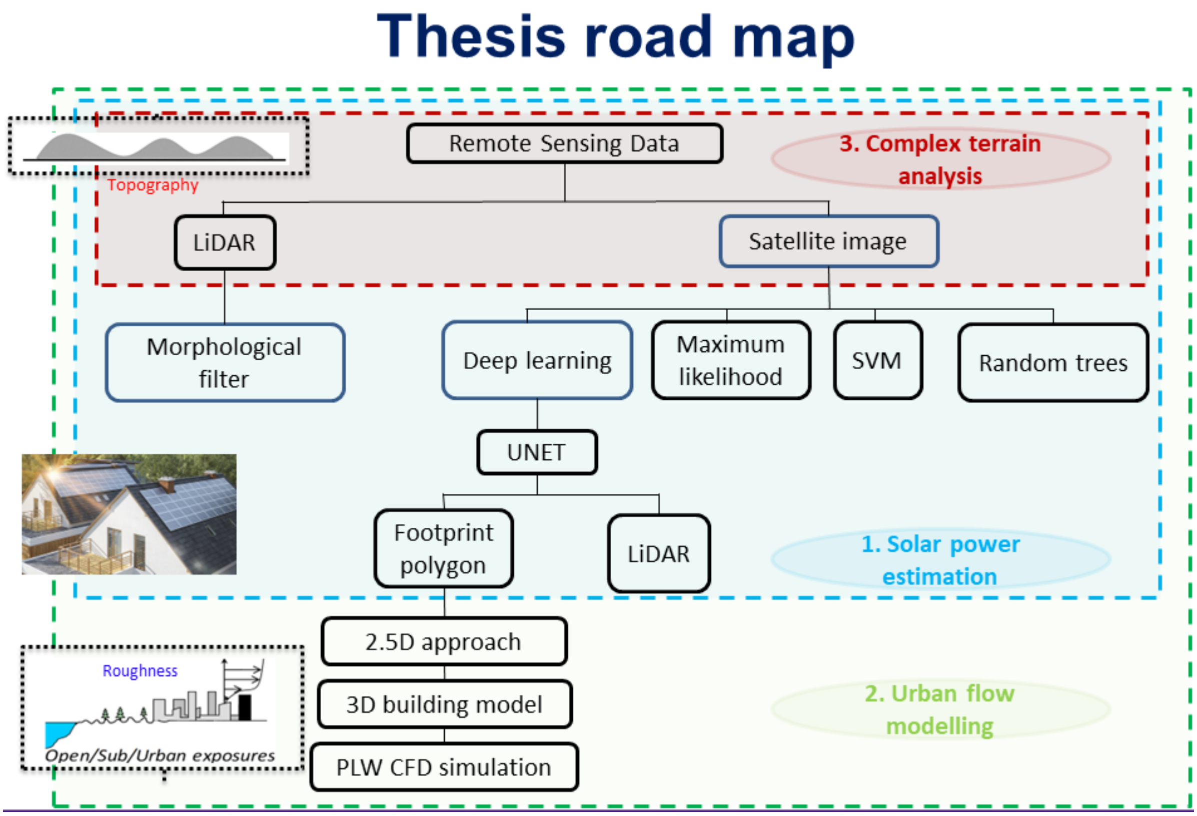


Figure 1. 2 General guiding framework of the research

References

Bui, L.K., Glennie, C.L., Hartzell, P.J., 2022. Rigorous Propagation of LiDAR Point Cloud Uncertainties to Spatially Regular Grids by a TIN Linear Interpolation. IEEE Geosci.

Remote Sens. Lett. 19, 1–5. https://doi.org/10.1109/LGRS.2021.3134587

Chen, L., Mak, C.M., 2021. Numerical evaluation of pedestrian-level wind comfort around “lift-up” buildings with various unconventional configurations. Build. Environ. 188, 107429. https://doi.org/10.1016/j.buildenv.2020.107429

Chen, L.Y., Teo, T., Rau, J., 2004. Fusion of lidar data and optical imagery for building modelling. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch. 35.

Dawes, W.N., Dhanasekaran, P.C., Demargne, A.A.J., Kellar, W.P., Savill, A.M., 2001. Reducing bottlenecks in the CAD-to-mesh-to-solution cycle time to allow CFD to participate in design. J. Turbomach. 123, 552–557. https://doi.org/10.1115/1.1370162

Espineira, J.P., Robinson, J., Groenewald, J., Chan, P.H., Donzella, V., 2021. Realistic LiDAR with Noise Model for Real-Time Testing of Automated Vehicles in a Virtual Environment. IEEE Sens. J. 21, 9919–9926. https://doi.org/10.1109/JSEN.2021.3059310

Fan, M.Y., Li, W.J., Luo, X.L., Shui, Q.X., Jing, L.Z., Gu, Z.L., Yu, C.W., 2022. Parameterised drag model for the underlying surface roughness of buildings in urban wind environment simulation. Build. Environ. 209. https://doi.org/10.1016/j.buildenv.2021.108651

Gao, F., Veberič, D., Stanič, S., Bergant, K., Hua, D.X., 2013. Performance improvement of long-range scanning Mie lidar for the retrieval of atmospheric extinction. J. Quant. Spectrosc. Radiat. Transf. 122, 72–78. https://doi.org/10.1016/j.jqsrt.2012.11.027

Holmgren, J., Nilsson, M., Olsson, H., 2003. Simulating the effects of lidar scanning angle for estimation of mean tree height and canopy closure. Can. J. Remote Sens. 29, 623–632. https://doi.org/10.5589/m03-030

Lee, D., Jung, H., Yom, J., Lim, S., Kim, J., 2006. Automatic Generation of Building Footprints From Airborne Lidar Data 44, 2523–2533.

Li, M., Rottensteiner, F., Heipke, C., 2019. Modelling of buildings from aerial LiDAR point clouds using TINs and label maps. ISPRS J. Photogramm. Remote Sens. 154, 127–138. https://doi.org/10.1016/j.isprsjprs.2019.06.003

Minato, A., Kobayashi, T., Sugimoto, N., 1998. Laser long-path absorption lidar technique for measuring methane using gas correlation method. Japanese J. Appl. Physics, Part 1 Regul. Pap. Short Notes Rev. Pap. 37, 3610–3613. https://doi.org/10.1143/JJAP.37.3610

Morsy, S., Shaker, A., El-Rabbany, A., 2017. Multispectral lidar data for land cover classification of urban areas. Sensors (Switzerland) 17. https://doi.org/10.3390/s17050958

Mosadegh, E., Nolin, A.W., 2022. A New Data Processing System for Generating Sea Ice Surface Roughness Products from the Multi-Angle Imaging SpectroRadiometer (MISR) Imagery. Remote Sens. 14. https://doi.org/10.3390/rs14194979

Nakahashi, K., Ito, Y., Togashi, F., 2003. Some challenges of realistic flow simulations by unstructured grid CFD. Int. J. Numer. Methods Fluids 43, 769–783. https://doi.org/10.1002/fld.559

Park, Y., Guldmann, J., 2019. Computers , Environment and Urban Systems Creating 3D city models with building footprints and LIDAR point cloud classification : A machine learning approach. Comput. Environ. Urban Syst. 75, 76–89. https://doi.org/10.1016/j.compenvurbsys.2019.01.004

Roriz, R., Cabral, J., Gomes, T., 2021. Automotive LiDAR Technology: A Survey. IEEE Trans. Intell. Transp. Syst. 1–16. https://doi.org/10.1109/TITS.2021.3086804

Rutzinger, M., 2009. Topographic Laser Ranging and Scanning—Principles and Processing, International Journal of Applied Earth Observation and Geoinformation. https://doi.org/10.1016/j.jag.2009.05.001

Shooshtarian, S., Lam, C.K.C., Kenawy, I., 2020. Outdoor thermal comfort assessment: A review on thermal comfort research in Australia. Build. Environ. 177, 106917. https://doi.org/10.1016/j.buildenv.2020.106917

Sola, A., Corchero, C., Salom, J., Sanmarti, M., 2020. Multi-domain urban-scale energy modelling tools: A review. Sustain. Cities Soc. 54, 101872. https://doi.org/10.1016/j.scs.2019.101872

Sowmya, A., Trinder, J., 2000. Modelling and representation issues in automated feature extraction from aerial and satellite images. ISPRS J. Photogramm. Remote Sens. 55, 34–47. https://doi.org/10.1016/S0924-2716(99)00040-4

Wang, W., Xu, Y., Ng, E., Raasch, S., 2018. Evaluation of satellite-derived building height extraction by CFD simulations: A case study of neighborhood-scale ventilation in Hong Kong. Landsc. Urban Plan. 170, 90–102. https://doi.org/10.1016/j.landurbplan.2017.11.008

Wu, Q., Zhong, R., Dong, P., Mo, Y., Jin, Y., 2021. Airborne lidar intensity correction based on a new method for incidence angle correction for improving land-cover classification. Remote Sens. 13, 1–22. https://doi.org/10.3390/rs13030511

Chapter 2

A deep learning algorithm building footprint polygon extraction for a GIS-based Solar power potential estimation

Abstract

Solar power is a renewable energy alternative that can be used to convert energy consuming buildings into energy producers. Solar power potential estimation for residential buildings focusses on assessing the incoming solar energy at the rooftops. In this study, a Light Detection and Ranging (LiDAR) data and residential building footprint polygons are used to identify the exposed rooftops. Novel footprint polygons extraction technique using a deep learning model trained on residential building rooftops has been developed and used for estimating the solar power generation potential. Compared to other traditional morphological filtering and machine learning algorithms, the deep learning using U-Net architecture has produced more accurate building footprint polygons. The LiDAR is used to generate a Digital Surface Model (DSM), which contains height information of the buildings, ground surface and trees in the vicinity area. A series of Geographical Information Systems (GIS) methodologies were employed to calculate solar power potential for the buildings. Solar energy availability is carried out for all seasonal changes including both during winter months when the trees shade their leaves, and in summertime when there is a considerable tree shading. This new methodology has been applied and illustrated for selected residential buildings in the city of London, Ontario.

Keywords: Deep Learning, Machine learning, LiDAR, U-Net, Satellite imagery, Autonomous, 3D models, Building, DSM, Solar power, rooftops, renewable energy, GIS

2.1 Solar power potential

Climate change and global warming are aggravated by the increasing world energy consumption from non renewable sources that produce significant CO_2 emissions. As a result, renewable energy developments are becoming the alternatives (Ordóñez et al., 2010). Studies have shown that conventional energy resources are limited and there exhaustive exploitation is causing a progressive deterioration of the environment (Jefferson, 2006, Jahangiri et al., 2021). Urban and residential buildings consume a bulk of a city's energy demand. Residential buildings represent around 25% of global energy consumption and 17% of global CO_2 emissions (Seddiki and Bennadji, 2019). Thus, renewable energy solutions for residential energy sector will have an impact on environmental pollution and global warming (Pablo-Romero et al., 2017). One of the ways to limit high energy consumption by buildings is to transform buildings into energy producers by using building-integrated solar energy technologies (Li et al., 2015)(Corcelli et al., 2019)(Chen et al., 2022). Particularly, the traditional functionalities of the building enclosure (roofs, walls) that are typically limited into control, support and distribute functions can be expanded into energy generation (Mora et al., 2011) by integrating for example solar thermal or photovoltaic (PV) systems to roofs of residential buildings. The first step in these sustainable efforts is to assess the availability of the solar energy at the building enclosure.

There have been various studies on estimating rooftop solar photovoltaic power. (Singh and Banerjee, 2015) used PVSyst simulations to estimate photovoltaic-available roof area. (Jakubiec and Reinhart, 2012) used LiDAR measurements and Geographic Information System (GIS) data for estimation of urban rooftop photovoltaic potential. (Izquierdo et al., 2008) carried out estimation of the technical potential of roof-integrated photovoltaic systems. The method was based on easily accessible data such as land use and population and building densities. (Bergamasco and Asinari, 2011) calculated the roof area suitable for solar applications through analysis of available GIS data and solar radiation maps. (Ordóñez et al., 2010) analyzed grid-connected solar photovoltaic capacity of residential rooftops using available data source showing gross roof surface area for each building type. (Wiginton et al., 2010) used geographic information systems and image recognition to determine the available rooftop area for PV deployment for a large-scale region in Ontario.

Solar energy systems, mostly in the form of solar thermal and photovoltaic systems, are currently widely being adopted out of the various energy systems that can be installed in the building sector in order to cover energy requirements (Tsalikis and Martinopoulos, 2015)(Huide et al., 2017)(Good et al., 2015). Most countries have started implementing the production of solar energy by means of building-integrated PV systems that are connected to the grid (Celik, 2006)(Bojić and Blagojević, 2006). (Castro et al., 2005) stressed that the roof area estimation is a fundamental input for the knowledge of the solar thermal potential in residential buildings. The solar power estimation involves various parameters like building type, orientation, roof tilt angle and shadow effect to name a few (Izquierdo et al., 2008). The quantification of the shadowing effects among buildings can be done with a digital 3D model of buildings for calculating suitable roof area (Robinson, 2006). Previous works on building integrated photovoltaic system installations assumed the available roof surface area as an input (Sørensen, 2001).

Urban and residential areas have a massive potential for energy efficiency improvement since these areas are the largest energy consumer groups (Sola et al., 2020)(Dwijendra et al., 2022). The complex urban environment, with varying building block densities and even more so building elevations, combined with limited available construction data about the existing building stock, are the main reasons for the difficulties emerging in the effort to assess solar potential (Karteris et al., 2013)(Jing et al., 2022). Geographic Information Systems (GIS) have proved to be a useful tool for regional renewable energy potential estimation and effective support for decision-making in energy planning at the urban scale (Groppi et al., 2018)(Voinontas, 1998)(Wong et al., 2016). Precise site and building-specific information are necessary to evaluate climate loads during environmental design (Zhai, 2006). The traditional way of manually classifying and entering building forms in CAD model are often a time-consuming process (Lach et al., 2006). Solar energy is critical for emerging and developing countries as well, as they may encounter the largest challenges in the energy transition ahead as their energy demand is growing fast and much of their energy production is still based on fossil fuels (Wegertseder et al., 2016). The use of solar energy in urban environments requires knowing the geographical distribution and characteristics of the best places to implement solar systems (Santos et al., 2014). Solar radiation from the sun is the primary energy source for the earth's physical and biological processes. Solar radiation is affected by atmosphere, topography and surface features. This thesis aims to contribute to automated residential neighborhood topology generation for solar energy potential estimation. Exposed residential building footprint polygons are used to estimate solar power potential. A deep learning model is trained on a collected training set of residential buildings for summer and winter conditions. Coupling with solar geometry, it is then used to estimate the solar power potential. Thus, this contributes to sustainable building design methods.

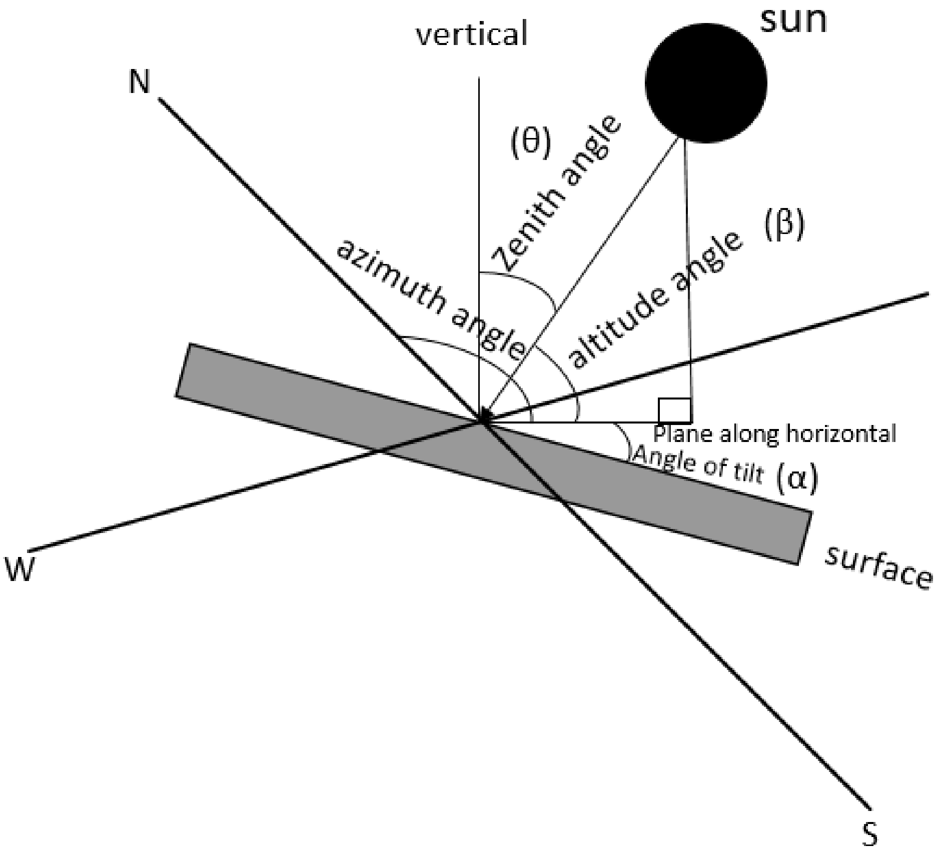


Figure 2.1 Solar geometry parameters

The total amount of radiation calculated for a specific area is given as global radiation (Maleki et al., 2017). The total solar radiation or global solar radiation is the sum of direct and diffuse solar radiation (Kodysh et al., 2013). The direct solar radiation from a sun is described with a centroid at zenith angle and azimuth angle as shown in Fig. 2.1 (Kodysh et al., 2013):

$$\mathbf{Direct\ radiation} = \sum \mathbf{Direct\ radiation}_{\theta,\boldsymbol{\alpha}} \qquad \text{Eq.2.1}$$

$$\boldsymbol{Direct\ rad}_{\theta,\boldsymbol{\alpha}} = \boldsymbol{S}_{\boldsymbol{const}} * \boldsymbol{\beta}^{\boldsymbol{m}(\theta)} * \boldsymbol{SunDur}_{\theta,\boldsymbol{\alpha}} * \boldsymbol{SunGap}_{\theta,\boldsymbol{\alpha}} * \mathbf{cos}\,(\boldsymbol{AngIn}_{\theta,\boldsymbol{\alpha}}) \qquad \text{Eq.2.2}$$

where S_{const} is the solar flux outside the atmosphere at the mean earth distance is known as the solar constant; β is the transmissivity of the atmosphere for the shortest path; *m(Θ)* is the relative optical path length, measured as a proportion relative to the zenith path length; $SunDur_{\Theta,\alpha}$ is the time duration represented by the sky sector; $SunGap_{\Theta,\alpha}$ is the gap fraction for the sun map sector; $AngIn_{\Theta,\alpha}$ is the angle of incidence between the centroid of the sky sector and the axis normal to the surface. Relative optical length, *m(Θ),* is determined by the solar zenith angle and elevation above sea level. For zenith angles less than 80˚, it can be calculated using the following equation:

$$\boldsymbol{m}(\Theta) = \boldsymbol{Exp}(-\mathbf{0.000118} * \boldsymbol{Elev} - \mathbf{1.638} * \mathbf{10^{-9}} * \boldsymbol{Elev^{2}})/\mathbf{cos}\,(\Theta) \quad \text{Eq.2.3}$$

where *Θ* the solar zenith angle and *Elev* is the elevation above sea level in meters. The effect of surface orientation is taken into account by multiplying by the cosine of the angle of incidence. The angle of incidence between the intercepting surface and a given sky sector with a centroid at the zenith angle and azimuth angle is calculated using the following equation:

$$\boldsymbol{AngIn}_{\Theta,\boldsymbol{\alpha}} = \mathbf{acos}\,(\mathbf{cos}(\Theta) * \mathbf{cos}(\boldsymbol{G_z}) + \mathbf{sin}(\Theta) * \mathbf{sin}(\boldsymbol{G_z}) * \mathbf{cos}(\boldsymbol{\alpha} - \boldsymbol{G_a})) \quad \text{Eq.2.4}$$

where G_z is the surface zenith angle and G_a is the surface azimuth angle. For each sky sector, the diffuse radiation at its centroid is calculated, integrated over the time interval, and corrected by the gap fraction and angle of incidence using the following equation (Kodysh et al., 2013):

$$\boldsymbol{Dif}_{\Theta,\boldsymbol{\alpha}} = \boldsymbol{R_{glb}} * \boldsymbol{P_{dif}} * \boldsymbol{Dur} * \boldsymbol{SkyGap}_{\Theta,\boldsymbol{\alpha}} * \boldsymbol{Weight}_{\Theta,\boldsymbol{\alpha}} * \mathbf{cos}\,(\boldsymbol{AngIn}_{\Theta,\boldsymbol{\alpha}}) \quad \text{Eq.2.5}$$

where R_{glb} is the global normal radiation; P_{dif} is the proportion of global normal radiation flux that is diffused; Dur is the time interval for analysis; $SkyGap_{\Theta,\alpha}$ is the gap fraction for the sky sector; $Weight_{\Theta,\alpha}$ is the proportion of diffuse radiation originating in a given sky sector relative to all sectors; $AngIn_{\Theta,\alpha}$ is the angle of incidence between the centroid of the sky sector and the intercepting surface. The global normal radiation can be calculated by summing the direct radiation from every sector without correction for the angle of incidence, then correcting for the proportion of direct radiation:

$$\mathbf{R_{glb}} = \frac{(\mathbf{S_{const}} \sum (\boldsymbol{\beta}^{\mathbf{m}(\Theta)}))}{(\mathbf{1-P_{dif}})} \qquad \text{Eq.2.6}$$

The uniform sky diffuse model is calculated as:

$$\mathbf{Weight}_{\Theta,\boldsymbol{\alpha}} = \frac{(\mathbf{cos}\Theta_{\mathbf{2}}-\mathbf{cos}\Theta_{\mathbf{1}})}{\mathbf{Div_{azi}}} \qquad \text{Eq.2.7}$$

where Θ_1 and Θ_2 are the bounding zenith angles of the sky sector; Div_{azi} is the number of azimuthal divisions in the sky map. The standard overcast sky model is calculated as:

$$\mathbf{Weight}_{\Theta,\boldsymbol{\alpha}} = \frac{(\mathbf{2cos}\Theta_{\mathbf{2}}+\mathbf{cos2}\Theta_{\mathbf{2}}-\mathbf{2cos}\Theta_{\mathbf{1}}-\mathbf{cos2}\Theta_{\mathbf{1}})}{\mathbf{4}*\mathbf{Div_{azi}}} \qquad \text{Eq.2.8}$$

The total diffuse solar radiation for the location is calculated as the sum of the diffuse solar radiation from all the sky map sectors:

$$\mathbf{Dif_{tot}} = \sum \mathbf{Dif}_{\Theta,\boldsymbol{\alpha}} \qquad \text{Eq.2.9}$$

The second data required for solar power estimation is LiDAR data. The area of interest is represented by point cloud data. The point cloud is converted to Digital Surface Model (DSM) so that it will be useful for solar energy estimation.

2.2. Automated building footprint polygon generation

Building footprint polygons are one of the input parameters to estimate solar power potential and simulate other micro-climate parameters in suburban and urban areas. For micro-climate studies the multifaceted relationship between the environment and the built structures needs to be accurately modelled, this will require accurate urban geometric modelling. The accessibility of free geospatial data like those obtained from satellites and 3D point cloud LiDAR are offering prospects to produce urban-scale city models in three-dimensional format at a low rate

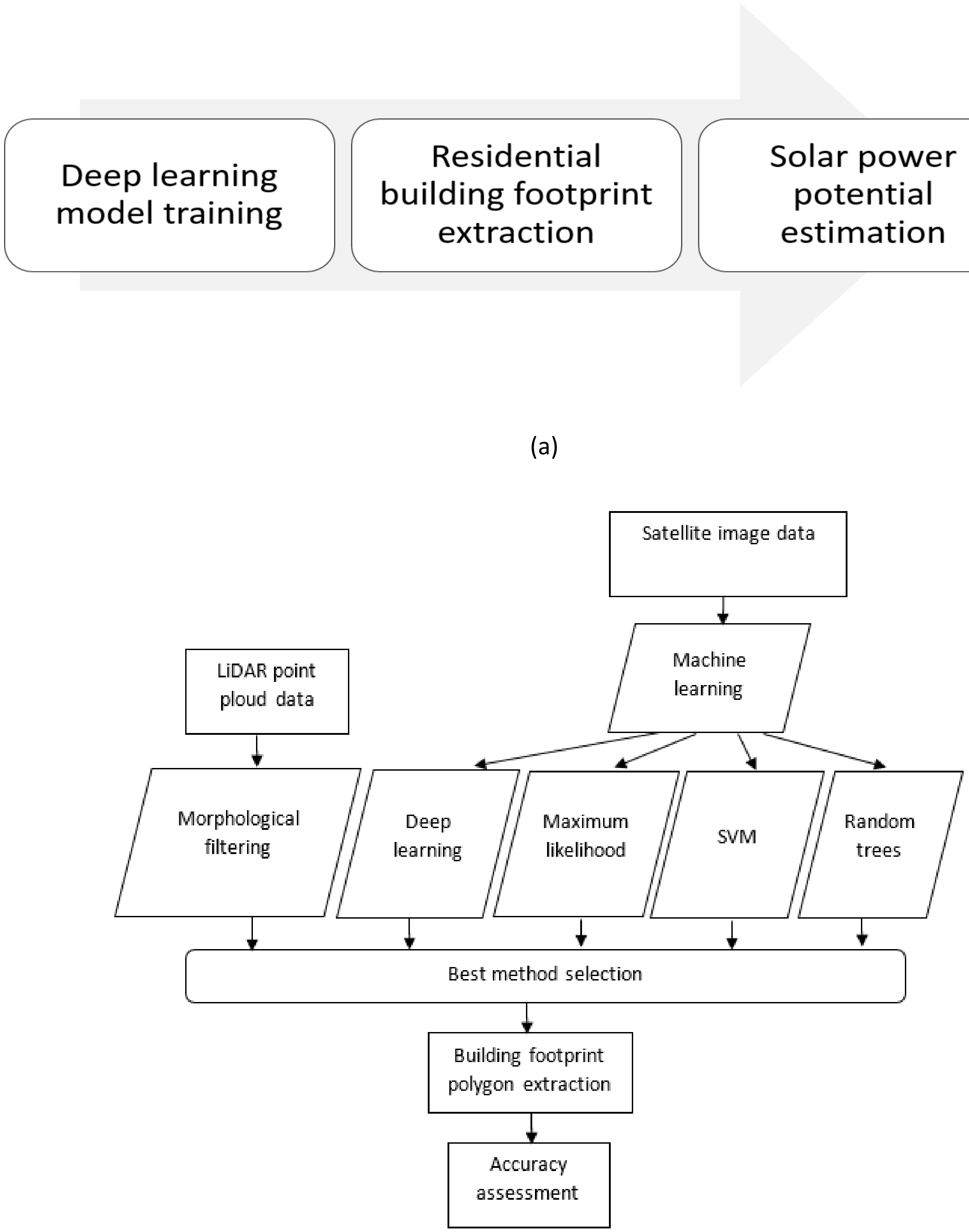
(Park and Guldmann, 2019) as modern deep learning methods have increasingly extended into photogrammetry, remote sensing, and machine learning (Park and Guldmann, 2019). Building footprint polygons can be extracted from an image representing a building as seen from the top (Vallet et al., 2011)(Gavankar and Ghosh, 2018)(Shackelford et al., 2004) which is the focus of this thesis. Different types of data and methods were explored by different researchers in an attempt to obtain building footprint polygons (Rutzinger et al., 2009)(Tomljenovic et al., 2015)(Milosavljević, 2020). In a specific study, a region growing method was tested to classify raw LiDAR data into coarse regions, a Triangulated Irregular Network (TIN) based roof primitive detection model was used to create the roof structure, and a contour vertex refinement algorithm was applied to regularize the edges of the roof (Li et al., 2019). A filtering algorithm in (Lach et al., 2006) was used to filter ground points from raw airborne point cloud measurements and generated an estimated digital terrain model (DTM). The algorithm utilized planar surface features and connectivity with locally lowest points to improve the extraction of ground points. A slope parameter used in the algorithm was updated after an initial estimation of the DTM, and thus local terrain information could be included. The algorithm extracted ground points from areas where different degrees of slope variation were interspersed. Specifically, along roads and streets, ground points were extracted from urban areas, from hilly areas such as forests, and from flat area such as riverbanks. However, most building polygon extraction models based on LiDAR failed to produce accurate roof shape representation and proved time consuming when applied to large scale projects due to their large file size and high computational demand. In addition, LiDAR data can be noisy, with points not always accurately reflecting the shape of the building. This can lead to errors in the extracted polygons, particularly when it comes to more complex roof shapes. LiDAR data are usually large file sizes, which can be time-consuming to process and analyze. This can be especially problematic when dealing with large-scale projects that require the processing of vast amounts of data. The high computational demand of LiDAR-based building polygon extraction models can also contribute to their time-consuming nature. These models require significant computing resources to process and analyze the data, which can slow down the process, particularly when working with large data sets.

Digital images and remote sensing image models that are applied to generate urban-scale 3D modelling are restricted by the time-consuming conventional processes (He et al., 2022). The manual conventional processes are arduous tasks that require attributing 3D and spatial data manually (Goldberg et al., 2018)(Das and Chand, 2021)(Li et al., 2021). The first initiative to bring autonomous workflow was presented by (Lach et al., 2006) to reduce the dependency on manual effort. Satellite imagery's resolution is critical for specific purposes that require higher-resolution data (Richner, 2011). Unlike high-resolution imagery which provides dense pixels, traditional photogrammetric methods do not offer this option (Chen et al., 2004).

Methods that employ point cloud data may suffer due to the elevation from which the point cloud data is recorded (Estornell et al., 2011). In such cases, some data may be missing in areas where a higher resolution is required (Holmgren et al., 2003). LiDAR data may also incur horizontal errors, where points may be recorded off their original location due to GPS and navigation unit operation anomalies (Park and Guldmann, 2019)(Liu, 2008)(Meng et al., 2010). Airborne LiDAR data is also more prone to suffer from missing data due to the absorption or reflection of laser energy (Minato et al., 1998)(Hodgson and Bresnahan, 2013)(Csanyi and Toth, 2007). LiDAR may contain noise that will affect the final output classification feature extraction phase (Gao et al., 2013)(Fang and Huang, 2004) and usually operate at a monochromatic wavelength measuring the range and the strength of the reflected energy (intensity) from objects limiting recording of a diversity of spectral reflectance (Morsy et al., 2017). After considering various methodologies, this thesis selected few techniques, and a test was carried out to choose the best method for building polygon extraction.

2.3 Methodology

The overall workflow in this chapter is summarized in Fig 2.2. First, a deep learning model was trained using images representing residential buildings. Then the trained model was used to predict residential building footprint polygons. Finally, the solar power potential for a selected neighborhood area was estimated using the extracted polygons as an input parameter. This thesis explored various ways to develop an autonomous site-specific building footprint polygon extraction. For this purpose, various machine learning techniques were tested to extract building footprint polygons.



(a)

(b)

Figure 2. 2 (a) General workflow (b) Building footprint polygon extraction methods evaluation

This section tested several algorithms before selecting the optimum method for building footprint polygon extraction (Fig. 2.2). Satellite images and LiDAR data were used to test morphological filtering and machine learning techniques. After evaluating the classification results, the best method for building footprint polygon extraction was selected. Finally, an accuracy assessment for the selected method is analysed and discussed.

The satellite images used are obtained from Google Earth. Google Earth combines satellite imagery, aerial photography, and other geographic data to provide a detailed, three-dimensional view of the Earth's surface. The satellite images used by Google Earth are captured by various satellites, including those operated by NASA, the European Space Agency (ESA), and other organizations. Google Earth primarily uses multiple satellite image data with varying resolutions merged together that are captured by various satellites in orbit around the Earth. These satellites capture high-resolution imagery using sensors that detect electromagnetic radiation in various wavelengths, such as visible light, infrared, and microwave. The LiDAR data is computationally demanding to process and analyze. Therefore, image data was used for the initial phase of building footprint polygon extraction process. Then those polygons were used to filter building points from the LiDAR data for the 3D modelling phase.

LiDAR stands for Light Detection and Ranging, and it is a remote sensing technology that uses laser light to measure distances and create three-dimensional representations of the Earth's surface. The basic concept of LiDAR involves emitting a laser beam from a sensor towards the ground or other objects, such as buildings or vegetation. The laser beam reflects off the object and returns to the sensor, where it is detected and measured. By measuring the time it takes for the laser beam to travel to the object and back, the distance between the sensor and the object can be calculated. LiDAR sensors emit thousands of laser pulses per second, creating a dense point cloud of 3D data that represents the surface of the Earth or other objects. The point cloud consists of *x,y,z* coordinates that represent the location of each point in three-dimensional space, along with intensity values that reflect the amount of laser light reflected by each point. The x and y coordinates of each point in the LiDAR point cloud are determined by the position of the LiDAR sensor and the direction and angle of the laser beam when it was emitted. The z coordinate of each point is determined by the time it takes for the laser beam to travel to the object and back, and the speed of light.

2.3.1 Morphological filter

In the case of LiDAR, since point clouds cover various surfaces (e.g. top of buildings, trees, vehicles, infrastructure, and terrain), the first step is to classify these points into distinct groups (Park and Guldmann, 2019).

The point cloud data used for the morphological filter in this thesis was obtained from Land Information Ontario (LIO) office. This LiDAR data is used throughout the thesis. The Ontario Point Cloud (Lidar-Derived) consists of points containing elevation and intensity information derived from returns collected by an airborne topographic lidar sensor. The point cloud is structured into non-overlapping 1 km by 1 km tiles in LAZ format. The following classification codes are applied to the data: unclassified, ground, water, high noise and low noise. This dataset is a compilation of lidar data from multiple acquisition projects, so specifications, parameters, accuracy and sensors may vary by project. This data is for geospatial tech specialists, and is used by government, municipalities, conservation authorities and the private sector for land use planning and environmental analysis. The LiDAR data of the London downtown area was extracted from the larger LiDAR data covering Ontario province.

The first methodology pursued, based on an extensive literature review, was the morphological filter method. A progressive morphological filter can be used to separate the ground and non-ground points from LiDAR data. The method uses a structuring element (which is a small pixel template), that is applied to a set of pixels to help produce a new image from an old one on rasterized LiDAR data. Using the structuring element, erosion and dilation operations are applied to separate parts of an image or to join them. The usage of these erosion and dilation operations will result in the opening which is erosion followed by dilation to break narrow bridges or eliminate thin structures, and closing which is dilation followed by erosion to fuse narrow breaks and eliminate small holes.

Dilation is defined as

$$dp = \max_{(xp,yp)\in w} (z_P) \qquad \text{Eq.2.10}$$

where points p is a LiDAR point in consideration, (x_p,y_p,z_p) represent p's neighbouring points within a window, w. The dilation output is the maximum elevation value in the neighbourhood of p. Erosion is a counterpart of dilation and is defined as

$$ep = \min_{(xp,yp)\in w} (z_P) \qquad \text{Eq.2.11}$$

Erosion operation with a smaller window size removes tree objects of sizes smaller than the window size, while dilation restores the shapes of large building objects.

The first step is to load the (x,y,z) LiDAR measurements. Then, a surface grid is constructed, and point coordinates are stored in each grid cell. If a cell contains no measurements, it is assigned the value of the nearest point measurements (Zhang et al., 2003).

In Fig. 2.3 (Zhang et al., 2003), the variable $dhp,1$ represents the height difference between an original LiDAR measurement and a filtered surface in an initial iteration at any given point p and $dhT,1$ represents the elevation difference threshold. Point p is classified as a ground measurement if $dhp,1 \leq dhT,1$ and as a nonground measurement if $dhp,1 \geq dhT,1$.

The threshold value of dhT,1 is typically determined empirically through trial and error. In other words, different values of dhT,1 are tested and evaluated to find the value that produces the most accurate classification of ground and non-ground points. One common approach is to first set a relatively large value for dhT,1 to capture all ground points, then iteratively decrease its value until the classification accuracy is optimized. Other factors that can influence the choice of dhT,1 include the terrain slope and vegetation cover, as well as the density and quality of the LiDAR data. In some cases, a multi-scale approach may be used where different values of dhT,1 are applied at different resolutions to account for variations in terrain complexity.

In general, the elevation difference threshold dhT,k is set to be the minimum height value of the building objects in an analyzed area at iteration k. Taking dhT,k as the threshold, for any given point p at k^{th} opening operation, p is marked as a ground measurement if $dhp,k \leq dhT,k$, and as a nonground measurement otherwise. In this way, the measurements for buildings with various sizes can be identified by gradually increasing the window sizes and applying an opening operation repeatedly until a window size is greater than the size of the largest building. Since there is also an abrupt elevation change from trees to the adjacent ground, the above building filtering procedure can be applied to the removal of tree measurements as well. This way, the point measurements can be classified as ground and non-ground measurements.

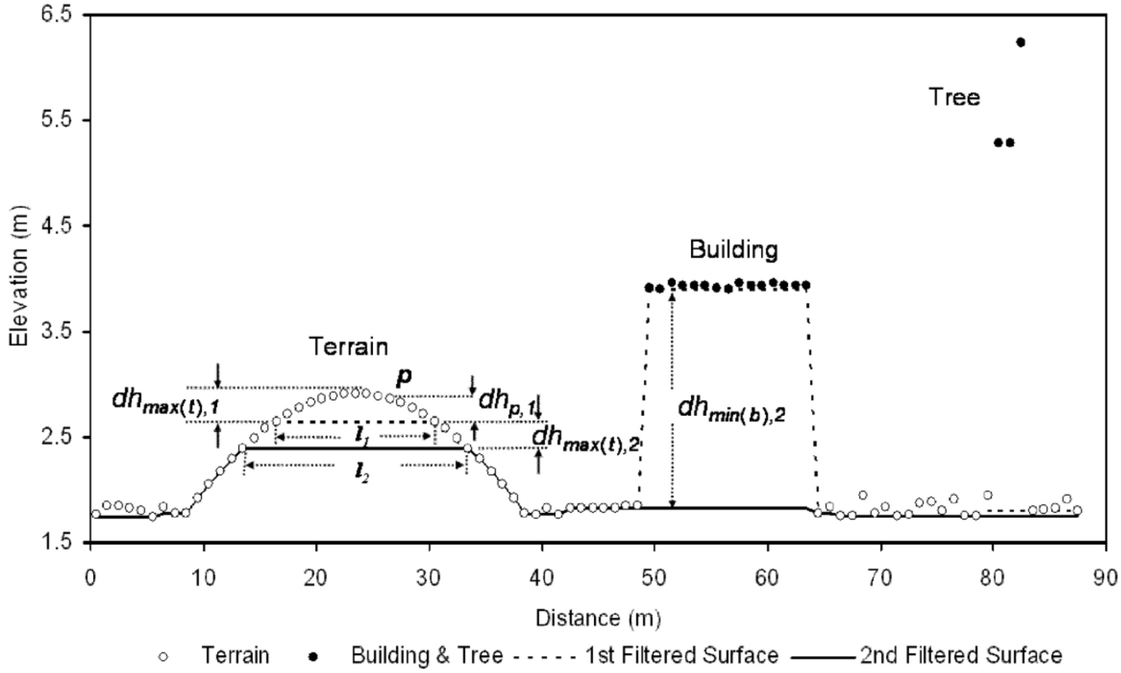


Figure 2. 3 Process of the progressive morphological filter to identify building measurements

2.3.2 Morphological filter results

The morphological filter concept was tested for building footprint polygons extraction. Initially, the input LiDAR data was converted into grid form after rasterization. Then the morphological principles were applied to extract building polygons (Fig. 2.4).

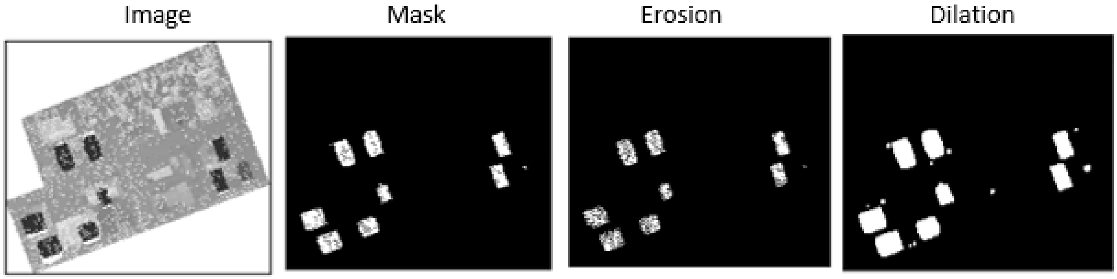


Figure 2. 4 Morphological filter results for extracting building footprints

```
Import cv2

import numpy as np

from matplotlib import pyplot as plt

img = cv2.imread('png.png', cv2.IMREAD_GRAYSCALE)

_, mask = cv2.threshold(img, 100, 200, cv2.THRESH_BINARY_INV)
```

The binary masks were applied using the code above. The code applies a binary threshold to the input image 'img' using a lower threshold value of 100 and an upper threshold value of 200. It then creates a binary mask using the thresholded image, where the foreground pixels are set to 0 and the background pixels are set to 1. The mask is inverted, so the foreground pixels are represented by 1 and the background pixels by 0, due to the use of the 'cv2.THRESH_BINARY_INV' flag.

This algorithm attempted to identify building shapes from a LiDAR-derived grayscale image based on a morphological filter. Once the building shapes are identified, the polygons are merged with the collected LiDAR data to classify building points in the point cloud data. Before that, the image must be georeferenced to match the projection coordinate system of the LiDAR data. The projection of the LiDAR data is Transverse Mercator. And North_American_1983_CSRS_UTM_Zone_17N projected coordinate system and NAD 1983 (CSRS) geographic coordinate system were used for the data.

The algorithm was able to identify high-rise building points (Fig. 2.5). However, the algorithm needs to be improved further to efficiently identify lower and medium-height buildings. The same method was applied to satellite images, but the output suffered due to shadow effects giving errors in classification (Fig. 2.6).

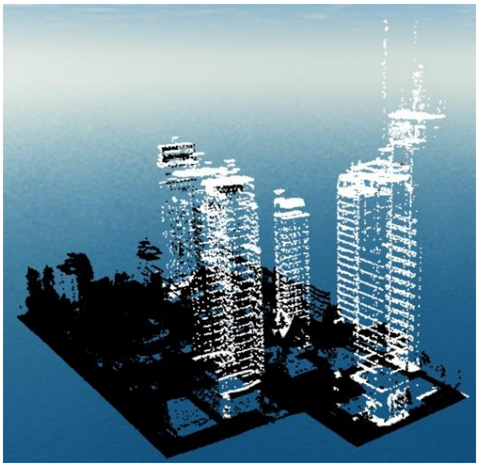


Figure 2. 5 3D view of the morphological filter result

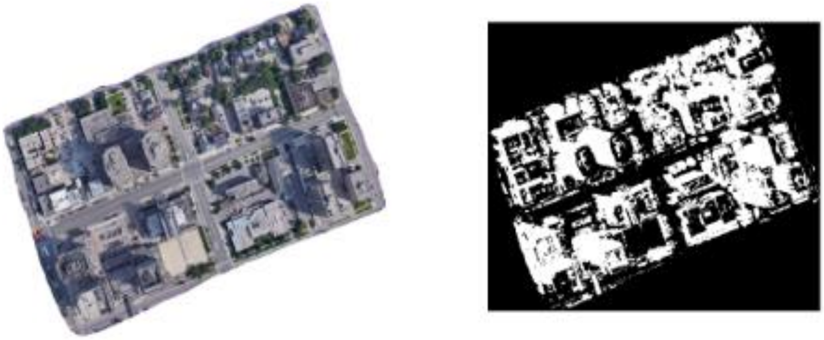


Figure 2. 6 Morphological filter applied to satellite imagery

The morphological filter showed limitations in identifying lower and medium height buildings. The method wrongly classified trees and other objects as buildings. The method didn’t improve for various iterations. Therefore, other methods like machine learning techniques and deep learning were explored to address these issues.

2.3.3 Machine learning approach

Artificial Intelligence (AI) refers to the development of computer systems that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation. AI is based on the idea that machines can learn from data, identify patterns, and make decisions with minimal human intervention. AI encompasses a wide range of techniques and approaches, including machine learning, deep learning, neural networks, natural language processing, and computer vision. These techniques enable machines to learn from data and improve their performance over time, without being explicitly programmed for each specific task.

Machine learning (ML) is a subset of artificial intelligence that involves the development of algorithms and models that can learn from data and make predictions or decisions without being explicitly programmed. It enables machines to automatically improve their performance on a given task as they are exposed to more data. The process of machine learning typically involves the following steps:

Data collection: Gathering and preparing a dataset that will be used to train the machine learning model.

Data preprocessing: Cleaning and transforming the data to ensure that it is ready for use by the machine learning algorithms.

Feature engineering: Selecting and extracting the relevant features or variables from the data that will be used to train the model.

Model training: Feeding the preprocessed data into the machine learning algorithm to train a model that can make predictions or decisions.

Model evaluation: Testing the trained model on a separate dataset to measure its performance and identify areas for improvement.

Model deployment: Integrating the trained model into a real-world system or application to automate a task or process.

There are different types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on labeled data, where the desired output is already known. Unsupervised learning involves training a model on unlabeled data, where the desired output is unknown. Reinforcement learning involves training a model to make decisions in an environment based on feedback in the form of rewards or punishments.

In line with the objective of this thesis, several machine learning algorithms were tested and compared for image classification purposes. After testing the morphological filter technique on the LiDAR dataset, the imagery data approach was tested to extract building footprint polygons. After analyzing the current trends in computer vision, the support vector machine, random trees and maximum likelihood were selected for testing. The machine learning tools were tested on imagery representing downtown area of London, Ontario (Fig. 2.7). The image in the figure is a satellite data used for a testing area for the image segmentation task. The satellite image is obtained from Google Earth and is captured by various satellites, including those operated by NASA, the European Space Agency (ESA), and other organizations. The use of Google Earth images is subject to the terms of service of Google, which state the images can be used for personal, non-commercial and research purposes. However, for commercial purposes, permission from Google or the owner of the imagery needs to be obtained. The training images were collected in the year 2022 over a course of a week.

Training data was prepared for each method based on a supervised classification approach to determine which classes exist in the image and assign each pixel or object to one of these classes.

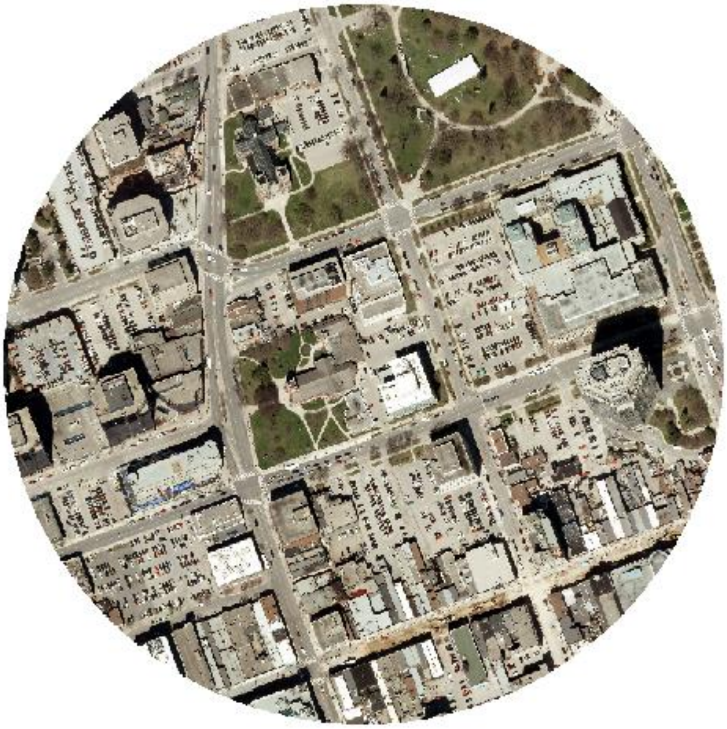


Figure 2. 7 The testing area in downtown London, Ontario

I. *Support Vector Machine classifier*

Support Vector Machines (SVM) can be used as a machine learning algorithm to separate an image into multiple regions or segments based on the visual characteristics of the image. To use SVM for image segmentation, the first step is to extract features from the image, such as texture, color, shape, and other relevant information. These features are then used to train an SVM model to recognize and classify different segments of the image. During the training process, the SVM algorithm tries to find the hyperplane that best separates the different segments of the image based on the extracted features. The hyperplane is designed to maximize the margin between the different segments, ensuring that the segments are well-separated and distinct from each other.

Once the SVM model is trained, it can be used to segment new images by predicting the class or segment of each pixel based on its features. The SVM algorithm assigns each pixel to the segment that it is most likely to belong to based on the visual characteristics of the image. SVM has been used in various image segmentation applications, including medical image analysis, object recognition, and computer vision. SVM is particularly useful in cases where the image has complex or non-linear features that cannot be easily segmented using traditional image processing techniques. SVM can help to improve the accuracy and efficiency of image segmentation tasks, especially when used in conjunction with other machine learning algorithms and techniques.

While Support Vector Machines (SVM) can be a powerful algorithm for image segmentation, there are some limitations to its use:

Computationally expensive: SVM is a computationally intensive algorithm, especially when dealing with large datasets or complex features. This can make it difficult to apply SVM to real-time image segmentation applications, where fast processing is required.

High-dimensional feature space: In order to use SVM for image segmentation, it is often necessary to transform the original image data into a high-dimensional feature space. This can result in a large number of features, which can be difficult to handle and may require additional preprocessing or feature selection.

Sensitivity to hyperparameters: SVM requires the selection of several hyperparameters, such as the kernel function, regularization parameter, and margin parameter. The performance of the SVM algorithm can be highly sensitive to the choice of these parameters, and selecting optimal hyperparameters can be a challenging task.

Limited ability to handle noise and outliers: SVM works best when the data is well-separated and the different segments of the image are clearly defined. However, in cases where there is a lot of noise or outliers in the data, SVM may struggle to correctly classify these regions.

Limited ability to handle multiple classes: SVM is typically used in binary classification problems, which may limit its ability to handle more complex segmentation tasks that involve multiple classes or labels.

Overall, while SVM can be a useful algorithm for image segmentation, its limitations need to be carefully considered and addressed in order to ensure optimal performance.

II. Random Trees

Random Trees (RT) is a machine learning algorithm that can be used for image segmentation. In this context, the algorithm works by classifying each pixel in an image into one of several predefined classes, based on the features extracted from the image. RT is an ensemble learning algorithm that combines the predictions of multiple decision trees, each of which is trained on a random subset of the training data. The algorithm works by recursively partitioning the feature space into smaller regions, using a set of if-then rules derived from the training data.

In image segmentation, RT can be used to partition the feature space into regions that correspond to different segments of the image. For example, the algorithm might use color, texture, and shape features to distinguish between different regions of an image. During training, the algorithm builds a forest of decision trees, each of which is trained on a different subset of the training data. The decision trees are constructed by recursively partitioning the feature space into smaller regions, using a random subset of the features at each step. The algorithm chooses the best split at each node based on the information gain, which measures how much the split improves the classification accuracy.

Once the forest of decision trees has been trained, it can be used to segment new images by predicting the class or segment of each pixel based on its features. The algorithm assigns each pixel to the segment that it is most likely to belong to based on the visual characteristics of the image.

While Random Trees (RT) can be a powerful algorithm for image segmentation, there are some limitations to its use:

Sensitivity to hyperparameters: RT requires the selection of several hyperparameters, such as the number of trees, the maximum depth of each tree, and the number of features used at each split. The performance of the RT algorithm can be highly sensitive to the choice of these parameters, and selecting optimal hyperparameters can be a challenging task.

Limited ability to handle complex features: RT works best when the features used to segment the image are relatively simple and easy to distinguish. In cases where the features are more complex or difficult to separate, RT may not be as accurate as other machine learning algorithms.

Limited ability to handle class imbalance: In cases where one class of pixels is much more common than the others, RT may have difficulty accurately segmenting the less common classes. This can be mitigated through techniques such as class weighting or oversampling of the minority classes.

Limited ability to handle long-range dependencies: RT is a local algorithm, meaning that it only considers a small region of the image at each decision node. This can make it difficult to accurately segment images where long-range dependencies between pixels are important.

Limited ability to handle temporal data: RT is a static algorithm and does not take into account the temporal dynamics of an image sequence.

Overall, while RT can be a useful algorithm for image segmentation, its limitations need to be carefully considered and addressed in order to ensure optimal performance.

III. Maximumlikelihood

Maximum likelihood (ML) is a statistical algorithm that can be used for image segmentation. The ML algorithm works by finding the optimal set of parameters that maximize the likelihood of observing the image given the underlying model of the data. In image segmentation, ML can be used to estimate the parameters of a statistical model that captures the properties of the image. For example, the algorithm might assume that each pixel in the image is generated from a Gaussian distribution with a mean and variance that depends on the segment it belongs to.

During training, the algorithm estimates the parameters of the model by finding the maximum likelihood estimate that maximizes the probability of observing the training data. This involves calculating the likelihood of observing the training data given the model parameters and adjusting the parameters to maximize this likelihood. Once the model parameters have been estimated, the algorithm can be used to segment new images by assigning each pixel to the segment that is most likely to generate that pixel. The algorithm computes the likelihood of each segment generating the pixel and assigns the pixel to the segment with the highest likelihood.

While Maximum Likelihood (ML) algorithm can be a powerful algorithm for image segmentation, there are some limitations to its use:

Sensitivity to model assumptions: ML requires assumptions to be made about the underlying statistical model that generates the image data. If the assumptions made by the model are incorrect, or if the model does not capture all relevant features of the image, then the segmentation results may be inaccurate.

Limited ability to handle complex features: ML works best when the features used to segment the image are relatively simple and easy to distinguish. In cases where the features are more complex or difficult to separate, ML may not be as accurate as other machine learning algorithms.

Limited ability to handle class imbalance: In cases where one class of pixels is much more common than the others, ML may have difficulty accurately segmenting the less common classes. This can be mitigated through techniques such as class weighting or oversampling of the minority classes.

Limited ability to handle long-range dependencies: ML is a local algorithm, meaning that it only considers a small region of the image at each decision node. This can make it difficult to accurately segment images where long-range dependencies between pixels are important.

Computationally expensive: ML can be computationally expensive, especially for high-dimensional feature spaces or large datasets, which may limit its practical use in some applications.

Limited ability to handle noise and outliers: ML assumes that the image data is generated from a known statistical model, which may not be the case for all types of images. Additionally, ML can be sensitive to outliers and may not perform as well in cases where the image contains significant amounts of noise or outliers.

Overall, while ML can be a useful algorithm for image segmentation, its limitations need to be carefully considered and addressed in order to ensure optimal performance.

2.3.4 Machine learning results

The results for the three methods are shown below (Fig. 2.8). The figure shows image segmentation results from the machine learning algorithms of SVM, maximum likelihood and random trees. The algorithms were trained on labelled images representing buildings, streets, vegetation etc. The machine learning methods didn’t achieve a satisfactory result. The random trees technique registered relatively better result than the other two. By contrast, the maximum likelihood method showed the lowest potential for classifying buildings.

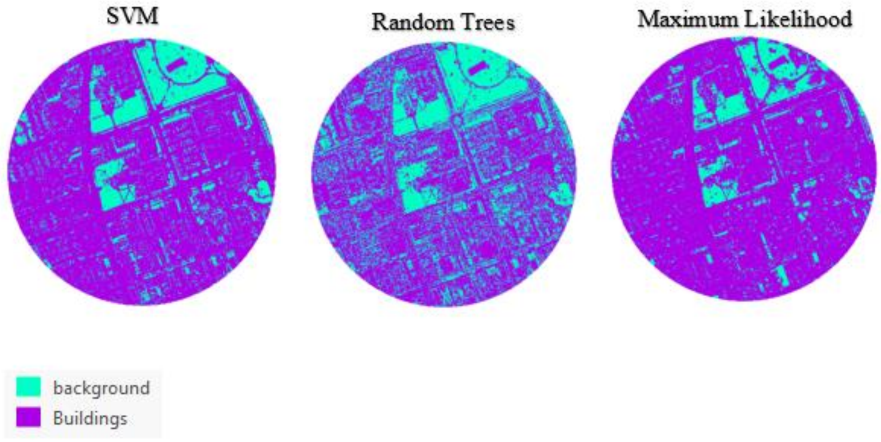


Figure 2. 8 SVM, random trees and Maximum likelihood segmentation results

2.3.5 Autonomous Extraction of Building Footprint polygons from high-resolution Satellite Image Data using U-Net based deep learning model

Convolutional Neural Networks (CNN) is a logistic type of algorithm where learning arises by adjusting the weight in the node using the recursive method and the error is backpropagated through the network to minimize the difference between output node activation and output (Misra et al., 2020). The obtained building footprint may contain noise details with irregular boundaries. The extracted building footprint polygons from CNN technique were simplified using the Douglas-Peucker algorithm (Lee et al., 2006).

The building footprint polygon extraction process started with collecting and processing training images (Fig. 2.9). The deep learning model took the input images and predicted building shapes after it was trained on image data. An accuracy assessment was done to check the network efficiency. Finally, the model extracted building footprint polygons from new satellite images.

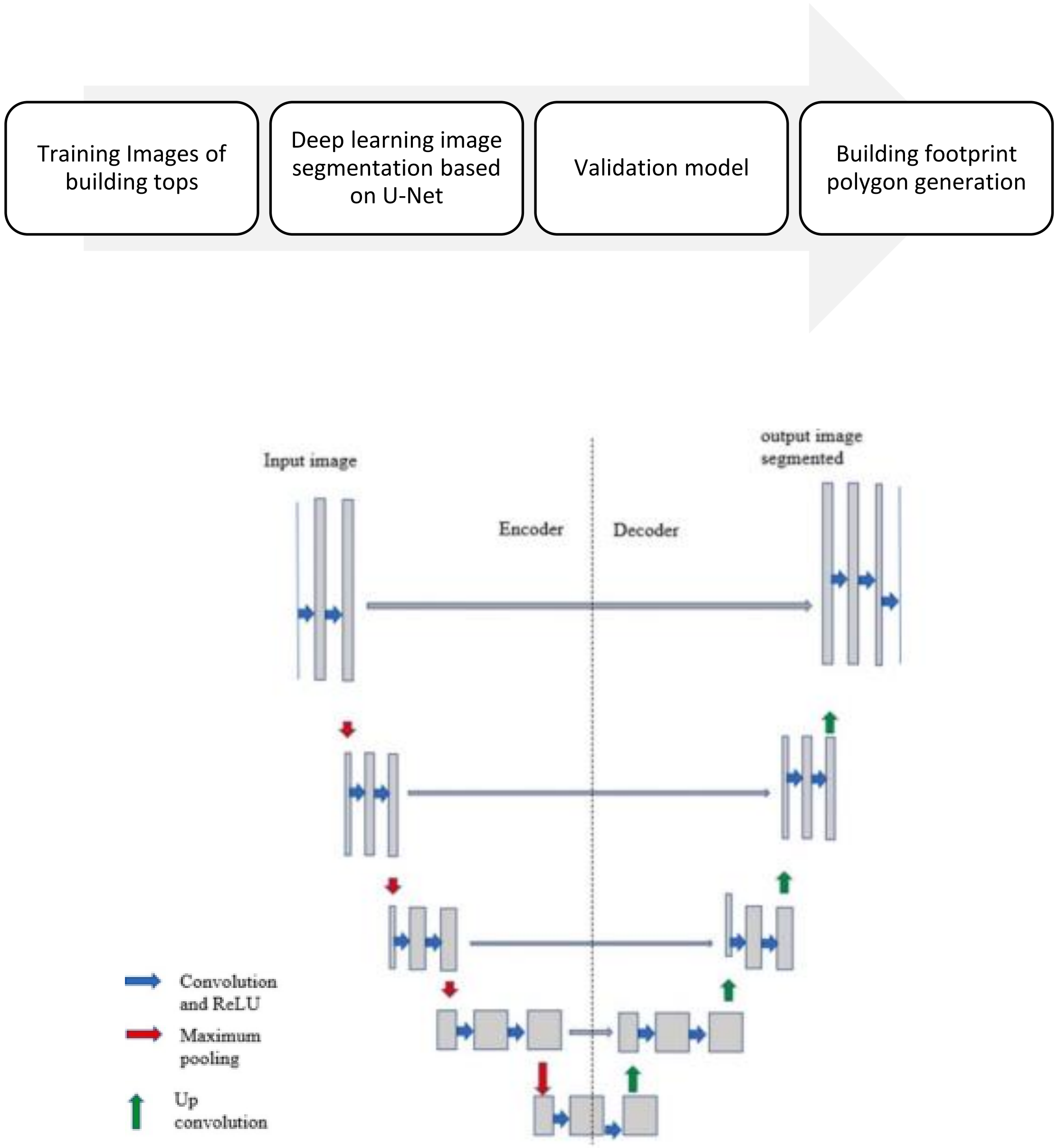

Training Images of building tops
Deep learning image segmentation based on U-Net
Validation model
Building footprint polygon generation
Input image
output image segmented
Encoder
Decoder
Convolution and ReLU
Maximum pooling
Up convolution

Figure 2. 9 (a) The building classification workflow, (b) Deep learning model architecture

The deep learning algorithm that was modelled based on U-Net (Weng and Zhu, 2021) like architecture registered an accuracy of 98%. In line with this thesis's purpose to address the building recognition problem in an urban and residential setting, the model achieved a higher accuracy level compared to other methods. The model was trained on building samples that were representative of the building construction style in the vicinity of London city. A thousand training images were collected which is a common minimum requirement for a deep learning model. As in any deep learning model, the training data should be relatable to the data that will be predicted post-training. The deep learning model was trained one thousand images. 10% of this data was used as a validation data during training the neural network to improve efficiency. A residential neighborhood area in London, Ontario was used as a testing ground to evaluate the accuracy of the deep learning model.

2.3.6 Deep learning model architecture

A deep learning technique was implemented to extract residential building footprint polygons from satellite imagery. Semantic segmentation associates each pixel of an image with a class label of either a residential building or background. The deep learning algorithm was tested in London, Ontario, Canada. The training and validation datasets used for the deep learning models were residential building satellite images collected from Kitchener in Ontario, Canada. A thousand training data were collected; out of which, 10% of the data was used for validation.

The deep learning algorithm was modelled based on U-Net (Weng and Zhu, 2021) like architecture. The model was trained on residential building samples that were representative of the residential building construction style in the vicinity of London city. A thousand training images were collected which is a common minimum requirement for a deep learning model.

As in any deep learning model, the training data should be relatable to the data that will be predicted post-training.

The first layer of the deep learning neural network is set to accept an image size of grayscale. Grayscale images are more convenient to process and make the training phase faster. The input image is specified as a row vector of integers. The image size [256 256 1] corresponds to [h w c], where h is the height of the image, w is the width of the image, and c is the number of channels. In this case, channel C 1 indicates it’s a grayscale image. For RGB images, the channel value will be 3, representing red, green and blue colours. If needed, the layer can be converted to accept RGB images. At this layer, the input image is processed so that the mean of the image lies at zero. This normalization helps to convert the image into a range of pixel values that are more familiar.

A convolutional layer in the neural network uses convolutional filters on an image. The filters move along the input image vertically and horizontally. This phase computes the dot product of weights and input and adds a bias term. There are 64 filters used in this stage with each of them of size 3 by 3 pixels. The stride describes the rate of movement for the filter. In this case, the first value 1 indicates a stride of one pixel in the vertical direction. The second value of 1 represents a horizontal movement of the filter.

The Rectified Linear Unit (ReLU) layer performs an operation on each element of input and sets any value less than zero to zero.

The maximum pooling layer down samples an incoming image by dividing the input into pooling regions and computes the maximum value of each region.

Image segmentation partitions an image into distinct parts based on the characteristics of the pixels in the image. Three types of error optimization algorithms were tested as part of the deep learning workflow to train a segmentation model.

a) Stochastic Gradient Descent with Momentum

The basic gradient descent algorithm updates the network parameters (weights and biases) to minimize the loss function by taking small steps at each iteration in the direction of the negative gradient of the loss as shown in Equation 2.12.

$$\boldsymbol{\theta_{\ell+1} = \theta_{\ell} - \alpha \nabla E(\theta_{\ell})} \qquad \text{Eq.2.12}$$

where ℓ is the iteration number, $\alpha > 0$ is the learning rate, θ is the parameter vector, and $E(\theta)$ is the loss function. In the standard gradient descent algorithm, the gradient of the loss function, $\nabla E(\theta)$, is evaluated using the entire training set, and the standard gradient descent algorithm uses the entire data set at once. By contrast, at each iteration, the stochastic gradient descent algorithm evaluates the gradient and updates the parameters using a subset of the training data. A different subset, called a mini-batch, is used at each iteration (Matlab, 2020). For this thesis, a mini-batch size of 4 was used. The size of the mini-batch size depends on the size of the training data and the computational power of a computer used.

The full pass of the training algorithm over the entire training set using mini batches is one epoch. Stochastic gradient descent is stochastic because the parameter updates computed using a mini batch are a noisy estimate of the parameter update that would result from using the full data set. The mini batch size and the maximum number of epochs were specified by using MiniBatchSize and MaxEpochs arguments respectively.

The stochastic gradient descent algorithm can oscillate along the path of steepest descent towards the optimum momentum term to the parameter update is one way to reduce this oscillation.

$$\boldsymbol{\theta_{\ell+1} = \theta_\ell - \alpha \nabla E(\theta_\ell) + \gamma(\theta_\ell - \theta_{\ell-1})} \qquad \text{Eq.2.13}$$

where γ determines the contribution of the previous gradient step to the current iteration.

b) RMSProp

Stochastic gradient descent with momentum uses a single learning rate for all the parameters. Other optimization algorithms seek to improve network training by using learning rates that differ by parameter and can automatically adapt to the loss function being optimized. RMSProp (root mean square propagation) is one such algorithm. It keeps a moving average of the element-wise squares of the parameter gradients

$$\boldsymbol{v_\ell = \beta_2 v_{\ell-1} + (1 - \beta_2)[\nabla E(\theta_\ell)]^2} \qquad \text{Eq.2.14}$$

β_2 is the decay rate of the moving average. The RMSProp algorithm uses this moving average to normalize the updates of each parameter individually

$$\boldsymbol{\theta_{\ell+1} = \theta_\ell - \frac{\alpha \nabla E(\theta_\ell)}{\sqrt{v_\ell} + \epsilon}} \qquad \text{Eq.2.15}$$

where the division is performed element-wise. Using *RMSProp* effectively decreases the learning rates of parameters with large gradients and increases the learning rates of parameters with small gradients.

c) Adam

Adam (derived from adaptive moment estimation) (Kingma and Ba, 2015) uses a parameter update that is similar to *RMSProp,* but with an added momentum term. It keeps an element-wise moving average of both the parameter gradients and their squared values,

$$\boldsymbol{m_\ell = \beta_1 m_{\ell-1} + (1 - \beta_1)\nabla E(\theta_\ell)} \qquad \text{Eq.2.16}$$

$$\boldsymbol{v_\ell = \beta_2 v_{\ell-1} + (1-\beta_1)[\nabla E(\theta_\ell)]^2} \qquad \text{Eq.2.17}$$

Adam uses the moving averages to update the network parameters as

$$\boldsymbol{\theta_{\ell+1} = \theta_\ell - \frac{\alpha m_l}{\sqrt{v_l}+\epsilon}} \qquad \text{Eq.2.18}$$

If gradients over many iterations are similar, then using a moving average of the gradient enables the parameter updates to pick up momentum in a certain direction. If the gradients contain mostly noise, then the moving average of the gradient becomes smaller, and so the parameter updates become smaller too. The full Adam update also includes a mechanism to correct a bias that appears at the beginning of training.

2.3.7 Training data preparation and analysis

A deep learning neural network process starts with collecting the required data for training a segmentation model. The data was collected from publicly available sources. Then the data was organized in a way for an image to represent a single residential building so that it would be convenient for labeling it as a training set. Once the image data are collected and organized the next task is to label each individual image. There are various image labelling applications out there both open-source and commercial ones. The VGG Image Annotator (VIA) open-source application was used to label the images. Then the training images were loaded into the VIA application and labelled using polygons to mark the boundary of the residential buildings in the image. The area enclosed by the yellow polygon will be registered as a residential building (Fig. 2.10). This way all the thousand images were labelled. After labelling, the labelled images were exported in JavaScript Object Notation (JSON) format. This helps to store data structures and objects in a standard data interchange format.

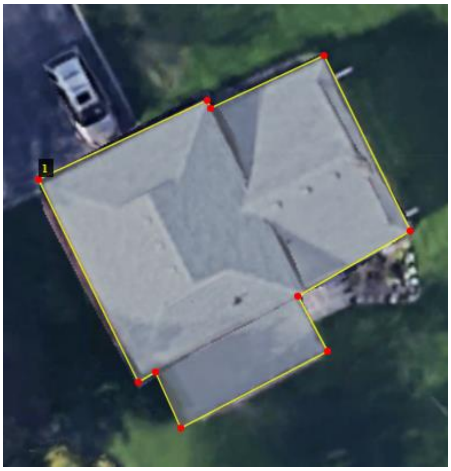


Figure 2. 10 Training image labelling

In order for the deep learning model to train using the training data, these JSON files need to be converted into image masks. The JSON files were converted to image masks using a python script.

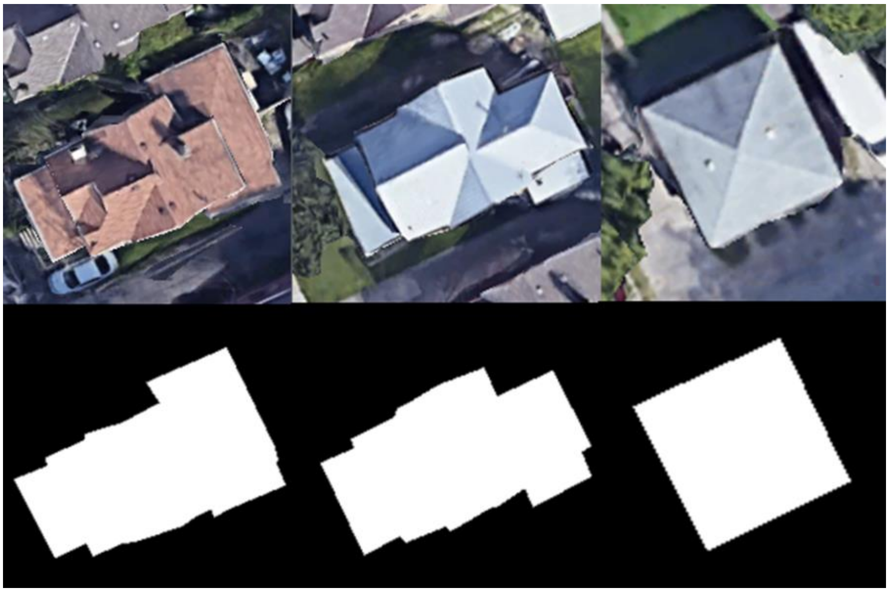


Figure 2. 11 Training data masks

At this stage, the collected original images are labelled and exported in JSON format and then corresponding masks are created (Fig. 2.11). Next, further pre-processing steps were applied to the images to make them more suitable for the neural network to absorb them. As a common rule in machine learning applications, images should be converted into grayscale images so as to simplify the dense data that comes with RGB images (Kanan and Cottrell, 2012). This makes the files smaller, demanding less computational power and the model doesn't have to waste time learning irrelevant information. Colour may introduce unnecessary information which increases the amount of training data required to achieve good performance. Then the images are processed to a size of 256-pixel length by 256-pixel width which is usually the rule of thumb in deep learning. MATLAB was used as a host for the algorithm code that runs the model. Deep learning models are known to achieve better accuracy when the training size increases. To address this, a data augmentation operation was applied. The original data was replicated four times to increase data size and then random scaling, horizontal reflection and rotation processes of augmentation were applied to further upsurge training data. Data augmentation is also critical to create images with new features that the model can learn from. After the data was split between training and validation, the model was trained on the augmented data.

Data augmentation is a technique used in deep learning to artificially increase the size of a training dataset by creating new examples through various transformations of the existing data. It involves applying a set of operations to an image or data sample, such as rotating, flipping, cropping, or changing the brightness, to create variations of the original sample.

The need for data augmentation in deep learning image segmentation context is because the success of deep learning models for image segmentation is highly dependent on the amount and quality of the training data available. Collecting and labeling large amounts of data for segmentation can be time-consuming and expensive. Additionally, the real-world variations and challenges encountered in the test data may not be fully represented in the training dataset, leading to overfitting or poor generalization performance.

Data augmentation helps address these challenges by generating new training data from the existing data, increasing the diversity and quantity of the dataset. This allows the model to learn more robust features and better generalize to new, unseen data. In the context of image segmentation, data augmentation techniques such as flipping, rotating, and resizing the images can help create additional training examples with different orientations and scales, which can improve the model's ability to segment objects accurately under different viewing conditions.

The image data were partitioned into training and validation data. The validation data is required to prevent the model from overfitting. The validation data is used during training in parallel with the training data. When writing the training code, two values were assigned to represent the residential buildings and background. The residential buildings were assigned a value of 255 and a 0 value was assigned for the background. Based on these values, the model assigned 255 values for pixels predicted as residential buildings. And the model assigned a 0 value for pixels predicted as background.

An image datastore was created to simplify data management. Image datastore enabled the deep learning model to import data in batches from image collections that are too large to fit in memory. A datastore makes it convenient to call images into training workflow without the need to store the data in a workstation which would otherwise be computationally demanding.

This way the code is allowed to read and analyze data from image folders in smaller portions that fit the available memory. The images are stored as PNG files in the datastore. For each training and validation image, a corresponding image mask is stored in a separate datastore ready to be called anytime the code runs. Basically, the datastore for the image mask is known as a pixel-label datastore since it contains images segmented at a pixel level.

The image datastore and its corresponding label datastore were combined into a single datastore so that it would be convenient for data augmentation. Then the labelled pixels are overlayed on the original image. This way the pixels representing the residential buildings will align exactly with the residential building boundaries.

Deep learning models register better performance when trained on larger datasets. Thus, data augmentation is required to increase the amount of training data. Data augmentation also applies different transformations to the training data which helps the model learn new features during training. For instance, augmentation applies randomized rotations to input images so that the model would be familiar with the presence of rotation in input images (Fig. 2.12).

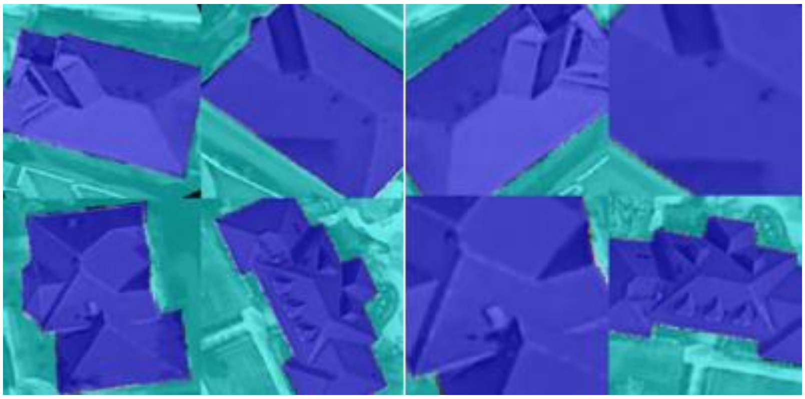


Figure 2. 12 Data augmentation sample results

2.3.8 Results for the U-Net-based model

The training progress was monitored by using validation data set as a control mechanism. The validation data is a set of data separate from training data. It is used during training process to check for overfitting and to test the network on new data that it has not seen. This process will improve the accuracy of the model. The training data is used during training of network to update the layer weights via backpropagation. The training data is fed to the network every iteration, the loss is calculated, and the layer weights are updated via backpropagation to reduce the loss for the iteration. The model can achieve higher accuracy through continuous training and saving the learned parameters and resuming training until there’s no change in the loss value.

This model showed improved results compared with other methods. The model was able to identify building outlines with considerable accuracy. The buildings labelled as ground truth are the prepared masks to evaluate the accuracy of the model’s prediction. The predicted building raster image layer is post processed to obtain final result. The layer is georeferenced and aligned with LiDAR data to extract the 3D point clouds representing a building. Regularization is also applied to refine the edges of the building outline (Fig 2.13). The figure shows how the predicted building polygons are matched to LiDAR data in a case where the polygons are extracted from images that are not georeferenced. The polygons are assigned the coordinate of the LiDAR and the building perimeter polygons are kept for the 3D modelling process. The final building footprint polygons extracted at the end of the process are shown in Fig. 2.14.

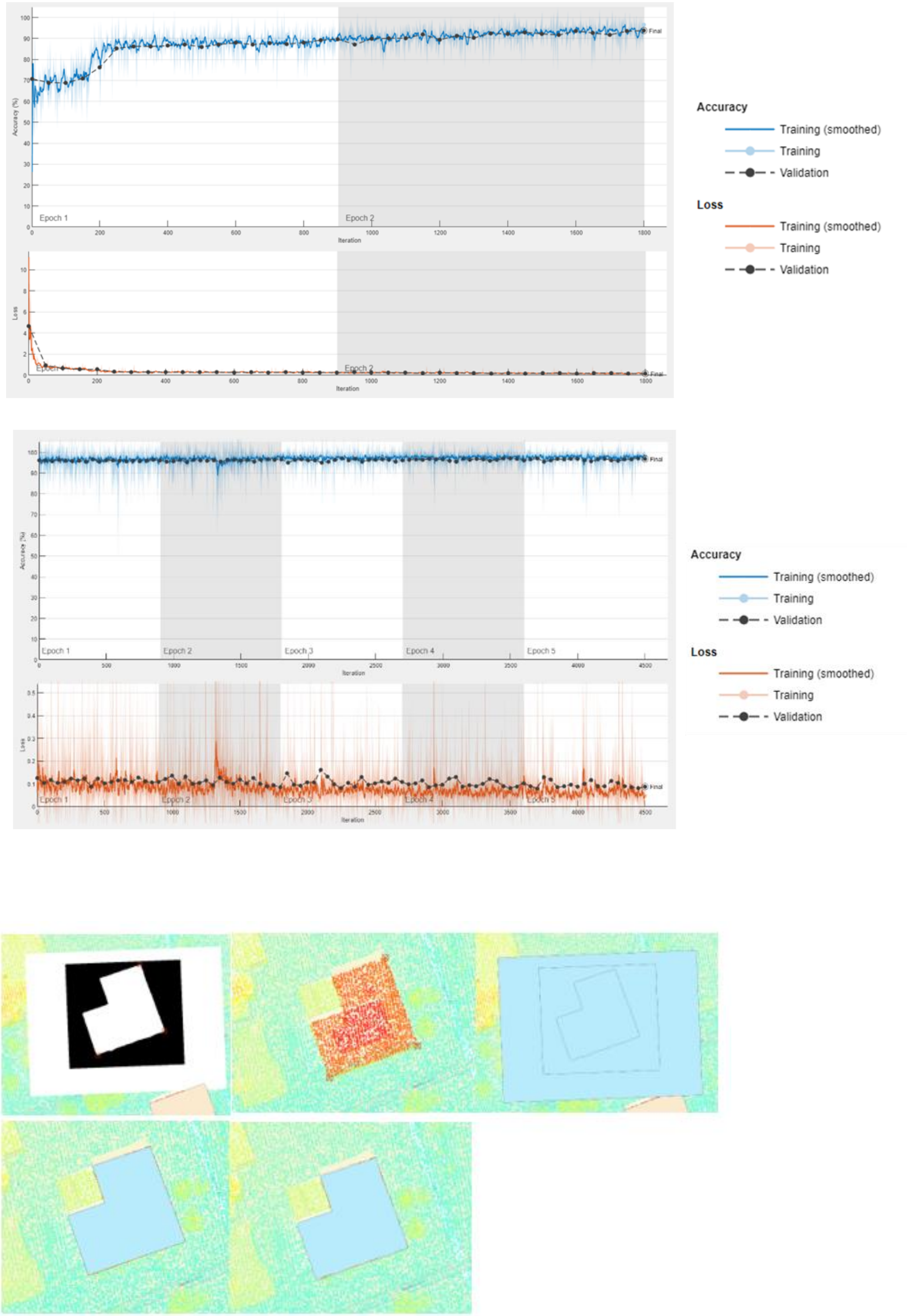


Figure 2. 13 (a) Neural network training (b) Final accuracy (c) Polygons extraction

The deep learning model is trained to identify only building tops using the label on each image marking the building surface area. The image masks generated from the training images help the model to clearly identify the building perimeter. Other details like streets, trees, cars etc are considered as background information by the neural network and are removed.

In order to achieve an efficient model, the network has to be trained extensively. The training progress was monitored by using validation data set as a control mechanism. The model can achieve higher accuracy through continuous training and saving the learned parameters and resuming training until there's no change in the loss value. This model showed improved results when data augmentation and validation data were applied.

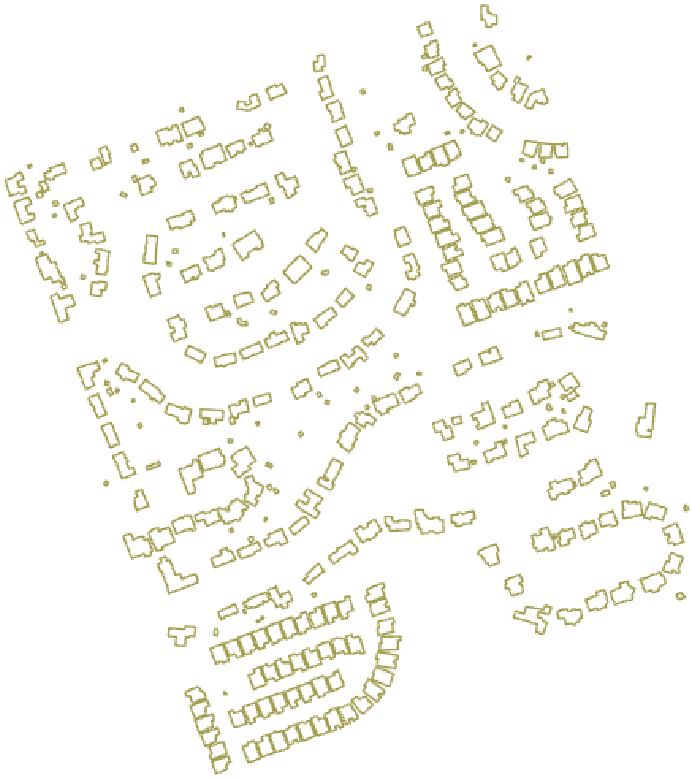


Figure 2. 14 Residential building footprints extracted by deep learning model

2.3.9 Validation model

The percentage of correctly identified pixels of each class is indicated by accuracy. This is important to know how well the residential building class pixels are identified. For the residential building class, the accuracy is the ratio of correctly classified pixels to the total number of actual pixels based on ground truth.

$$Accuracy\ score = TP/(TP+FN) \qquad \text{Eq.2.19}$$

The segmentation model was tested on selected residential buildings in London, Ontario.

$$metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTruth); \qquad \text{Eq.2.20}$$

The above equation computes various metrics to evaluate the quality of the semantic segmentation results which are predicted pixel labels (*pxdsResults*) against the ground truth segmentation which are ground truth pixel labels (*pxdsTruth*).

$$pxdsResults = semanticseg(imds,net,\text{"WriteLocation"},\text{'data\\single\_output'}) \qquad \text{Eq.2.21}$$

$$pxdsTruth = pixelLabelDatastore(testLabelsDir,classNames,labelIDs); \qquad \text{Eq.2.22}$$

Table 2. 1 Accuracy of the segmentation results

GlobalAccuracy	MeanAccuracy	MeanIoU	WeightedIoU	MeanBFScore
91%	95%	62%	88%	41%

Global accuracy is the ratio of correctly classified pixels, all classes, to the total number of pixels. The mean accuracy is the average accuracy of all classes in an image. The intersection over union (IoU) is the ratio of correctly classified pixels to the total number of ground truth and predicted pixels in the building class. The mean BF score indicates how well the predicted boundary of each building class aligns with the true boundary of the building.

$$\text{IoU score} = TP/(TP+FP+FN) \qquad \text{Eq.2.23}$$

2.4. Solar power potential analysis for residential building

The workflow developed for solar power potential analysis for residential buildings is illustrated in Fig. 2.15. A solar radiation raster is generated from a DSM layer and residential footprint polygons that were generated from the previous step using the deep learning model. Then based on the rooftop aspect, slope and amount of solar radiation reaching its surface, suitable surfaces are selected. Rooftops with suitable area of 30 square meters or greater are generally selected for solar panel installations. The building's suitable area and its average solar radiation per square meter gives solar power potential estimates.

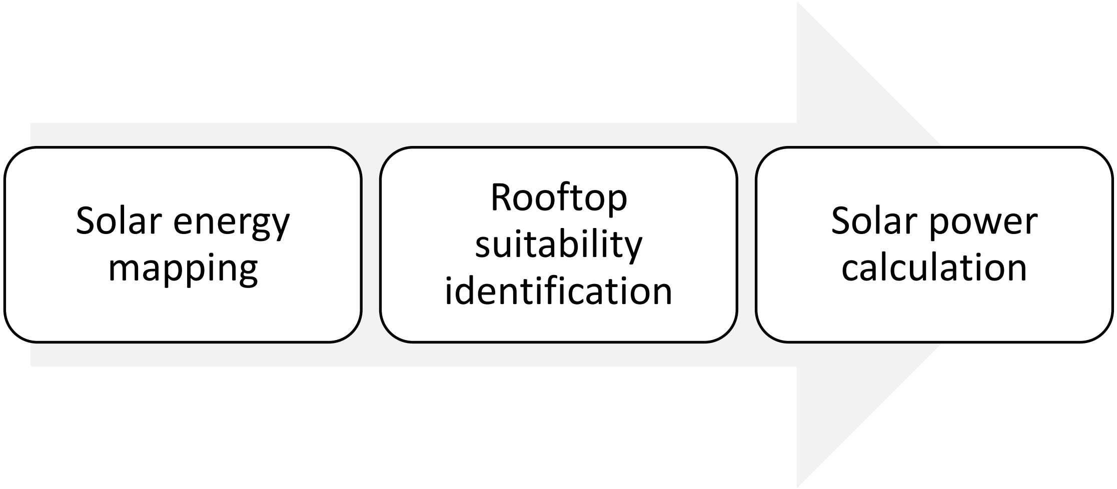


Figure 2. 15 Workflow of the rooftop solar power estimation

LiDAR data is extracted from a larger dataset that represented London, Ontario as a collection of 3D points (Fig. 2.16a). The LiDAR is converted to Digital Surface Model (DSM) that represents the area of interest. The DSM represents the elevation of the ground and features on the surface, such as trees and buildings. The DSM is a raster layer that shows data in a grid where each cell contains a numeric value. It is symbolized so that darker gray cells have lower elevations and the lighter gray and white cells have higher elevations (Fig. 2.16b). Each of the cells in the raster represent a surface of 0.5 by 0.5 meters resolution. For better visualization of the buildings and vegetation the DSM raster represents, a 3D visualization process is applied (Fig. 2.16c). The solar radiation is calculated using ArcGIS tool and taking the DSM and residential buildings footprint polygons as input. The tool calculated radiation by considering the position of the sun throughout the year and at different times of day, obstacles that may block sunlight such as nearby trees or buildings, and the slope and orientation of the surface (Fig. 2.16d). The DSM provides the required information on obstacles, orientation and slope. The output is a raster layer where each cell value is the amount of solar radiation in watt-hours per square meter at that location.

Once a solar radiation raster layer is created, the next step is to identify suitable rooftops. There are three criteria to consider when identifying suitable rooftops. First, suitable rooftops should have a slope of 45 degrees or less, as steep slopes tend to receive less sunlight. Second, suitable rooftops should receive at least 800 kWh/m^2 of solar radiation. Third, suitable rooftops should not face north, as north facing rooftops in the northern hemisphere receive less sunlight. A slope raster layer is generated from the DSM raster (Fig. 2.17a). The cells in the layer contain a slope value ranging from 0 to 90 degrees. The lighter colors represent steeper slopes while darker colors represent milder slopes. An aspect raster layer is also generated from the DSM raster (Fig. 2.17b). The cells in the layer contain a value expressing orientation in degrees, with 0 representing absolute north and 180 representing absolute south. If a cell has a slope steeper than 45 degrees, its value will be changed to NoData in the output layer (Fig. 2.17c). In the northern hemisphere, rooftops facing north are likely to receive less solar radiation than surfaces facing other directions. The aspect raster layer is used to remove slopes that face north having a value less than 22.5 degrees or more than 337.5 degrees (Fig. 2.17d). Since slopes of 10 degrees or less are more or less flat, such surfaces are kept regardless of their aspect. Rooftop surfaces receiving less than 800 kWh/m^2 solar radiation are also removed (Fig. 2.17d). After all the criteria are met, the resulting rooftop surface suitable for solar panel installation is mapped (Fig. 2.18a). Finally, the solar radiation each suitable raster cell receives is aggregated to determine how much solar radiation each building receives in a year. Then, the solar radiation is converted to electric power production potential (Fig. 2.18b).

A Digital Surface Model (DSM) is a digital representation of the Earth's surface or any other topographic surface, such as buildings, trees, and other above-ground objects. It is a 3D model that represents the elevation of the Earth's surface or objects on it, using a grid of elevation points. DSMs are typically created from remote sensing data, such as satellite or aerial imagery, or in this case LiDAR data, which uses lasers to measure the distance between the sensor and the Earth's surface.

A Digital Terrain Model (DTM) is a digital representation of the bare ground surface, which removes the impact of above-ground features such as trees, buildings, and other objects. It is a 3D model that represents the elevation of the Earth's terrain using a grid of elevation points. A DTM is created by filtering the elevation data from sources such as satellite or aerial imagery or LiDAR data, to remove the above-ground objects' elevation. The resulting model represents the Earth's terrain, free from any other features.

A Normalized Digital Surface Model (nDSM) is a digital model that represents the elevation of above-ground features such as buildings, trees, and other objects, relative to the Earth's bare ground surface. It is a 3D model that shows the height of the objects above the ground surface, which can be used to estimate the object's size, shape, and location accurately. The nDSM is created by subtracting the bare earth elevation model (such as a DTM) from the digital surface model (DSM), which includes the above-ground features. The resulting model represents only the above-ground features' height, which is normalized to the bare earth surface.

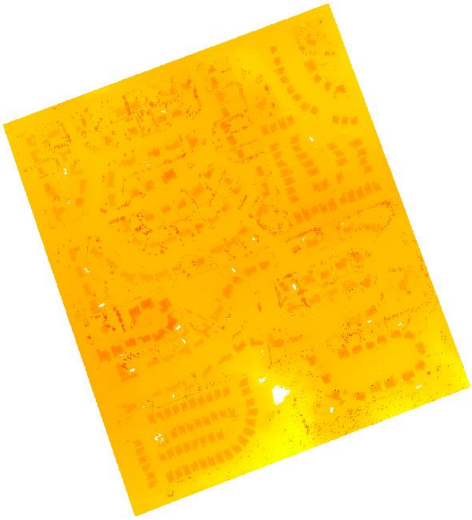
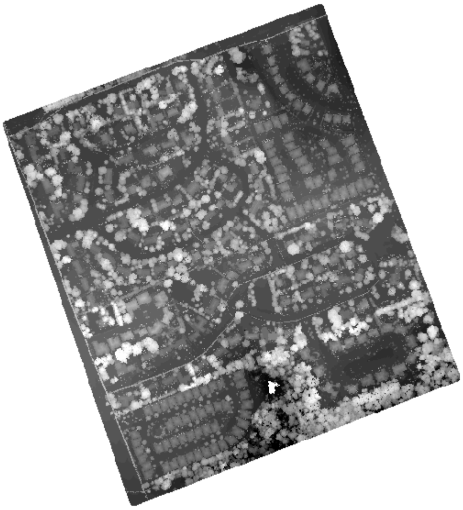
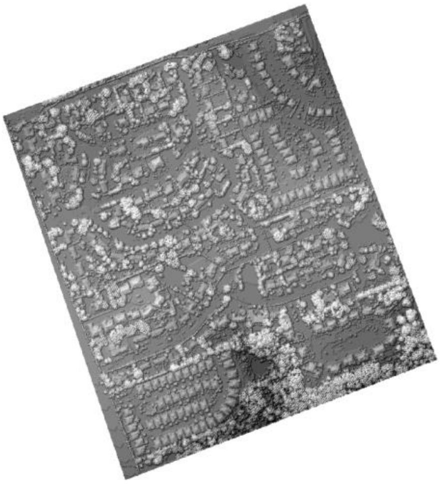
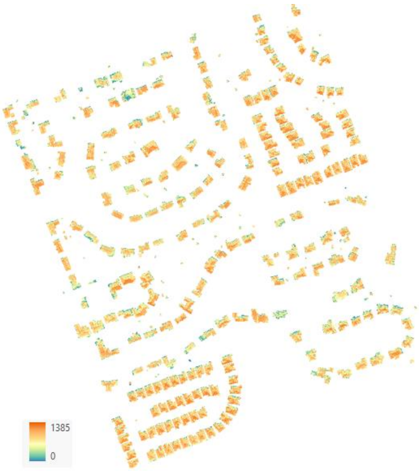
In Chapter 2, only the DSM layer was used for the solar power potential estimation. The DSM, DTM and nDSM layers together were used in Chapter 3 for the 3D building modelling process.

The DSM represents elevation information of topography and structures above ground. It is useful to obtain building height that is used to analyze solar potential. Based on the DSM, solar radiation is calculated for each roof surface. The solar radiation is in kilowatt-hours per square meter(kWh/m^2). An average Canadian household consumes 6.9 MWh of electricity per year.

When estimating solar radiation received using a DSM layer, the shadow effect from trees and adjacent buildings is taken into consideration by incorporating a shadowing model. The shadowing model is used to simulate the effect of objects casting shadows on the ground surface, which alters the amount of solar radiation that reaches the surface. A sky view factor (SVF) analysis measures the percentage of visible sky from a point on the ground surface. The SVF is calculated by analyzing the geometry of the surrounding objects, such as buildings and trees, and the sky dome's orientation. The resulting model can be used to estimate the amount of solar radiation that reaches the ground surface, taking into account the shadowing effect.

Roofs that are steeper than 45 degrees are typically removed from solar power potential estimation because they are not suitable for traditional solar panel installations. Solar panels are typically installed on roofs that have a pitch (angle) between 15 and 40 degrees. Roofs that are steeper than 45 degrees may not provide a stable surface for mounting solar panels, and the panels may not receive optimal sunlight exposure. In addition, the installation of solar panels on steep roofs can be challenging, as it may require specialized equipment and safety precautions. Furthermore, the energy output from solar panels decreases as the angle of the panel increases relative to the sun's position. For roofs that are steeper than 45 degrees, the angle may be too steep for the solar panels to receive optimal sunlight, resulting in a reduced energy output. Therefore, when estimating solar power potential, roofs that are steeper than 45 degrees are usually removed from consideration, as they are not ideal for solar panel installations and may not provide significant energy output.

Roofs that face the north direction in the northern hemisphere are typically removed from solar power potential estimation because they receive limited sunlight throughout the year. In the northern hemisphere, the sun is primarily located in the southern part of the sky. Therefore, roofs that face south or southwest receive the most sunlight throughout the day, making them the most suitable for solar panel installations. Roofs that face north or northeast receive little to no direct sunlight, which significantly reduces the energy output from solar panels. Furthermore, even if solar panels were installed on north-facing roofs, they would not receive optimal sunlight exposure, resulting in a reduced energy output. Therefore, when estimating solar power potential, roofs that face north or northeast in the northern hemisphere are typically removed from consideration, as they are not ideal for solar panel installations and may not provide significant energy output.

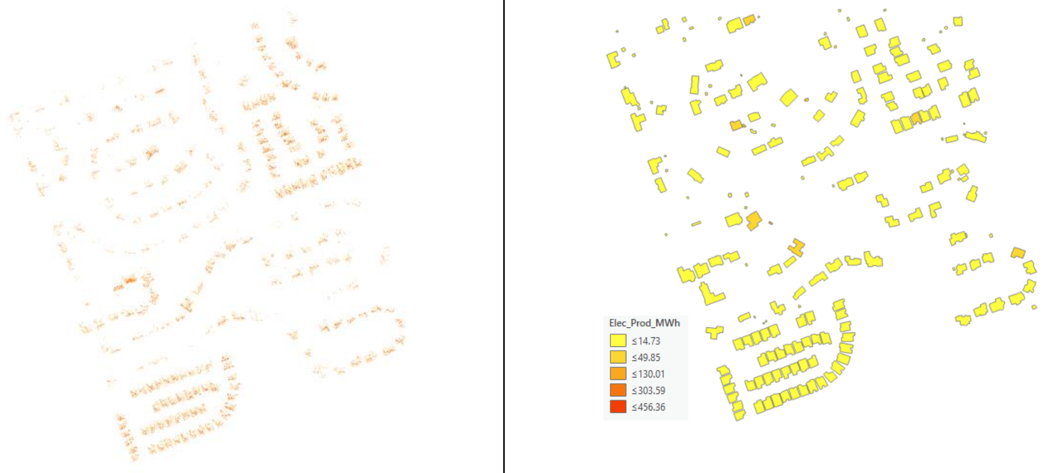
	
(a) LiDAR data analysis of the area where the residential buildings are located	(b) DSM generation of the location where the residential buildings are located
	

<table>
<tr><td>(c) DSM with 3D visualization of the lower height buildings</td><td>(d) Estimation of solar radiation that reaches the rooftops of the residential houses</td></tr>
<tr><td colspan="2"><b>Figure 2. 16 Solar energy mapping</b></td></tr>
<tr><td> </td><td> </td></tr>
<tr><td>(a) Slope estimation of the roofs between 0 and 90 degrees</td><td>(b) Aspect estimation of the roofs showing whether the roofs face north or south</td></tr>
<tr><td> </td><td> </td></tr>
</table>

(c) Rooftop cells that are steeper than 45 degrees are removed

(d) Rooftop cells steeper than 45 degrees and receiving less than 800kWh/m2 are removed

Figure 2. 17 Rooftop suitability identification



(a) The final roof surfaces that are suitable for solar panel installation

(b) Electric production in MWh of each building suitable for solar panel installation

Figure 2. 18 Solar power calculation

2.5 Conclusion

In this thesis, autonomous building footprint polygon extraction method was successfully developed. The deep learning technique achieved an improved result over traditional machine learning algorithms such as random trees, SVM object-based classification, maximum likelihood. The advantage of the deep learning model is that it can be applied to a new set of input image data to extract building footprint polygons for autonomous applications once it's trained. In addition, the model can be improved over time with minimum adjustments when quality data is available. Although it is recommended to deep learning and provides a higher accuracy result over machine learning techniques, it still requires upgrading and to continuously learn with new data to predict unique building forms that are unusual. In addition, when the number of buildings being projected at a single moment surge, the model struggles to capture specifics. This is mainly due to the limitation of the training set and computational requirements and hence it will perform better with more training data.

By combining the extracted residential building footprint polygons with LiDAR generated DSM, it was possible to automate the calculation of household solar panel estimation. Constraints such as removing rooftops with a slope of more than 45˚, north-facing buildings were implemented. The solar power potential estimation can be scaled to include entire neighborhoods as long as high-density LiDAR data is available. But most importantly, the application of machine learning to extract the residential house's polygon area used for the solar panel calculation paved the way for a more robust and automated workflow.

2.6 Reference

Bergamasco, L., Asinari, P., 2011. Scalable methodology for the photovoltaic solar energy potential assessment based on available roof surface area: Application to Piedmont Region (Italy). Sol. Energy 85, 1041–1055. https://doi.org/10.1016/j.solener.2011.02.022

Bojić, M., Blagojević, M., 2006. Photovoltaic electricity production of a grid-connected urban house in Serbia. Energy Policy 34, 2941–2948. https://doi.org/10.1016/j.enpol.2005.04.024

Castro, M., Delgado, A., Argul, F.J., Colmenar, A., Yeves, F., Peire, J., 2005. Grid-connected PV buildings: Analysis of future scenarios with an example of Southern Spain. Sol. Energy 79, 86–95. https://doi.org/10.1016/j.solener.2004.09.022

Celik, A.N., 2006. Present status of photovoltaic energy in Turkey and life cycle techno-economic analysis of a grid-connected photovoltaic-house. Renew. Sustain. Energy Rev. 10, 370–387. https://doi.org/10.1016/j.rser.2004.09.007

Chen, Q., Kuang, Z., Liu, X., Zhang, T., 2022. Transforming a solar-rich county to an electricity producer: Solutions to the mismatch between demand and generation. J. Clean. Prod. 336, 130418. https://doi.org/10.1016/j.jclepro.2022.130418

Corcelli, F., Fiorentino, G., Petit-Boix, A., Rieradevall, J., Gabarrell, X., 2019. Transforming rooftops into productive urban spaces in the Mediterranean. An LCA comparison of agri-urban production and photovoltaic energy generation. Resour. Conserv. Recycl. 144, 321–336. https://doi.org/10.1016/j.resconrec.2019.01.040

D. VOIVONTAS, D.A. and A.M., 1998. and COROMINAS. Renew. Energy Vol. 13, N.

Dwijendra, N.K.A., Rahardja, U., Kumar, N.B., Patra, I., Zahra, M.M.A., Finogenova, Y., Guerrero, J.W.G., Izzat, S.E., Alawsi, T., 2022. An Analysis of Urban Block Initiatives Influencing Energy Consumption and Solar Energy Absorption. Sustainability 14, 14273. https://doi.org/10.3390/su142114273

Good, C., Andresen, I., Hestnes, A.G., 2015. Solar energy for net zero energy buildings - A comparison between solar thermal, PV and photovoltaic-thermal (PV/T) systems. Sol. Energy 122, 986–996. https://doi.org/10.1016/j.solener.2015.10.013

Groppi, D., de Santoli, L., Cumo, F., Astiaso Garcia, D., 2018. A GIS-based model to assess buildings energy consumption and usable solar energy potential in urban areas. Sustain. Cities Soc. 40, 546–558. https://doi.org/10.1016/j.scs.2018.05.005

Huide, F., Xuxin, Z., Lei, M., Tao, Z., Qixing, W., Hongyuan, S., 2017. A comparative study on three types of solar utilization technologies for buildings: Photovoltaic, solar thermal and hybrid photovoltaic/thermal systems. Energy Convers. Manag. 140, 1–13. https://doi.org/10.1016/j.enconman.2017.02.059

Izquierdo, S., Rodrigues, M., Fueyo, N., 2008. A method for estimating the geographical distribution of the available roof surface area for large-scale photovoltaic energy-potential evaluations. Sol. Energy 82, 929–939. https://doi.org/10.1016/j.solener.2008.03.007

Jahangiri, M., Akinlabi, E.T., Sichilalu, S.M., 2021. Assessment and Modelling of Household-Scale Solar Water Heater Application in Zambia: Technical, Environmental, and Energy Analysis. Int. J. Photoenergy 2021. https://doi.org/10.1155/2021/6630338

Jakubiec, J.A., Reinhart, C.F., 2012. Towards validated urban photovoltaic potential and solar radiation maps based on lidar measurements , gis data , and hourly daysim simulations. SimBuild 2012. Fifth Natl. Conf. IBPSA-USA Madison, Wisconsin August 1-3, 2012 10.

Jefferson, M., 2006. Sustainable energy development: Performance and prospects. Renew. Energy 31, 571–582. https://doi.org/10.1016/j.renene.2005.09.002

Jing, R., Liu, J., Zhang, H., Zhong, F., Liu, Y., Lin, J., 2022. Unlock the hidden potential of urban rooftop agrivoltaics energy-food-nexus. Energy 256, 124626. https://doi.org/10.1016/j.energy.2022.124626

Kanan, C., Cottrell, G.W., 2012. Color-to-grayscale: Does the method matter in image recognition? PLoS One 7. https://doi.org/10.1371/journal.pone.0029740

Karteris, M., Slini, T., Papadopoulos, A.M., 2013. Urban solar energy potential in Greece: A statistical calculation model of suitable built roof areas for photovoltaics. Energy Build. 62, 459–468. https://doi.org/10.1016/j.enbuild.2013.03.033

Kingma, D.P., Ba, J.L., 2015. Adam: A method for stochastic optimization. 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc. 1–15.

Kodysh, J.B., Omitaomu, O.A., Bhaduri, B.L., Neish, B.S., 2013. Methodology for estimating solar potential on multiple building rooftops for photovoltaic systems. Sustain. Cities Soc. 8, 31–41. https://doi.org/10.1016/j.scs.2013.01.002

Lach, S.R., Brown, S.D., Kerekes, J.P., 2006. Semi-automated DIRSIG scene modelling from 3D LIDAR and passive imaging sources. Laser Radar Technol. Appl. XI 6214, 62140I. https://doi.org/10.1117/12.666096

Li, D., Liu, G., Liao, S., 2015. Solar potential in urban residential buildings. Sol. Energy 111, 225–235. https://doi.org/10.1016/j.solener.2014.10.045

Maleki, S.A.M., Hizam, H., Gomes, C., 2017. Estimation of hourly, daily and monthly global solar radiation on inclined surfaces: Models re-visited. Energies 10. https://doi.org/10.3390/en10010134

Matlab, 2020. Try This Example 1–15.

Mora, R., Bitsuamlak, G., Horvat, M., 2011. Integrated life-cycle design of building enclosures. Build. Environ. 46, 1469–1479.

https://doi.org/10.1016/j.buildenv.2011.01.018

Ordóñez, J., Jadraque, E., Alegre, J., Martínez, G., 2010. Analysis of the photovoltaic solar energy capacity of residential rooftops in Andalusia (Spain). Renew. Sustain. Energy Rev. 14, 2122–2130. https://doi.org/10.1016/j.rser.2010.01.001

Pablo-Romero, M. del P., Pozo-Barajas, R., Yñiguez, R., 2017. Global changes in residential energy consumption. Energy Policy 101, 342–352. https://doi.org/10.1016/j.enpol.2016.10.032

Robinson, D., 2006. Urban morphology and indicators of radiation availability. Sol. Energy 80, 1643–1648. https://doi.org/10.1016/j.solener.2006.01.007

Santos, T., Gomes, N., Freire, S., Brito, M.C., Santos, L., Tenedório, J.A., 2014. Applications of solar mapping in the urban environment. Appl. Geogr. 51, 48–57. https://doi.org/10.1016/j.apgeog.2014.03.008

Seddiki, M., Bennadji, A., 2019. Multi-criteria evaluation of renewable energy alternatives for electricity generation in a residential building. Renew. Sustain. Energy Rev. 110, 101–117. https://doi.org/10.1016/j.rser.2019.04.046

Singh, R., Banerjee, R., 2015. Estimation of rooftop solar photovoltaic potential of a city. Sol. Energy 115, 589–602. https://doi.org/10.1016/j.solener.2015.03.016

Sola, A., Corchero, C., Salom, J., Sanmarti, M., 2020. Multi-domain urban-scale energy modelling tools: A review. Sustain. Cities Soc. 54, 101872. https://doi.org/10.1016/j.scs.2019.101872

Sørensen, B., 2001. GIS management of solar resource data. Sol. Energy Mater. Sol. Cells

67, 503–509. https://doi.org/10.1016/S0927-0248(00)00319-6

Tsalikis, G., Martinopoulos, G., 2015. Solar energy systems potential for nearly net zero energy residential buildings. Sol. Energy 115, 743–756. https://doi.org/10.1016/j.solener.2015.03.037

Wegertseder, P., Lund, P., Mikkola, J., García Alvarado, R., 2016. Combining solar resource mapping and energy system integration methods for realistic valuation of urban solar energy potential. Sol. Energy 135, 325–336. https://doi.org/10.1016/j.solener.2016.05.061

Weng, W., Zhu, X., 2021. INet: Convolutional Networks for Biomedical Image Segmentation. IEEE Access 9, 16591–16603. https://doi.org/10.1109/ACCESS.2021.3053408

Wiginton, L.K., Nguyen, H.T., Pearce, J.M., 2010. Quantifying rooftop solar photovoltaic potential for regional renewable energy policy. Comput. Environ. Urban Syst. 34, 345–357. https://doi.org/10.1016/j.compenvurbsys.2010.01.001

Wong, M.S., Zhu, R., Liu, Z., Lu, L., Peng, J., Tang, Z., Lo, C.H., Chan, W.K., 2016. Estimation of Hong Kong’s solar energy potential using GIS and remote sensing technologies. Renew. Energy 99, 325–335. https://doi.org/10.1016/j.renene.2016.07.003

Zhai, Z., 2006. Application of computational fluid dynamics in building design: Aspects and trends. Indoor Built Environ. 15, 305–313. https://doi.org/10.1177/1420326X06067336

Chapter 3

Autonomous urban topology generation for urban flow modelling

Abstract

One of the challenges in realistic numerical urban micro-climate modelling for wind, heat transfer, and building energy simulation applications is the complexity of urban topology and complex building geometries. My original contribution in this thesis presents a deep learning modelling for building footprint polygon extraction from satellite imagery that is integrated with Light Detection and Ranging (LiDAR) data to generate 3D building models. The deep learning model registered an overall accuracy of 98%. The trained deep learning model can then be applied to a new set of input image data to extract building footprint polygons for autonomous application, and it can also be incrementally retrained with good quality data when it becomes available. A framework is developed that integrates the autonomous urban topology generator with urban flow modelling. The modelling steps are explained through an application example of urban flow modelling encountered during a pedestrian-level wind assessment for the city of London, Ontario.

Keywords:

Deep Learning, Satellite Image, LiDAR, Building footprints, Urban topology, CFD, Pedestrian level wind

3.1 Introduction

With approximately 70% of the world’s population expected to live in urban areas by 2050, and cities being one of the largest energy consumers and emitters of greenhouse gases, urban areas offer a large potential for energy efficiency improvement (Sola et al., 2020). While embracing sustainability, maintaining the resiliency of the built environment against climate stressors is also critical. Recent changes in urban ecosystems have had a negative impact on the liveability of outdoor built environments (Cureau et al., 2022). The collective effects of these changes in urban outdoor spaces challenge effective urban planning which aims to create successful and usable outdoor spaces. Among the determinants of outdoor environment quality, a high priority is given to wind and thermal environments (Shooshtarian et al., 2020). Therefore, a clear understanding and realistic modelling of the complex interaction between the climate and the built environment, characterized by complex transport mechanisms, is essential. This necessitates urban climate modelling (wind speed, pressure, humidity, temperature, etc.) at high temporal and spatial resolution.

One of the major challenges in realistic and pragmatic numerical urban micro-climate modelling for wind engineering, environmental, and building energy simulation applications is the complexity of the geometry (topology) of the computational domain and the variability of surface types involved in urban exposures (Liu et al., 2018, Tominaga et al., 2008, Liu et al., 2017). Accurate site and building-specific information are required to assess climate loads such as wind, for example, during environmental design (Zhai, 2006). Traditionally, building forms and surface classification are individually and manually entered into a CAD model through information collected from designers, on-site-observations and publicly available information, which are often time-consuming and cause delays as the area covered by a project in- crease (Lach et al., 2006). Hence, the generation of the geometry of complex urban topology and the associated grid used for numerical modelling represents one of the most engineering time intensive and costly processes in the computational process. The challenges to generating geometric and physics boundary conditions in an automated manner are either hindering the progress of computational methods in urban design or resulting in an oversimplified geometry that does not accurately represent the urban topology (Shirinyan and Petrova-Antonova, 2022). The latter is one of the main contributors to the sustainability performance gap seen in the industry. In building aerodynamics, generating accurate geometry both for a study building and its sur- rounding are the most important elements (links 2 and 3) of the Alan Davenport wind load chain (Fig. 3.1), which explains the linked steps to assess wind effects. Aerodynamics in fact means the study of the effect of shape. Similar linked steps are followed for the design of other climate stressors as well.

In recent years, however, the availability of open geospatial data, such as satellite imagery, building footprint vector data and LiDAR point clouds is creating opportunities to generate large-scale 3D city models at low cost as new machine learning techniques have significantly

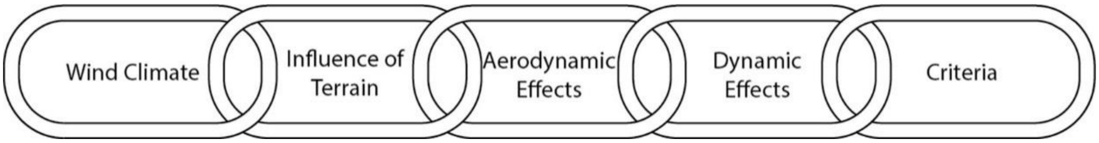


Figure 3. 1 The Alan G. Davenport wind loading chain (Isyumov, 2012).

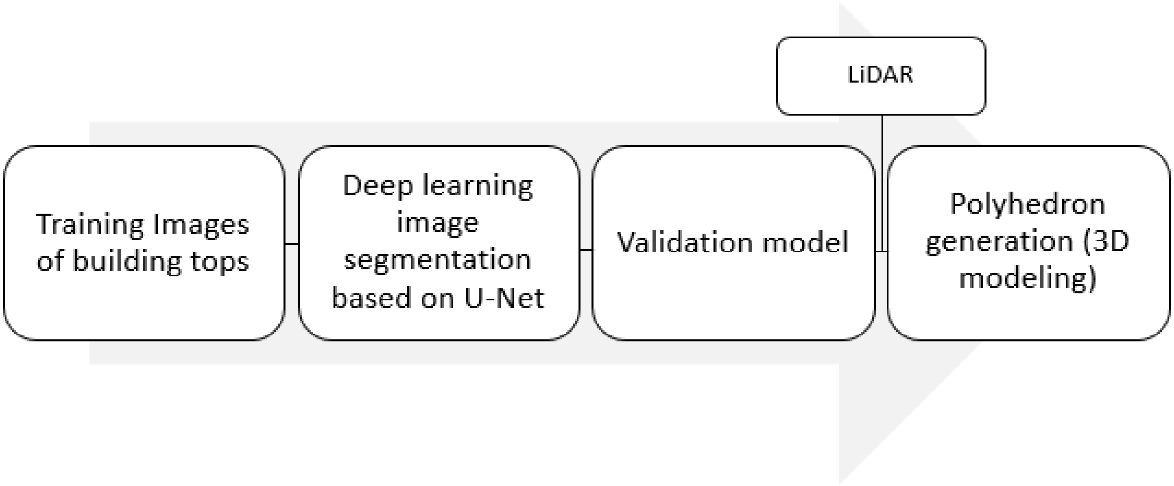

LiDAR
Training Images of building tops
Deep learning image segmentation based on U-Net
Validation model
Polyhedron generation (3D modeling)

Figure 3. 2 Workflow for 3D modelling of buildings.

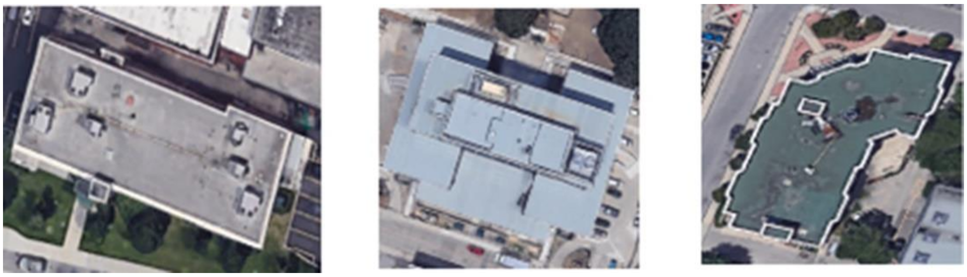


Figure 3. 3 Sample of training images used to train the deep learning network.

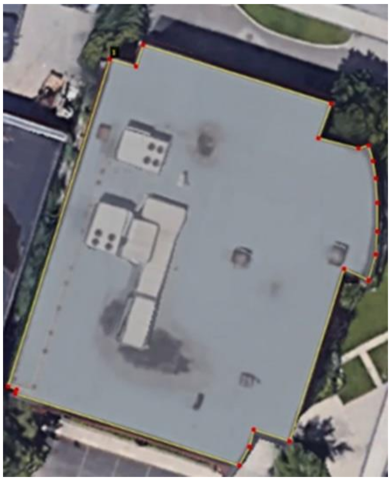


Figure 3. 4 Training image labelling.

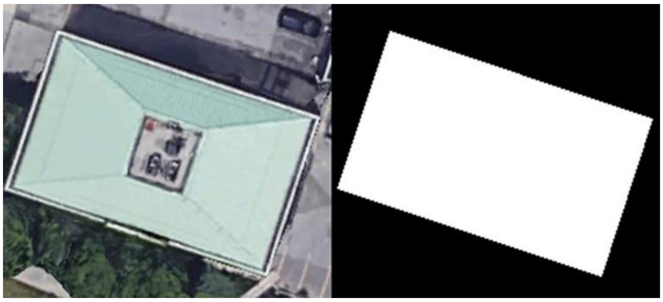


Figure 3. 5 Sample training image with its corresponding mask.

expanded in photogrammetry, remote sensing, and machine vision (Park and Guldmann, 2019). The present study focuses on developing a framework that combines automated urban topology generation and numerical modelling. In autonomous urban 3D modelling topology generation, the first and critical part is building footprint polygon extraction from a given input image data. Various researchers have attempted to extract building footprint polygons from different types of data by employing various techniques such as region growing method (Li et al., 2019), a filtering algorithm (Susaki, 2012) that was used to filter ground points (GPs) from raw airborne point cloud measurements and generate an estimated digital terrain model (DTM).

The extensive time required to create city-scale digital mapping has limited the application of digital images and remote-sensing image models (Guler and Yomralioglu, 2022). To date, scene generation for urban physics applications is a laborious, time-intensive process, as the building polygons, the terrain model, CAD objects and background maps must be created and

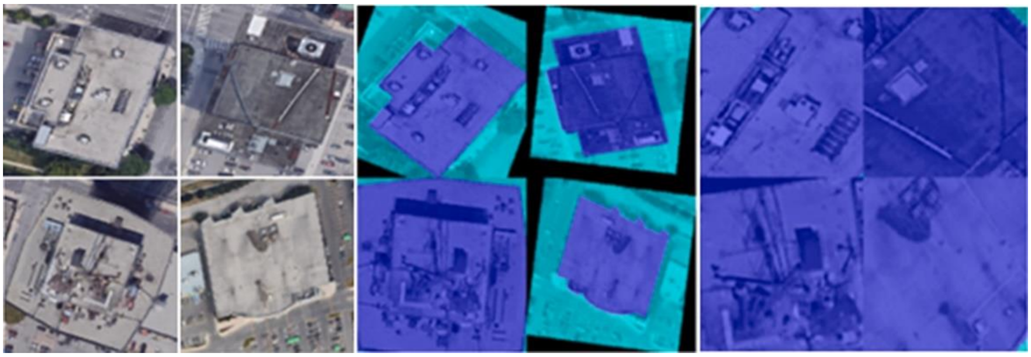


Figure 3. 6 Data augmentation sample results.

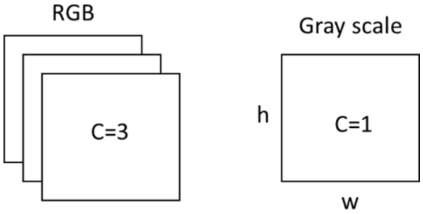


Figure 3. 7 RGB and grayscale images with their respective dimensions.

attributed manually. To shorten the time required for this process, (Lach et al., 2006) initiated an effort that aimed to reduce the man-in-the-loop requirements for several aspects of synthetic hyperspectral scene construction. Satellite imagery may be used for specific purposes if effective algorithms are used to obtain the required information (Richner, 2011). However, traditional photogrammetric methods do not offer this option (Chen et al., 2004). Thus, this thesis proposes a cutting-edge deep learning modelling approach trained on satellite imagery which can identify buildings from new sets of image data and combine it with LiDAR data to generate the 3D models.

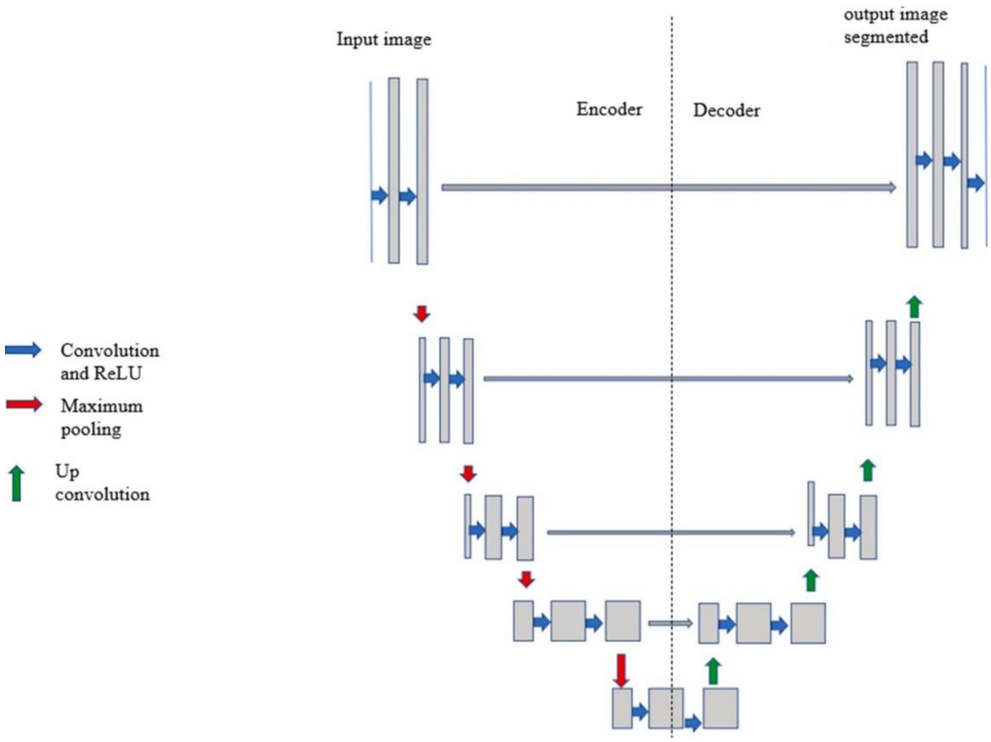


Figure 3. 8 U-Net-based deep learning model architecture.

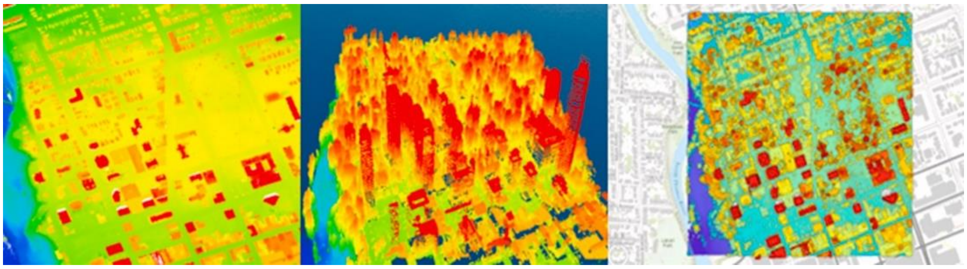


Figure 3. 9 3D LiDAR data used for model generation.

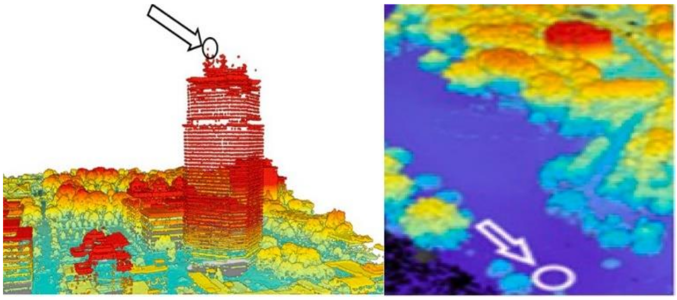


Figure 3. 10 Point clouds on the tallest building can help to filter out outliers and the lowest points are found on rivers usually.

It is recognized that methods that employ LiDAR point cloud data may suffer (a) vertically if the height position from which the point cloud data obtained is limited and useful data may be missing in areas where a higher resolution is required (Holmgren et al., 2003), (b) horizontally, where points may be recorded off their original location due to GPS and navigation unit operation anomalies (Park and Guldmann, 2019), and (c) overall, from missing data due to absorption or reflection of laser energy (Minato et al., 1998) and it may contain noise that will affect the final output classification feature extraction phase (Gao et al., 2013) as LiDAR usually operates at a monochromatic wavelength measuring the range and the strength of the reflected energy (intensity) from objects limiting recording of a diversity of spectral reflectance (Morsy et al., 2017). The proposed deep learning model to extract building footprints from imagery data by justifying its validity will also help address some of these shortcomings.

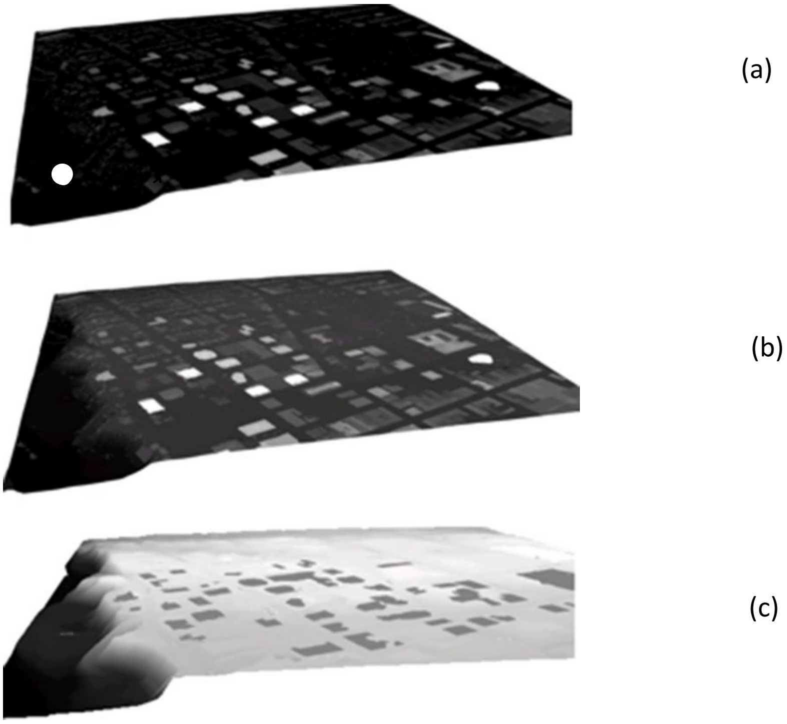


Figure 3. 11 (a) normalized Digital Surface Model (nDSM), (b) Digital Surface Model (DSM) and (c) Digital Terrain Model (DTM) values of the LiDAR data.

As we live in a 3D world, the recognition and analysis of 3D geometric models is an inevitable problem. With the emergence of large 3D repositories in the last several years, classification, retrieval, and semantic labelling of 3D objects is becoming possible, and these areas have drawn great attention from researchers.

Realistic building models required in computational fluid dynamics (CFD) studies are not always readily available for engineers carrying out a study in a specific project spot (Lee et al., 2006). The manual computer-aided design (CAD) modelling approach may achieve the required level of accuracy, but it is a time-consuming procedure, and it takes the most valuable engineering and CAD technician time. This time-consuming process gets even more problematic when the study area is large and the building form is more elaborate and complex, a common case in modern architecture. Furthermore, importing, and fixing CAD models to make them air/watertight for CFD analysis takes longer time and guaranteeing surface integrity during mesh generation is difficult. If the 3D model generated has intersecting faces, it may cause difficulty in generating a grid around a complex geometry which hinders the efficient use of the CFD in engineering analysis and design. Thus, this will cause the surface grid-generating process to be a time-consuming procedure. The proposed workflow in this thesis is aimed at alleviating some of these issues.

When zooming in on the construction of 3D building models from building footprint polygons, there are studies that used hierarchical Euclidean clustering (Li et al., 2019) and graph cut algorithm (Lee et al., 2006) but with limited success. Convolutional neural networks (CNNs) are showing promising results in the 2D pattern recognition field (Weng and Zhu, 2021, He et al., 2020) and 3D object classification (Wang et al., 2019). A study from Hong Kong Polytechnic University explored and attempted to provide a solution to the problems of developing a methodology to fuse terrestrial laser scanner-generated 3D point cloud data and high-resolution digital images. Four phases of the methodology that have been investigated were reported in the study (i) data pre-processing (fusion of data from the two sensors), (ii) automatic measurements (feature detection and correspondence matching), (iii) mapping (creation of point cloud visual index), and (iv) orientation (calculation of exterior orientation parameters) (Telkamp, 1981). Although the researchers were able to obtain compact 3D models, artifacts were observed whenever there is raw data that has a lower resolution, missing data due to absorption or reflection of laser energy, a gap because of a small alley between buildings and curved structures were not accurately represented. An urban geometry reconstruction technique paper for real-life urban geometries reproduced roof shapes and for ground profiles it presented digital geographic information approach (Oshima et al., 2014). The types of the geographic dataset used for the

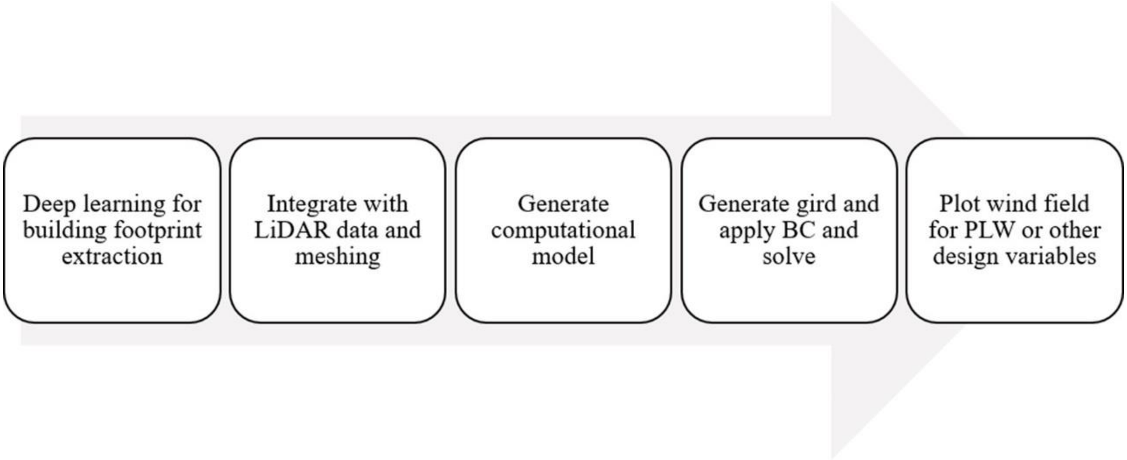

Deep learning for building footprint extraction
Integrate with LiDAR data and meshing
Generate computational model
Generate gird and apply BC and solve
Plot wind field for PLW or other design variables

Figure 3. 12 Summary of workflow for urban geometry generation and climate modelling.

Table 3. 1 Inlet boundary conditions (Richards and Hoxey, 1993, Richards and Norris, 2011).

Velocity profile	Turbulence intensity profile	Integral length
$u(z) = (u_*/ k) \ln((z + z_0) /z_0)$	$I_u(z) = 1/\ln((z + z_0) /z_0)$	$L_u(z) = 12z^{0.6}$
	$Iv (z) = 0.75Iu (z)$	$Lv (z) = 0.25Lu (z)$
	$Iw (z) = 0.5Iu (z)$	$Lw (z) = 0.5ILu (z)$

reconstructions were a digital surface model and a two-dimensional building outline map. But this method will only be valid if there are up-to-date building outline polygons. In workflows that use only satellite imagery analysis methodology, the most obvious shortcoming is that the image only displays the top part of buildings and height information is missing (Sowmya and Trinder, 2000). That is why the point cloud data should be combined with the image data to give an overall representative 3D model of a building. The attempts of the above methods to generate 3D building models gave rise to a consensus that combining image and point cloud data will generate a better outcome. In line with this, the research by (Kwak et al., 2012) automatically generated building models using the Minimum Bounding Rectangle algorithm and sequentially adjusted them with LiDAR datasets to generate better results. After weighing the pros and cons of different approaches and acknowledging the capability of deep learning, this thesis proposes a deep learning-based image segmentation combined with point cloud data to generate 3D building models.

The present study by combining the proposed deep learning-based building footprint extraction and LiDAR informed 3D building construction with computational modelling, will present a novel workflow for autonomous urban topology generation and urban flow modelling.

The developed method will be illustrated through a pedestrian-level wind assessment (PLW) application for the downtown region of the City of London, ON, Canada. One of a city's environmental design parameters is pedestrian assessment issues related to the accelerated wind (comfort and safety), or lack of wind (ventilation) caused by local climate inter- action with local terrain, buildings, and vegetation.

Taller buildings block the wind and redirect it to the low-velocity region near the ground or accelerate it to the sides of the building causing a higher wind speed at the pedestrian level. City corridors also create a venturi effect resulting in increased wind speed.

The wind speed could increase to three or four times more than commonly experienced in towns (Adamek et al., 2017, van Druenen et al., 2019). PLW speed can lead to uncomfortable and even dangerous conditions for pedestrians.

Untenanted shops because of a windy environment and the death of two elderly people due to a fall caused by high wind speeds at the base of a high-rise building were observed (van Druenen et al., 2019).

The opposite phenomenon happens when the city grid prevents having sufficient airflow to clean pollutants from city corridors or causing low flow regions where snow drifts and excessively accumulate at entrances or pathways (Sowmya andTrinder, 2000, Phillips et al., 2019) (similar problem is also observed in dry regions with sand drifts).

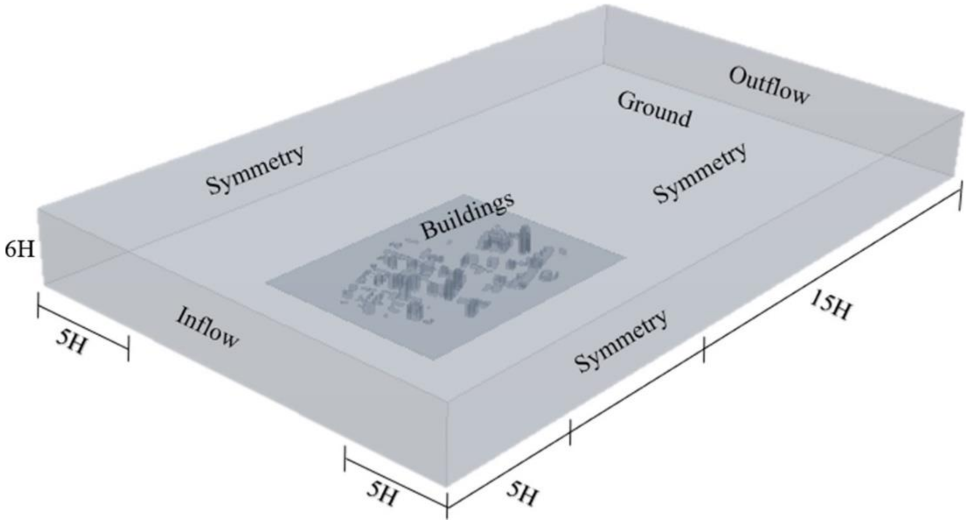


Figure 3. 13 Computational domain that used the automated urban topology.

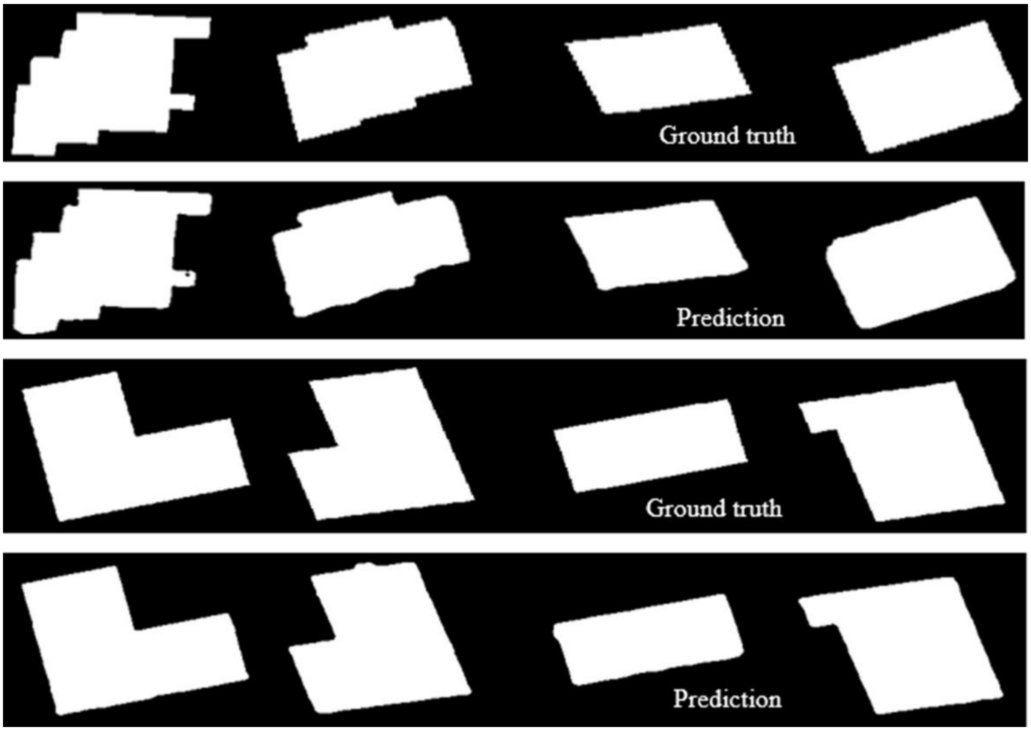


Figure 3. 14 Prediction results from the deep learning model training.

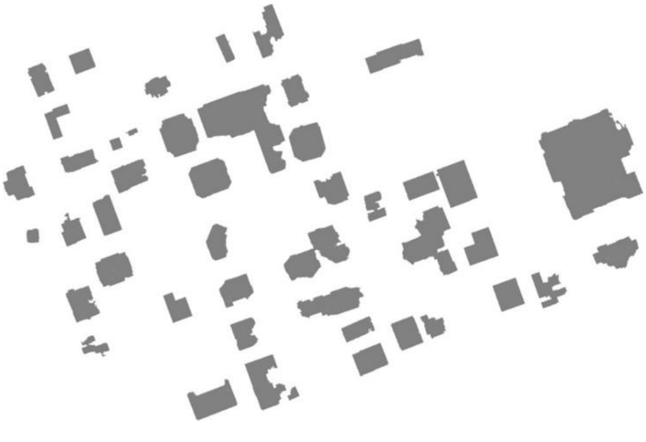


Figure 3. 15 The predicted building footprints of downtown area in London, ON.

In spite of the increasing awareness of the importance of wind comfort and wind safety, the relation between urban geometry and PLW comfort is often unclear for designers (van Druenen et al., 2019). Alterations or additions in favour of PLW comfort and wind safety are sometimes found to be visually displeasing, impractical, and expensive, and their effects are rarely investigated and often appear disproportionately small (van Druenen et al., 2019). Computational fluid dynamics (CFD) PLW simulations can provide near-optimal solutions through iterative simulations that involve the key stakeholders.

In order to determine PLW comfort, statistical meteorological data of nearby weather stations, aerodynamic information of the area and mechanical wind comfort criteria are combined; these represent the first three links in the Alan G. Davenport wind loading chain Fig. 3.1 (Isyumov, 2012). The aerodynamic information is needed to transform the statistical meteorological data from the weather station to the location of interest at the building site, after which it is combined with a comfort criterion to judge local wind comfort (Janssen et al., 2013). The aero-dynamic information usually consists of two parts: the terrain-related contribution and the design-related contribution. The terrain-related contribution represents the change in wind statistics from the meteorological site to a reference location near the building site (Tong et al., 2005). The design-related contribution represents the change in wind statistics due to the local urban design, i.e. the configurations of buildings (Janssen et al., 2013). The latter is the focus of the present study.

The 3D building models generated are used as an input to determine aerodynamic information of the area of interest using CFD simulation. CFD simulations are routinely performed by the relatively low-cost steady Reynolds-Averaged Navier Stokes (RANS) approach (Tominaga and Stathopoulos, 2010) due to the high Reynolds number flow problems that render Large

Eddy Simulation (LES) based simulations too costly for project-by-project simulation. As a result, RANS is one of the widely used approaches for PLW studies (Phillips et al., 2019), and is likewise popular in research areas such as near-field pollutant dispersion in urban areas with high plan area density, urban thermal environment, natural ventilation of buildings and indoor airflow (Blocken, 2018). The output from the current study can also be used for other types of CFD applications such as wind loading evaluation that require unsteady simulations, simulation of flows over terrain with topographical or orographic features, complements experimental data on flow generation (Bitsuamlak et al., 2010), assessment of the true potential for energy savings in a city (Sola et al., 2020) and improvement of the accuracy of CFD modelling for a 3D urban model that is generated by the combination of LiDAR data with remote sensing images (Su et al., 2014).

3.2 Methodology

The overall steps of the 3D building modelling are described in Fig. 3.16. Training images of building tops are initially collected to train a neural network. The network is then evaluated for accuracy based on its predicted building footprint polygons. Finally, the vector polygon and the LiDAR point cloud data are combined to generate 3D building surface mesh using the 2.5D approach. The details are discussed in the following sections.

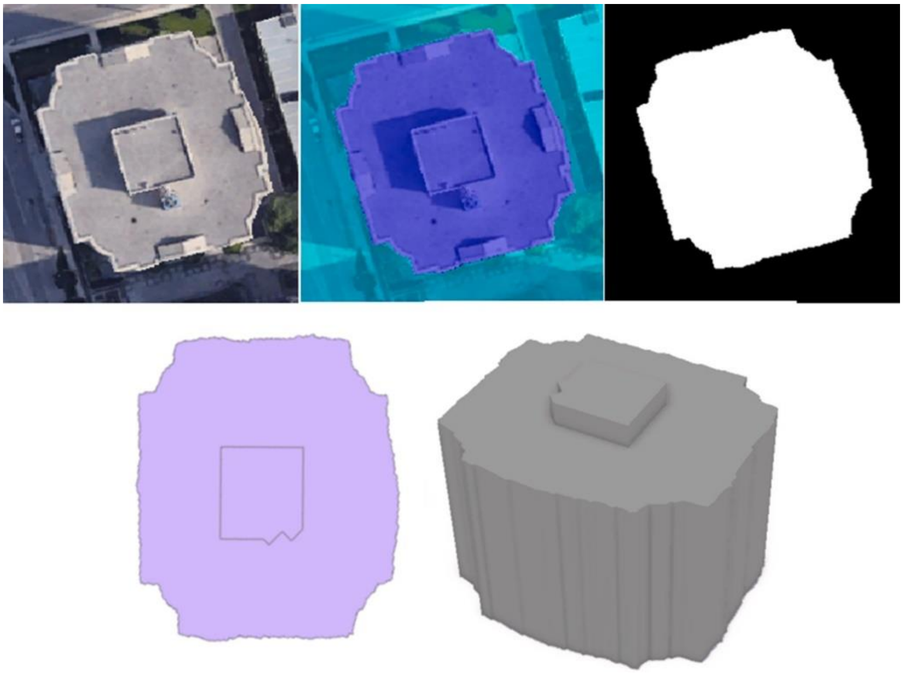


Figure 3. 16 A series of images showing the 3D model generation workflow transformations.

Table 3. 2 Accuracies of the segmentation results.

Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	Mean BF Score
98%	98%	97%	97%	92%

3.3 Deep learning model for building footprint extraction

3.3.1 Training Data Preparation

A deep learning process starts with collecting the required data for training a segmentation model. The data is collected from publicly available sources and local government offices (in the present case City of London, On). The data is organized in a way for an image to represent a single building so that it would be convenient for labeling it as a training set. The model is trained on building samples that are representative of the building construction style in the vicinity of the city and comprises images from cities which are found near London i.e., Hamilton, Kitchener, Mississauga and Guelph, ON (Fig. 3.3). A thousand training images are collected inline with the typical minimum requirement for a deep learning model. As in any deep learning model, the training data should be relatable to the data that will be predicted post-training.

Once the image data are collected and organized the next task is to label each individual image. There are various image labelling applications out there both open-source and commercial ones. In this study, VGG Image Annotator (VIA) open-source application is used to label the

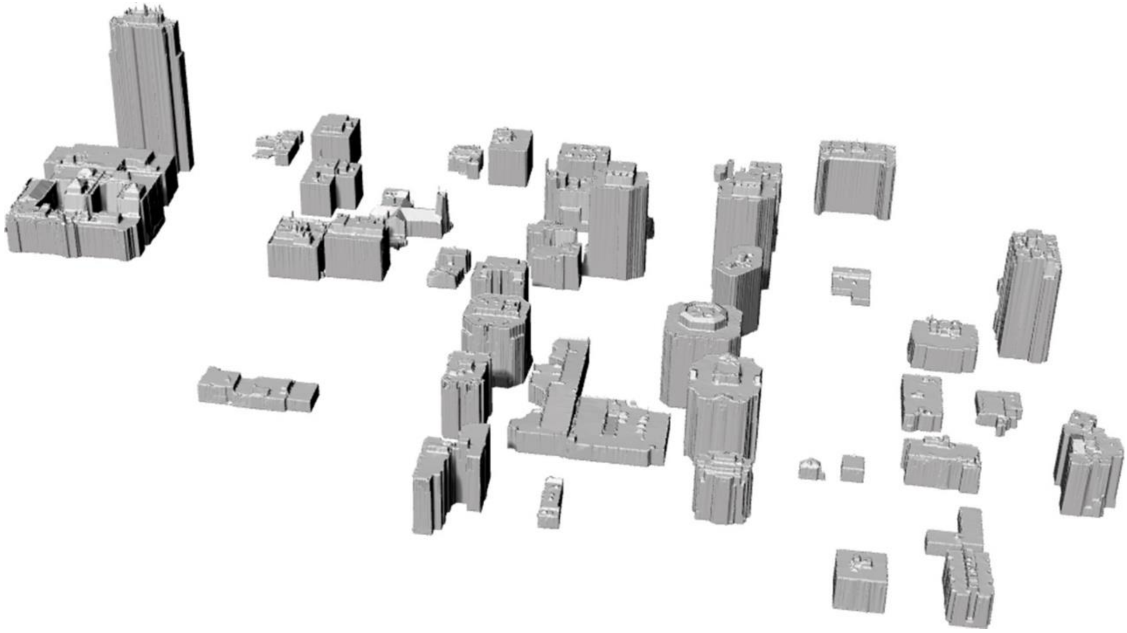


Figure 3. 17 3D building models generated using Deep learning technique.

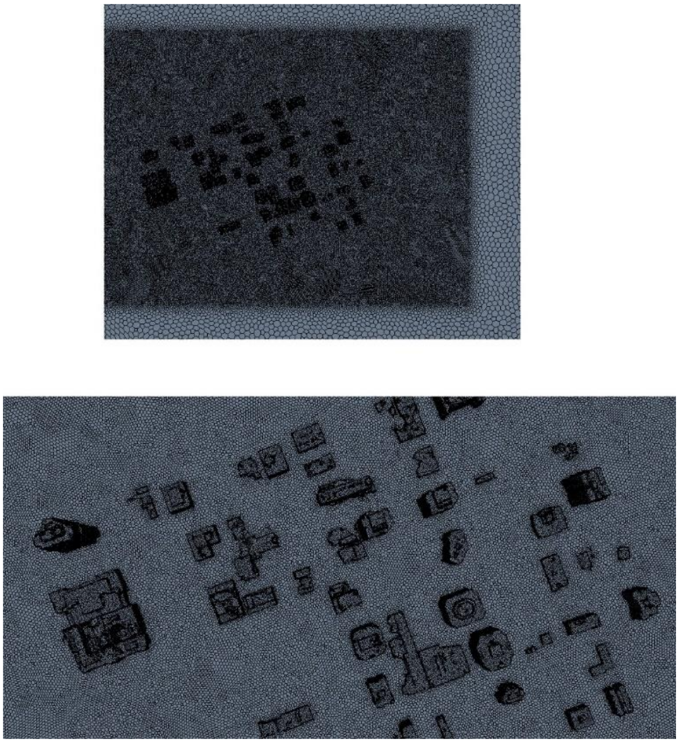


Figure 3. 18 Grid discretization of the 3D building models

images. The training images are loaded into the VIA application and labelled using polygons to mark the boundary of the buildings in the image. Fig. 3.4 shows a labelled image with the building outline marked by yellow lines. The area enclosed by the yellow polygon will be registered as a building. This way all the thousand images are labelled. The labelled images are then exported in JavaScript Object Notation (JSON) format. This helps to store data structures and objects in a standard data interchange format.

In order for the deep learning model to train using the labelled images efficiently, these JSON files need to be converted to image masks. The JSON files are converted to image masks using a python script. Sample images with their corresponding masks are shown in Fig. 3.5. Next, further pre-processing steps are applied to the images to make them more suitable for neural network training. As a common rule in machine learning applications, images are converted into grayscale images so as to simplify the dense data that comes with RGB images. Grayscale representations are often used for extracting shapes instead of colour images since grayscale simplifies the algorithm and reduces computational requirements (Kanan and Cottrell, 2012). Then the images are cropped to a size of 256-pixel length by 256-pixel width which is usually the rule of thumb in deep learning.

At this stage, the data is partitioned into training and validation data. A 10% validation data is used by taking the training size into account so that the model avoids overfitting. During writing the training code, two values are assigned to represent the buildings and background. The buildings are assigned a value of 255, and the background is assigned a value of zero to perform semantic segmentation.

Further, an image datastore is created to simplify data management. Image datastore enables the deep learning model to import data in batches from image collections that are too large to fit in memory. A datastore makes it convenient to call images into training workflow without the need to store the data in a workstation which would otherwise be computationally demanding. This way the code is allowed to read and analyze data from image folders in smaller portions that fit the available memory. The images are stored as PNG files in the data-store. For each training and validation image, a corresponding image mask is stored in a separate datastore ready to be called anytime the code runs. Basically, the datastore for the image mask is known as a pixel-label datastore since it contains images segmented at a pixel level.

The image datastore and its corresponding label datastore are combined into a single datastore so that it would be convenient for data augmentation. Then the labelled pixels are overlayed on the original image. This way the pixels representing the buildings align exactly with the building boundaries. Deep learning models register better performance when trained on a larger dataset. Thus, data augmentation is required to increase the amount of training data. Data augmentation also applies different transformations to the training data which helps the model learn new features during training. For instance, augmentation applies randomized rotations and resizing to input images so that the model would be familiar with the presence of rotation and sizing in an input image (Fig. 3.6).

3.3.2 Deep learning model architecture

The deep learning model's first layer is set to accept an image size of grayscale. As mentioned earlier, grayscale images are more convenient to process and make the training phase faster. The input image is specified as a row vector of integers. The image size [256 256 1] corresponds to [h w c], where h is the height of the image, w is the width of the image, and c is the number of channels. A channel value of 1 indicates it's a grayscale image. For RGB images, the channel value will be 3, representing red, green, and blue colours, respectively (Fig. 3.7). If needed, the layer can be converted to accept RGB images. At this layer, the input image is processed so that the mean of the image lies at zero. This normalization helps to convert the image into a range of pixel values that have similar data distribution. This helps the network to converge faster during training.

The deep learning approach implemented is an algorithm that used the U-Net architecture (Weng and Zhu, 2021) as a backbone to execute semantic segmentation of building footprint (Fig. 3.8). The encoder part applies convolution blocks followed by a maximum pooling down sampling to encode the input image into feature representations at multiple different levels. The decoder part semantically projects the lower resolution features learnt by the encoder onto pixel space (higher resolution) to get a dense classification. A convolutional layer uses convolutional filters on an image. The filters move along the input image vertically and horizontally. This phase computes the dot product of the filter (kernel) and input image pixels and adds a bias term. There are 64 filters used in this stage with each of them of size 3 by 3 pixels. The stride describes the rate of movement for the filter. In this case, the first value 1 indicates a stride of one pixel in the vertical direction. The second value of 1 represents a horizontal movement of the filter. Padding is applied to keep the size of the output image the same as the input image. Padding helps to achieve a more accurate analysis of images by adding an outer frame on an image to allow for more space for the filter to cover an image. The general expression of a convolution kernel filter is defined by the expression in equation 3.1.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k] \qquad \text{Eq.3.1}$$

The input image is denoted by *f* and the filter kernel by h. The resulting matrix or the filtered image is marked by G *m*, *n* and *j* and *k* represent every element of the filter kernel.

The Rectified Linear Unit (ReLU) layer performs an operation on each element of input and sets any value less than zero to zero for better computation performance (equation 3.2). The maximum pooling step-down samples of an incoming image by dividing the input into pooling regions and computing the maximum value of each region.

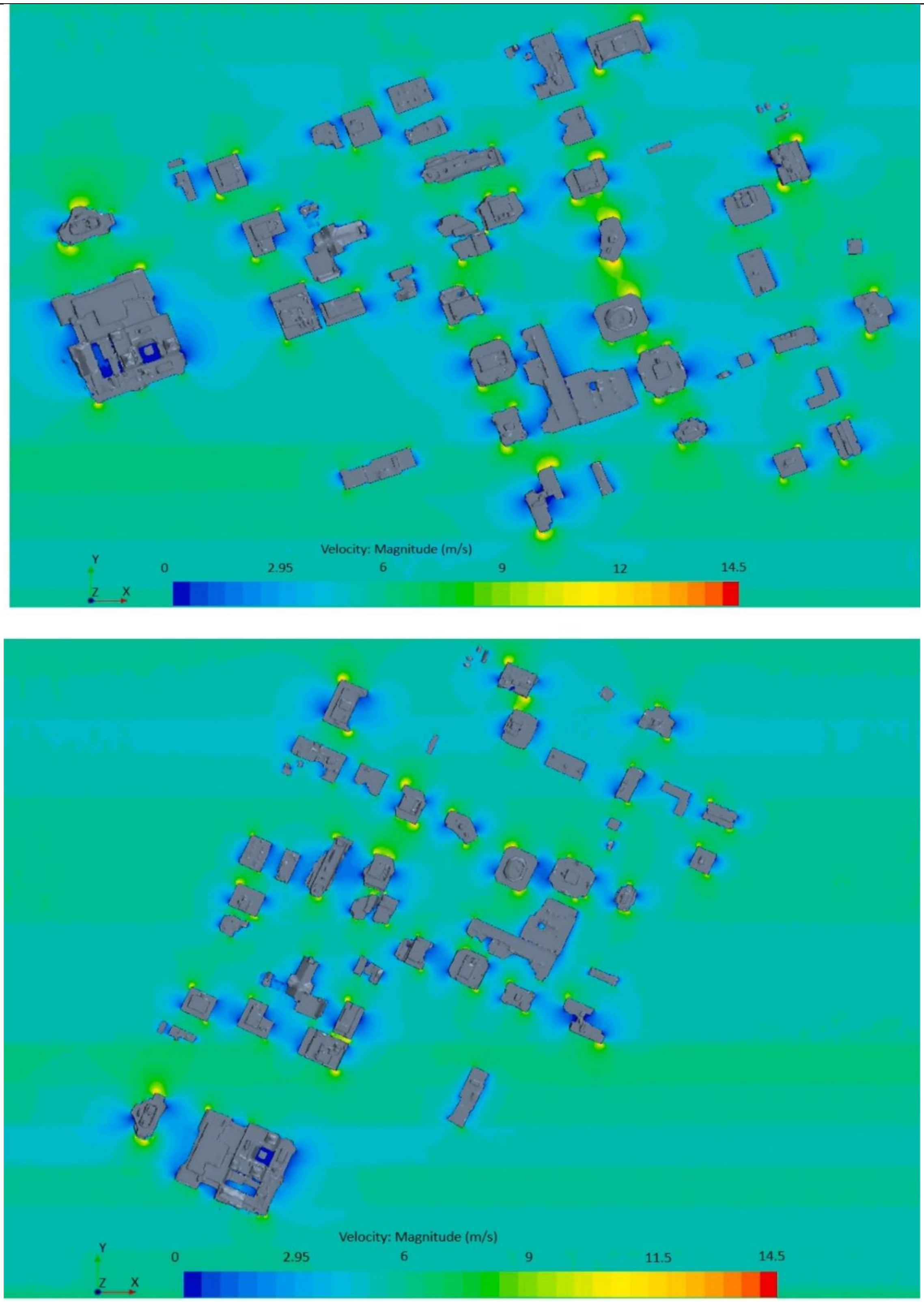


Figure 3. 19 RANS CFD simulation at 0- and 45-degree wind angle of attacks.

Table 3. 3 Pedestrian wind comfort criteria (Adamek et al., 2017).

Comfort category	Gust Equivalent Mean Speed* m/s (km/h)	Description
Sitting	≤ 2.7 (10)	Calm or light breezes are desired for outdoor restaurants and seating areas where one can read a paper without it blowing away
Standing	≤3.8 (14)	Gentle breezes suitable for main building entrances and bus stops
Strolling	≤4.7 (17)	Moderate winds that would be appropriate for window shopping and strolling along a downtown street, plaza, or park
Walking	≤5.5 (20)	Relatively high speeds can be tolerated if one's objective is to walk, run or cycle without lingering
Uncomfortable	>5.5 (20)	Strong winds of this magnitude are

		considered a nuisance for most activities, and wind mitigation is typically recommended
Exceeded	> 25 (90)	Excessive gust speeds can adversely affect a pedestrian's balance and footing. Wind mitigation is typically required.

* *GEM is defined as the maximum mean wind speed or gust speed divided by 1.85 (whichever is larger).*

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad \text{Eq. 3.2}$$

where *x* is the maximum value selected from a region of pixels. If *x* has a negative value, then a 0 value is assigned. The maximum pooling reduces the number of parameters to learn, and the amount of computation performed in the network.

After testing three types of optimizing algorithms (stochastic gradient descent with momentum, RMSProp and Adam), the Adam (derived from adaptive moment estimation) optimizer provided a better performance compared with the other methods during training. Hence, Adam is used to training the neural network. The Adam algorithm is used for the deep learning workflow to train a segmentation model. Adam (Kingma and Ba, 2015) uses a parameter update that is similar to RMSProp, but with an added momentum term. It keeps an element-wise moving average of both the parameter gradients and their squared values,

$$m_\ell = \beta_1 m_{\ell-1} + (1-\beta_1)\nabla E(\theta_\ell) \qquad \text{Eq.3.3}$$

where m_ℓ is a moving average, ℓ is the iteration number, β_1 is the gradient decay rate, θ is the parameter vector, $E(\theta_\ell)$ is the loss function, $\nabla E(\theta_\ell)$ is the gradient of the loss function

$$v_\ell = \beta_2 v_{\ell-1} + (1-\beta_1)[\nabla E(\theta_\ell)]^2 \qquad \text{Eq.3.4}$$

where v_ℓ is a moving average, β_2 is the squared gradient decay rate. Adam uses the moving averages to update the network parameters as

$$\theta_{\ell+1} = \theta_\ell - \frac{\alpha m_l}{\sqrt{v_l}+\epsilon} \qquad \text{Eq.3.5}$$

where $\alpha > 0$ is the learning rate, ϵ is a small constant added to avoid division by zero.

If gradients over many iterations are similar, then using a moving average of the gradient enables the parameter updates to pick up momentum in a certain direction. If the gradients contain mostly noise, then the moving average of the gradient becomes smaller, and so the parameter updates become smaller too. The full Adam update also includes a mechanism to correct a bias that appears at the beginning of training.

3.4 3D building model generation by integrating building footprint and 3D Lidar point cloud data

This thesis tested the deep learning model in London, Ontario downtown area. A smaller portion area of downtown is selected for the test. The area is shown in Fig. 3.9 represented by the LiDAR data. One of the requirements to model a 3D building is to identify its height. That information is obtained from LiDAR data. The data has 13M LiDAR points. The LiDAR point density is 26 points/m^2. The advantage of using LiDAR data is that it comes with data that is already georeferenced. This makes it easy to integrate with existing data as shown in Figure 3.9. Thus, the time it takes to geo-reference data is saved.

First, a check for outliers which are point cloud points that may have different values is done. Outliers may be birds, planes or any other object that may have been picked up during data collection. It is important to remove outliers so that they won't undermine the overall output result. The outliers are filtered out by setting a maximum height and the minimum height that can be used in the test. In the raw LiDAR data, the maximum elevation observed is 371.4 meters and the minimum elevation is 150 meters (Fig. 3.10). Using such measurements directly in the workflow may cause the 3D models to be distorted. Thus, a reasonable height range of the study area should be specified. The observed elevation of the tallest building in the data is 368 meters and the lower elevation on the riverside is 230 meters. Therefore, these values are set as the new minimum and maximum values and any values below or above the specified figures are considered outliers. The tallest building in London is the One London Place building which is a 24-storey, 113.4 meters in height. The ArcGIS platform is used to merge the LiDAR data and the building footprints.

The elevation of buildings is extracted from the LiDAR dataset by converting the point cloud data into a raster format. The rasterization process resulted in a digital terrain model (DTM), which shows only the elevation of the ground, without buildings or other features. Secondly, the digital surface model (DSM) which shows the elevation of the ground and features on the ground is generated. Finally, normalized DSM (nDSM) which shows the height of features above ground (the normalized elevation) is generated during rasterization (Fig. 3.11). All three-elevation data are used in the workflow to calculate buildings' height.

Once the building outline is obtained, the next step is to georeference the image with data that is already georeferenced. The LiDAR data can be used as a georeferencing image since it comes already georeferenced. Using control points, in this case building corner points, a correlation is created between the building polygon image and its corresponding pattern in the LiDAR data.

The raster image is converted into polygon shapes so that the building polygon can be extracted in the next phase. Once the polygons are created, the building polygon needs to be further refined to remove the extra layers that are part of the raster image. Finally, the building footprint boundary has been successfully extracted from the raster image. But still, the building lines may appear slightly irregular at this phase. So, applying regularization gives a well-defined edge to the building sides in this step. This thesis is able to achieve a level of detail 3 (LOD3), which includes the building boundary defined by the building footprints extracted earlier and the height and roof structure details defined using the LiDAR data.

3.5 Pedestrian Level Wind Assessment on Autonomously Generated 3D Building Models

3.5.1 Governing equations and boundary conditions

In this study, a high-resolution, Reynolds-Averaged Navier-Stokes (RANS) CFD steady simulations using Shear Stress Transport (SST) k-ω is used (Fig. 3.12). The inlet boundary conditions specify the velocity, turbulence intensity and integral length profiles as given in Table 3.1. respectively. Symmetry boundary conditions on the side walls and ceiling of the computational domain are used. A pressure outlet is used on the downstream outlet. The flow parameters and initial conditions used for simulation are mass density of the air, ρ = 1.29 kg/m3, static pressure of air p = 101.3 KPa, ground roughness length (Z_0) = 1m for an area in which at least 15% of the surface is covered with buildings and their

Figure 3. 20 RANS CFD simulation at 90- and 135-degrees wind angle of attacks.

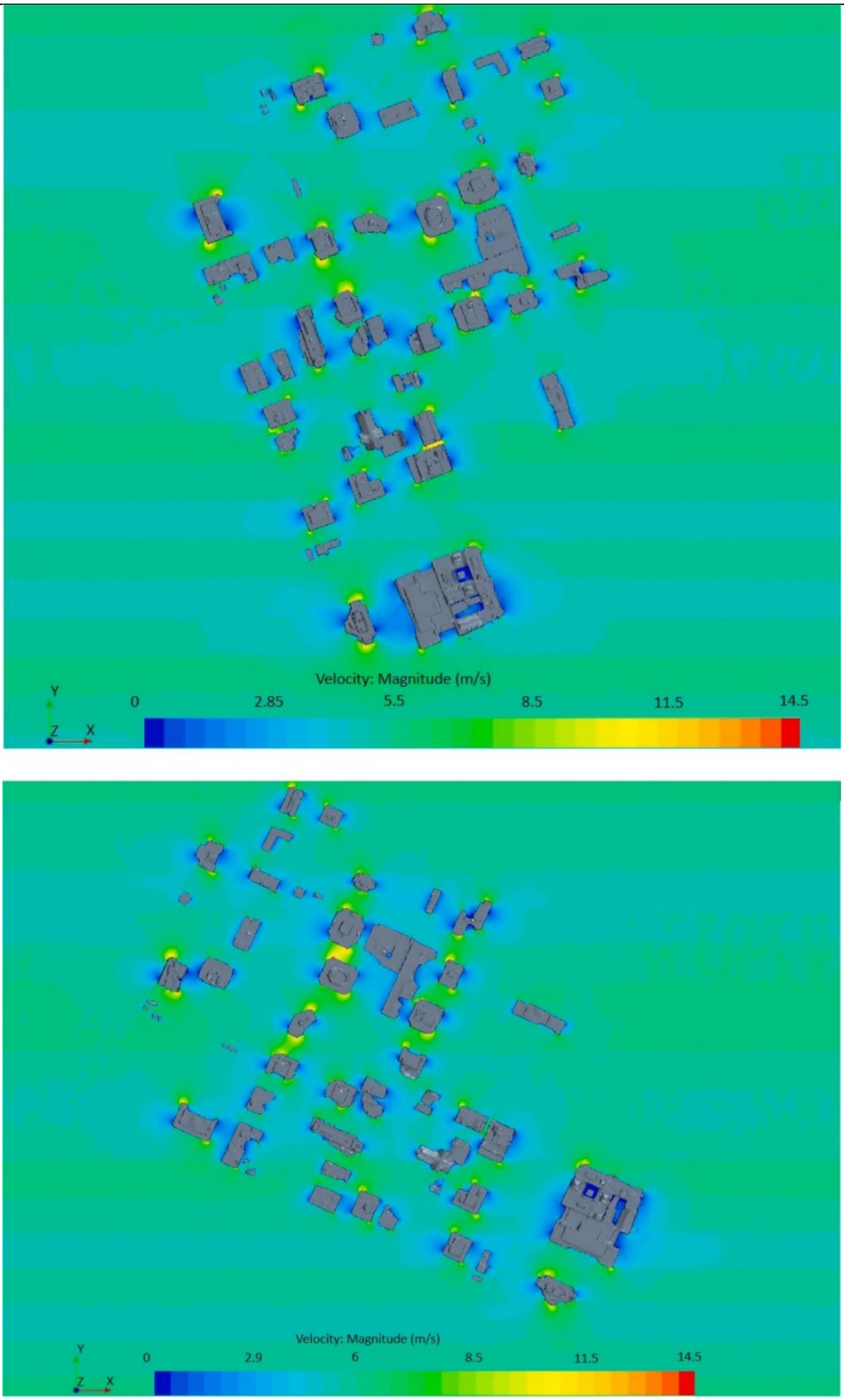


Figure 3. 20 RANS CFD simulation at 90- and 135-degrees wind angle of attacks.

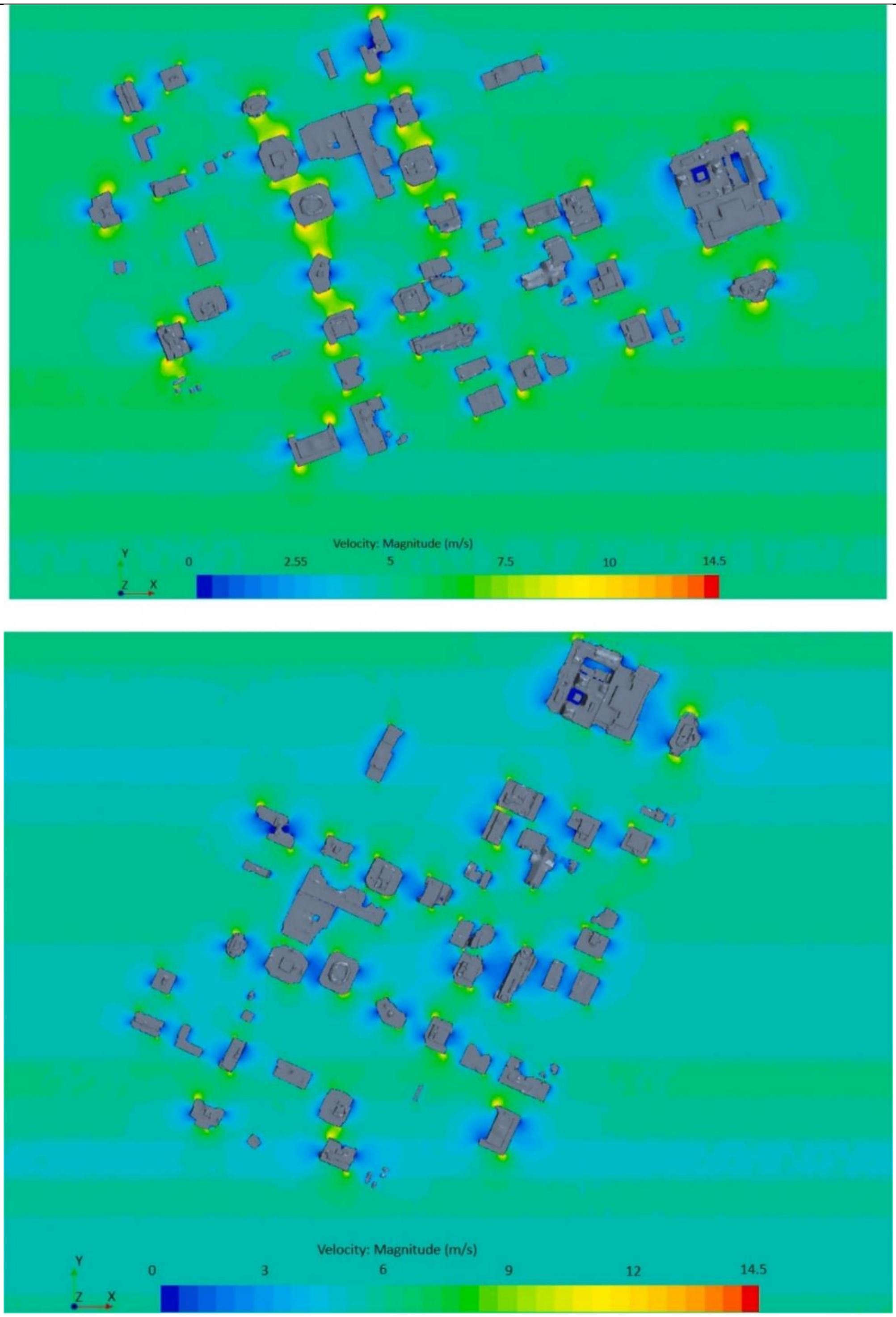


Figure 3. 21 RANS CFD simulation 180- and 225-degrees wind angle of attacks.

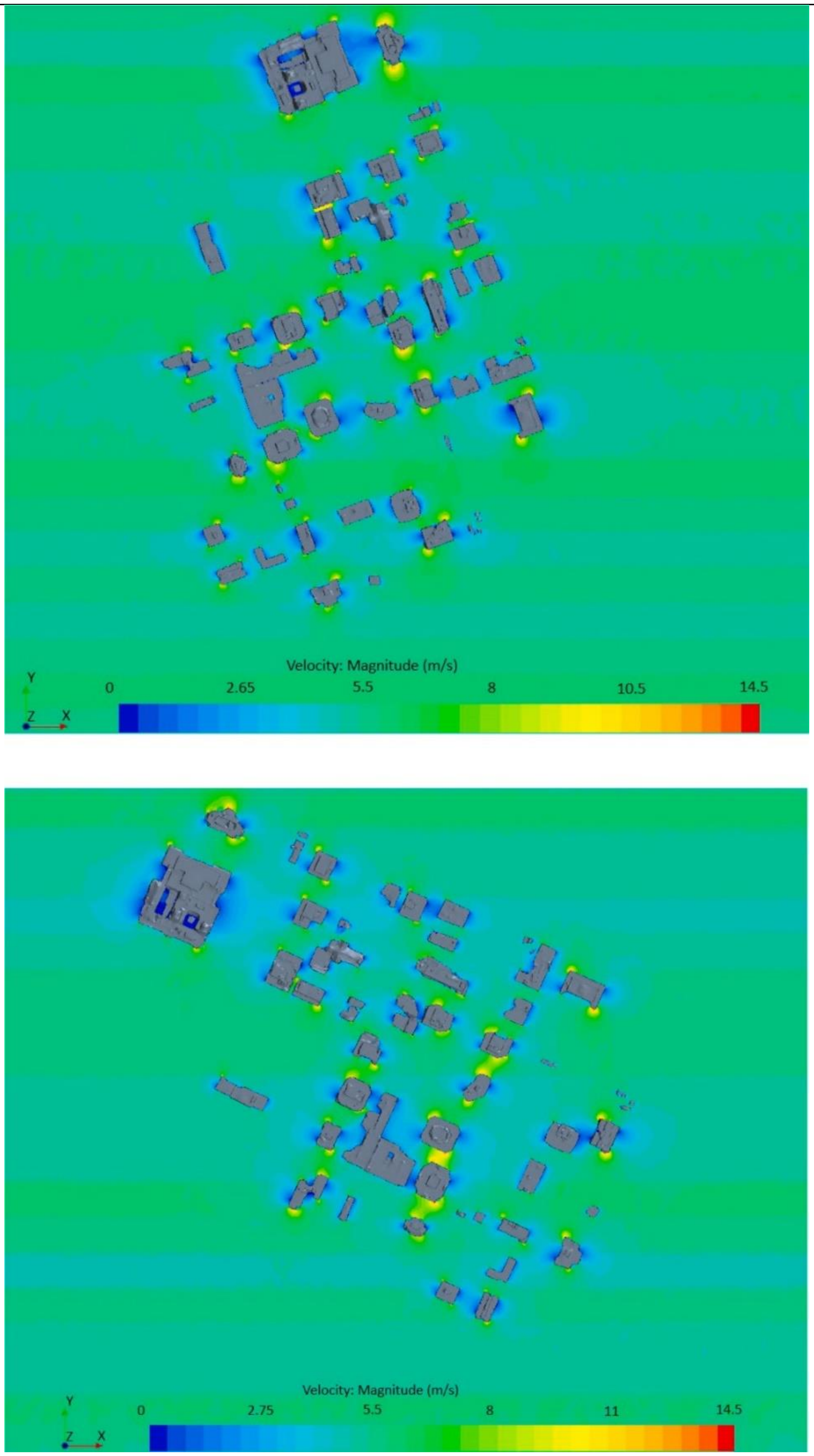


Figure 3. 22 RANS CFD simulation at 270- and 315-degrees wind angle of attacks.

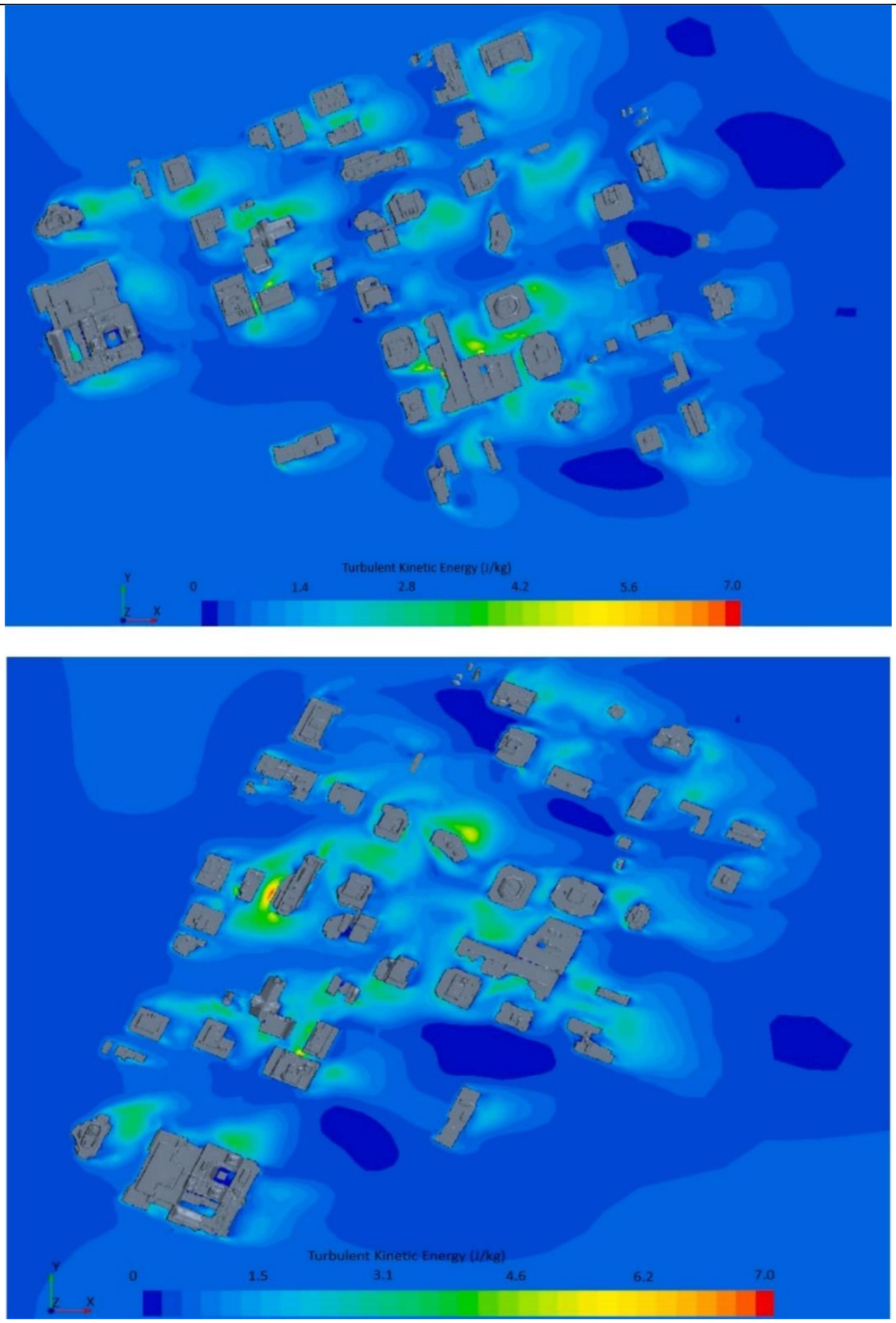


Figure 3. 23 Turbulent kinetic energy simulation at 0- and 45-degree wind angle of attacks.

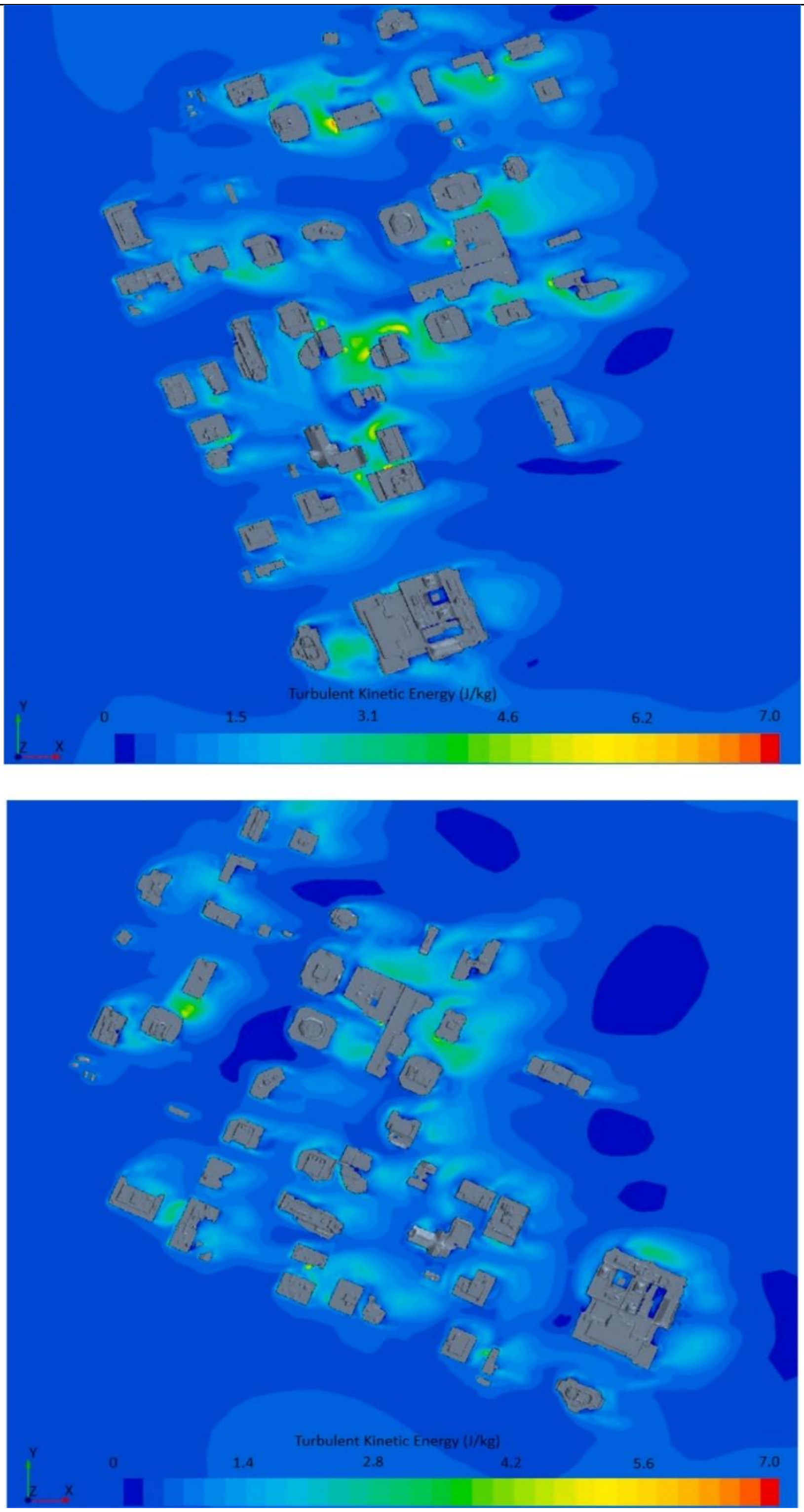


Figure 3. 24 Turbulent kinetic energy simulation at 90- and 135-degrees wind angle of attacks.

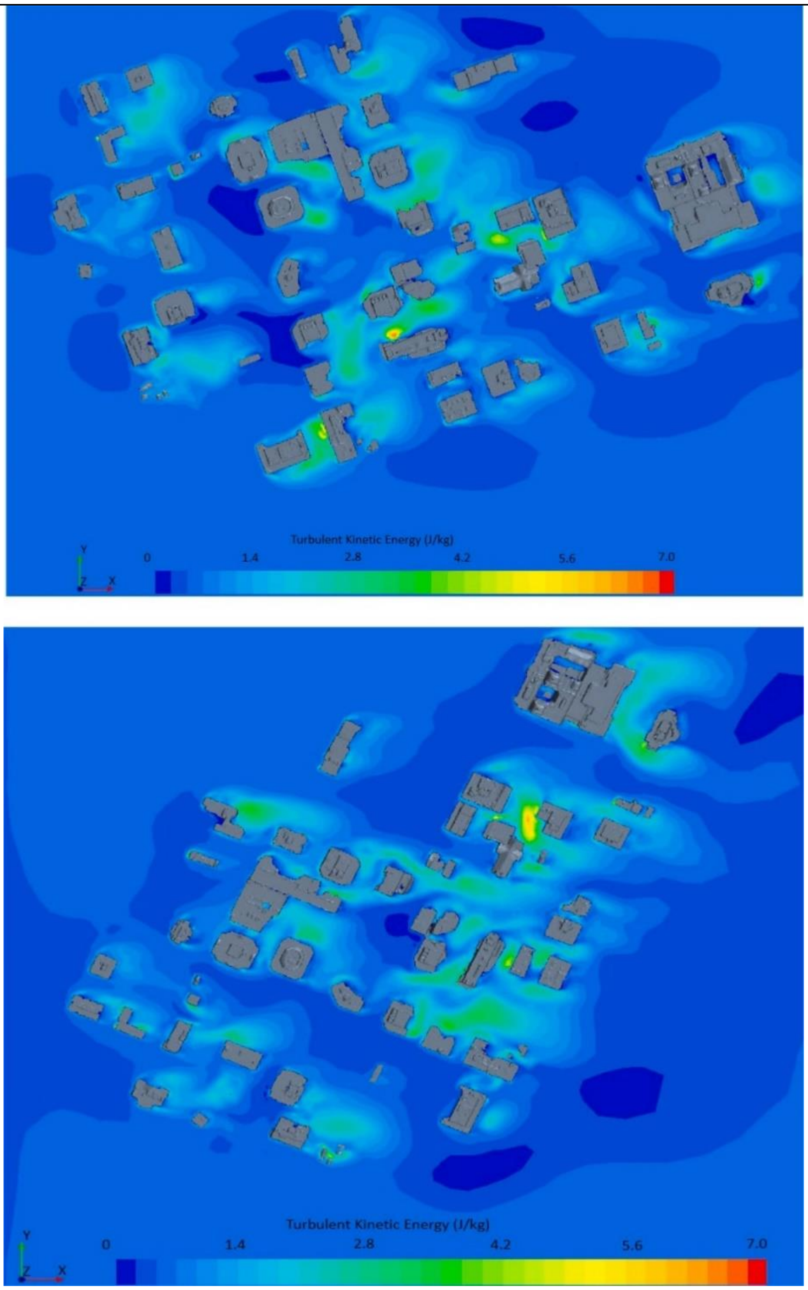

Turbulent Kinetic Energy (J/kg)
0
1.4
2.8
4.2
5.6
7.0
Y
Z
X
Turbulent Kinetic Energy (J/kg)
0
1.4
2.8
4.2
5.6
7.0
Y
Z
X

Figure 3. 25 Turbulent kinetic energy simulation 180- and 225-degrees wind angle of attacks.

Figure 3. 26 Turbulent kinetic energy simulation at 270- and 315-degrees wind angle of attacks.

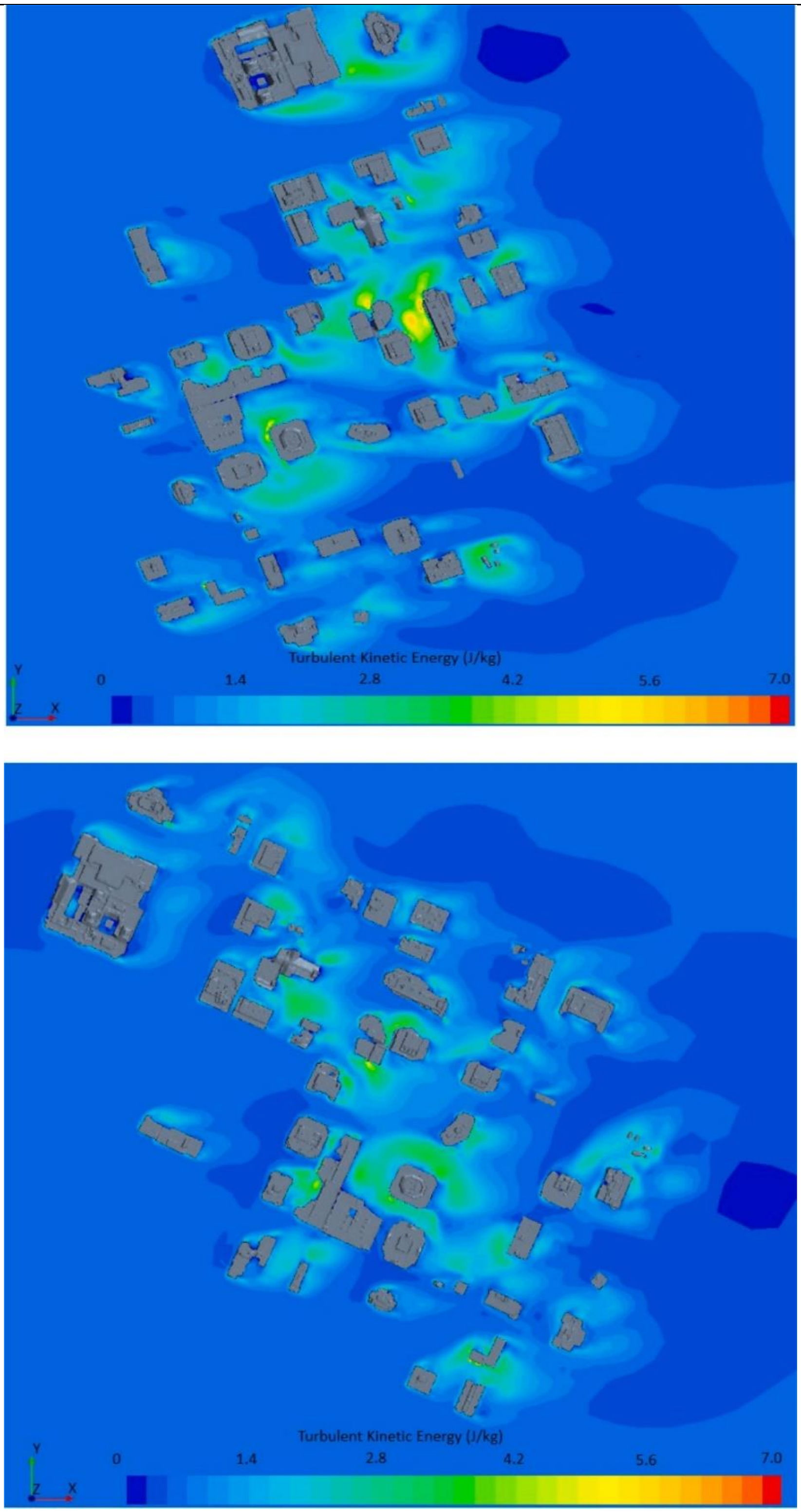
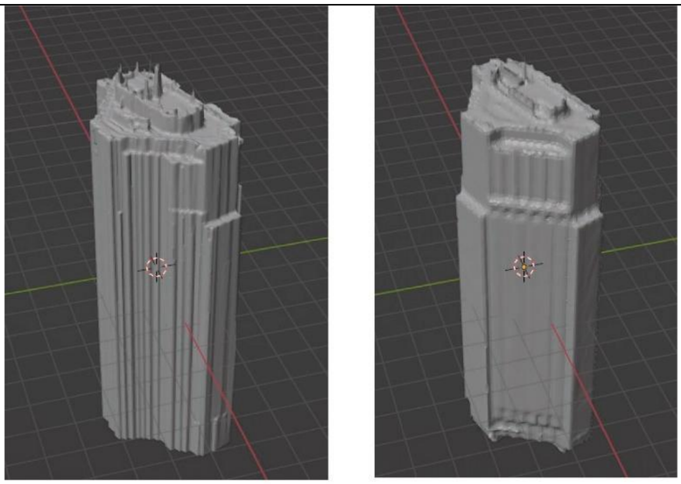


Figure 3. 26 Turbulent kinetic energy simulation at 270- and 315-degrees wind angle of attacks.



Predicted 3D model

Actual 3D model

Figure 3. 27 Comparison of 3D models

average height exceeds 15m, where z is the height above the ground surface in meters. The building surfaces are assumed to a smooth walls. A commercial CFD solver (CD-adapco, 2018) has been used in the present study.

A RANS CFD simulation is performed with an arbitrary U_{10} (10 m/s) velocity at the inlet. U_{10} is a wind speed measured at a height of 10m above the ground. z_0 is the roughness length with a value of 0.05m. The simulation gives the building-induced velocity alteration factor $F_{s,B}$. This is taken as the maximum of the two ratios of the mean velocity and gust speed on the PLW plane at the downstream location of interest at the same elevation at the inlet. Where gust speeds are approximated from the mean wind speed and kinetic energy as follows $v_{gust} = v(1$

$+g\sqrt{2k/v}$), where v is the local mean wind velocity, k is the local kinetic energy, and g is the normalized peak factor, g = 3.5 assuming a normal distribution. The terrain-related alteration factor ($F_{s,T}$) can be found from $(1 z_g. 5)_{\alpha_s}$ where z_g is gradient height, and α_s the aerodynamic exponent for the terrain upstream of the domain inlet velocity at the location of interest based on gradient velocity becomes, $v_N = F_{s,B} F_{s,T} v_g$.

3.5.2 Computational domain

The 3D model generated in the first section of the thesis is used to set up the computational model. The study domain shown in Fig. 3.13 is defined using StarCCM, a commercially available CFD solver. The tallest building height H is used as a reference to design the domain based on (Tominaga et al., 2008). The computational domain is set following the standard guidelines described in Dagnew and Bitsuamlak (Dagnew and Bitsuamlak, 2013) where 5H upstream, 5H distance on the sides and 15H downstream and 5H height above the tallest building (Franke et al., 2007). Neighbourhood scales use a modified computation domain to accommodate the increased domain due to consideration of the neighbourhood as discussed in (Oshima et al., 2014, van Druenen et al., 2019) to limit blockage issues.

3.6 Results and discussion

3.6.1 Building Footprint extraction result

To achieve an effective model, the network must be trained extensively. The training progress is monitored by using the validation data set as a control mechanism. The model can achieve higher accuracy through continuous training and saving the learned parameters and resuming training with additional information until there’s no change in the loss value. For example, the model showed improved results when data augmentation is applied. The model is able to identify building outlines with considerable accuracy as shown in Fig. 3.14. The buildings indicated as ground truth are the manually prepared masks to evaluate the accuracy of the model's prediction. In this manner building, footprints shown in Fig. 3.15 are produced.

Model validation

The percentage of correctly identified pixels of each class is indicated by accuracy. This is important to know how well the building class pixels are identified. For building class, the accuracy is the ratio of correctly classified pixels to the total number of actual pixels based on ground truth as shown in Equation 3.6.

$$Accuracy\ score = TP/(TP + FN) \qquad \text{Eq.3.6}$$

The accuracy score is a ratio of the number of true positives (TP) that are correctly classified pixels to the total number of pixels ($TP + FN$), where FN is the number of false negatives.

The segmentation model is tested on selected buildings in London, Ontario (Fig. 3.16).

$$metrics = evaluateSemanticSegmentation(pxdsResults, pxdsTruth); \qquad \text{Eq.3.7}$$

Equation 3.7 computes various metrics to evaluate the quality of the semantic segmentation results which are predicted pixel labels (*pxdsResults*) against the ground truth segmentation which are ground truth pixel labels (*pxdsTruth*). The '*evalauteSemanticSegmentation*' is a MATLAB function that computes all available metrics, including the confusion matrix, normalized confusion matrix, data set metrics, class metrics, and image metrics.

In MATLAB, the performance of a semantic segmentation algorithm can be evaluated using the "*evaluateSemanticSegmentation*" function. This function compares the predicted labels from the deep learning algorithm with the ground truth labels for a set of images, and computes several metrics to evaluate the accuracy of the algorithm. Here's an example of how to use the function:

```
% Load the ground truth labels and predicted labels
groundTruth = imageDatastore('path/to/ground/truth/labels');
predictedLabels = imageDatastore('path/to/predicted/labels');
% Create a metrics object and evaluate the performance
metrics = evaluateSemanticSegmentation(predictedLabels, groundTruth);
% Display the results
disp(metrics)
```

The "*imageDatastor*e" function is used to create a datastore for the ground truth labels and predicted labels. This function loads the image files and organizes them into a format that can be easily processed by MATLAB. The "*evaluateSemanticSegmentation*" function takes these datastores as input, and computes several metrics, including the overall accuracy and mean intersection over union (IoU). These metrics can be accessed using the fields of the output "*metrics*" object. The ground truth labels and predicted labels must have the same size and format. The ground truth labels should be grayscale images, where each pixel is labeled with an integer value representing the class label. The predicted labels should have the same format as the ground truth labels, with each pixel labeled with an integer value representing the predicted class label.

Global accuracy calculates the percentage of correctly classified pixels over all pixels in the image. It is the most basic metric and provides an overall measure of the model's performance.

Mean accuracy calculates the average percentage of correctly classified pixels across all classes. Mean Intersection over Union (IoU) measures the degree of overlap between the predicted and ground truth segmentation masks. Weighted IoU is calculated as a weighted average of the IoU scores for each class, where the weight is the proportion of pixels belonging to that class in the ground truth labels. Mean BF score is a combination of precision and recall, and it measures the balance between them. It is calculated as the harmonic mean of precision and recall, where precision is the fraction of true positives among all predicted positives, and recall is the fraction of true positives among all ground truth positives. BF score takes into account both false positives and false negatives. Overall the global accuracy is used for evaluating the overall performance of the deep learning model accuracy since the class distribution in the data is balanced. Therefore the overall accuracy of the model is 98%.

$$pxdsResults = semanticseg\ imds, net,{}^{''}WriteLocation^{''},{}'\ source\ path'\ ; \qquad \text{Eq.3.8}$$

The ‘*semanticseg’* in Equation 3.8 is a function that returns a semantic segmentation of the input images using a deep learning model, in this case, represented by a variable ‘*net’*. The ‘*imds’* is a variable that represents the collection of building image data to be classified. The ‘*WriteLocation’* is the path where the predicted images will be stored while *‘source_path’* is the location where the images to be predicted are located.

$$pxdsTruth = pixelLabelDatastore(testLabelsDir, classNames, labelIDs); \qquad \text{Eq.3.9}$$

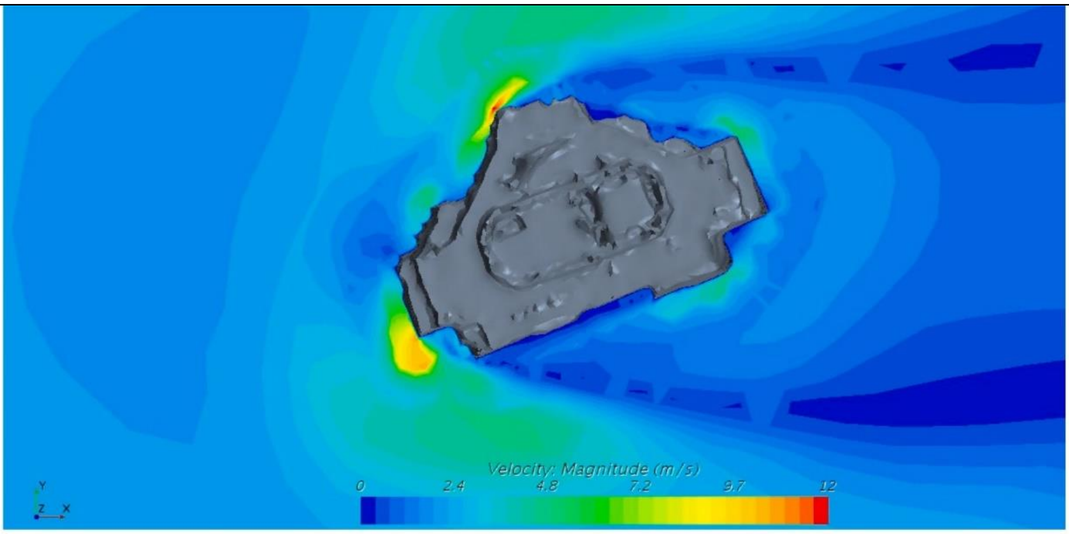
The ‘*pixelLabelDatastore’* function creates datastore for ground truth pixel labels (ground truth labelled images). The ‘*testLabelsDir’* variable represents image files. The ‘*classNames’* represents the building and background classes. The ‘*labelIDs’* represent identification numbers to relate pixel labels to class names.

In Table 3.2, the global accuracy is the ratio of correctly classified pixels, all classes, to the total number of pixels. The mean accuracy is the average accuracy of all classes in an image. The intersection over union (IoU) is the ratio of correctly classified pixels to the total number of ground truth and predicted pixels in the building class. i.e. *IoU score TP/(TP FP FN).* The mean BF score indicates how well the predicted boundary of each building class aligns with the true boundary of the building.

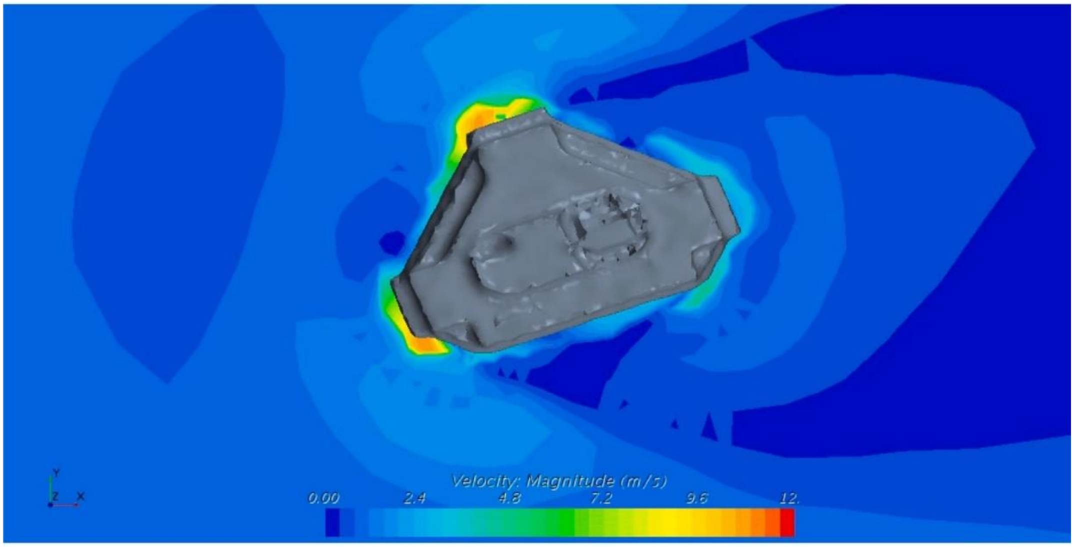
3.6.2 3D building model generation result

The 3D model generation workflow steps, where an input image goes through transformations before it is converted into a three-dimensional object are shown in Fig. 3.16. The 3D models and mesh for the downtown core of London, Ontario generated for CFD simulation are shown in Figs. 3.17 and 3.18, respectively. The grid resolution of the 3D building models is 1.5m.

Overall, the computational domain has 2.2 million polyhedrane cells and it was generated following the guideline (Franke et al., 2007). The mesh of the buildings is reduced to H/50 to capture important details of flow based on (Adamek et al., 2017) where H is the height of the tallest building.



Velocity contour from the predicted model



Velocity contour from the actual model

Figure 3. 28 Comparison of velocity contour generated using predicted and actual models.

3.6.3 CFD modelling results

The CFD simulation in this thesis is performed by the relatively computational cost-effective RANS approach for PLW applications and conclusions are made based on various pedestrian wind comfort criteria from Table 3.3 which shows wind speed in Guest equivalent mean (GEM). GEM is defined as the maximum mean wind speed or gust speed divided by 1.85 (whichever is larger). Since steady simulation is conducted in the present study, the gust speed can be approximated from the mean wind speed and kinetic energy as follows $v_{gust} = v(1 + g \sqrt[2]{2k/v}$, where v is the local mean wind velocity, k is the local kinetic energy, and g is the normalized peak factor, g 3.5 assuming a normal distribution (as discussed in Section 3.5.1). The pedestrian wind speeds are calculated at 1.5m in height from the ground. The velocity contours for eight wind directions are shown in Figs. 3.20–3.22, for U_{10} of 10 m/s. The turbulence kinetic energy contour plots are shown in Figs. 3.23–3.26. Wind speeds observed between the building spaces are higher than in the surrounding area. Wind speeds observed around some buildings may be a reason for concern for pedestrians using these streets. Some of the wind speeds observed on the buildings' sides may result in some degree of discomfort for pedestrians as shown in Table 3.3.

3.6.4 Validation

For the tallest building, a comparison analysis is done between the actual model and the predicted model (see Fig. 3.27). The comparison is done by carrying out a pedestrian-level wind velocity assessment in a CFD setting. The actual model is done by simulating a flight path of a drone on a computer. The simulated drone captured video footage of the building's google earth 3D representation. The video is then used as input for a software application to convert the video into a set of discrete images. Then those images are used to generate a 3D model of the building. Due to the complex number of steps and the large file size that demands huge computational power, the drone flight simulation method can only be used for validation purposes.

Overall, the predicted and actual models have similar CFD results in terms of wind velocity magnitude anticipated for a pedestrian-level comfort study (see Fig. 3.28). The simulation is carried out for wind in the x- direction (i.e., from left to right). In both models, increased wind speeds were observed on the east side of the buildings due to venturi and downwash effects. The north side of the predicted model misses a façade on the edge and that resulted in lower wind speed observation, in the middle of the north surface. But in reality, there is a slightly higher wind speed on the northern edge of the building as shown by the actual model. On the east side of the predicted model, a façade is also missing which resulted in a different wind speed than the actual. All these shortcomings are expected to improve when the good quality of images, LiDAR data and grid resolution are coupled with the use of a high-end advanced research computing facility following the workflow developed in this study.

Conclusion

A new framework is developed that integrates the autonomous urban topology generator with urban flow modelling. A new deep learning model for building footprints extraction from satellite imagery is developed and used to generate 3D building models by integrating it with Light Detection and Ranging (LiDAR) data to generate 3D building models. The 3D models are meshed and used in the CFD modelling modules. The entire process is explained through an application example of urban flow modelling encountered during a pedestrian-level wind assessment.

The U-Net-based deep learning algorithm achieved satisfactory accuracy. The model has also achieved a 3D building model with a LOD3 value. The advantage of the deep learning model is that it can be applied to a new set of input data to extract building footprints for autonomous applications. In addition, the model can be improved over time with minimum adjustments when more quality data is available. The deep learning method is promising because the model keeps improving over time when more data is available for training.

The image segmentation part of the building 3D modelling workflow is critical to the autonomous concept since the deep learning model can be reused to predict a new set of images. The model can be adapted to image samples that are significantly different from the images on which it was trained with minimum training samples. The combination uses of images and point cloud data addressed the issues of roof structure modelling and most of the building facades are also captured with the workflow used. Since the buildings are classified one at a time, it would be more convenient if the collected testing images are georeferenced beforehand to make the workflow quicker. The resolution of the LiDAR data should always be of higher quality, as lower quality will cause some structures to not be complete. Especially, the roof structures are more affected by lower-quality point cloud resolution. The predicted building outline polygons can further be processed to achieve more refined edges. But this will be limited to the type of application that these polygons will be used as in some cases the extra processing seems to chop off critical details from the polygons.

Although the model can accurately identify different building shapes, it still requires improvement to predict unique building shapes that are uncommon. The model needs additional training to make a more accurate prediction of buildings which have various extra details placed on them. Also, when the number of buildings being predicted at a single moment increases, the model struggles to capture more details. This is mainly due to the limitation of the training set and computational requirements. But the workflow described in this thesis can be considered as a starting point for accurate and improved building model predictions at a city scale when quality training set is provided along with sufficient computational resources. The accuracy of the final 3D model outcome of the buildings is also affected by the density of the LiDAR data. The absence of dense point cloud data has affected some details of the building models generated in this thesis.

The autonomously generated 3D models are seamlessly integrated with the physics modelling to simulate turbulence flow in urban areas that are used among other applications for pedestrian-level wind assessment, thus, reducing the Engineering and Tech time-intensive manual computational model generation process. Similar approaches can be used for other urban flow and heat transfer modelling.

References

Sola, A., Corchero, C., Salom, J., & Sanmarti, M. (2020). Multi-domain urban-scale energy modelling tools: A review. *Sustain. Cities Soc., 54*(February 2019).

Shooshtarian, S., Lam, C. K. C., & Kenawy, I. (2020). Outdoor thermal comfort assessment: A review on thermal comfort research in Australia. *Build. Environ., 177* (January), Article 106917.

Liu, S., et al. (2018). Influence of surrounding buildings on wind flow around a building predicted by CFD simulations. *Build. Environ., 140*(May), 1–10.

Tominaga, Y., et al. (2008). AIJ guidelines for practical applications of CFD to pedestrian wind environment around buildings. *J. Wind Eng. Ind. Aerodyn., 96*(10–11), 1749–1761.

Liu, S., Pan, W., Zhang, H., Cheng, X., Long, Z., & Chen, Q. (2017). CFD simulations of wind distribution in an urban community with a full-scale geometrical model. *Build. Environ., 117*, 11–23.

Zhai, Z. (2006). Application of computational fluid dynamics in building design: Aspects and trends. *Indoor Built Environ, 15*(4), 305–313.

Lach, S. R., Brown, S. D., & Kerekes, J. P. (2006). Semi-automated DIRSIG scene modelling from 3D LIDAR and passive imaging sources. *Laser Radar Technol. Appl. XI, 6214*(May), 62140I.

Isyumov, N. (2012). Alan G. Davenport’s mark on wind engineering. *J. Wind Eng. Ind. Aerodyn., 104–106*, 12–24.

Park, Y., & Guldmann, J. (2019). Computers, Environment and Urban Systems Creating 3D city models with building footprints and LIDAR point cloud classification : A machine learning approach. *Comput. Environ. Urban Syst., 75*(November 2018), 76–89.

Li, M., Rottensteiner, F., & Heipke, C. (2019). Modelling of buildings from aerial LiDAR point clouds using TINs and label maps. *ISPRS J. Photogramm. Remote Sens., 154* (June), 127–138.

Susaki, J. (2012). Adaptive slope filtering of airborne lidar data in urban areas for Digital Terrain Model (DTM) generation. *Remote Sens., 4*(6), 1804–1819.

Richner, R. P. (2011). Research Collection. *Brisk Bin. Robust Invariant Scalable Keypoints*, 12–19.

Chen, L. Y., Teo, T., & Rau, J. (2004). Fusion of lidar data and optical imagery for building modelling. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch., 35*(January).

Holmgren, J., Nilsson, M., & Olsson, H. (2003). Simulating the effects of lidar scanning angle for estimation of mean tree height and canopy closure. *Can. J. Remote Sens., 29* (5), 623–632.

Minato, A., Kobayashi, T., & Sugimoto, N. (1998). Laser long-path absorption lidar technique for measuring methane using gas correlation method. *Japanese J. Appl. Physics, Part 1 Regul. Pap. Short Notes Rev. Pap., 37*(6 A), 3610–3613.

Gao, F., Veberiˇc, D., Staniˇc, S., Bergant, K., & Hua, D. X. (2013). Performance improvement of long-range scanning Mie lidar for the retrieval of atmospheric extinction. *J. Quant. Spectrosc. Radiat. Transf., 122*, 72–78.

Morsy, S., Shaker, A., & El-Rabbany, A. (2017). Multispectral lidar data for land cover classification of urban areas. *Sensors (Switzerland), 17*(5).

D. Lee, H. Jung, J. Yom, S. Lim, and J. Kim, “Automatic Generation of Building Footprints From Airborne Lidar Data,” vol. 44, no. 9, pp. 2523–2533, 2006.

Weng, W., & Zhu, X. (2021). INet: Convolutional Networks for Biomedical Image Segmentation. *IEEE Access, 9*, 16591–16603.

He, K., Gkioxari, G., Dolla´r, P., & Girshick, R. (2020). Mask R-CNN. *IEEE Trans. Pattern*

Anal. Mach. Intell., 42(2), 386–397.

Wang, C., Cheng, M., Sohel, F., Bennamoun, M., & Li, J. (2019). NormalNet: A voxel- based CNN for 3D object classification and retrieval. *Neurocomputing, 323*, 139–147.

Telkamp, G. J. (1981). Note to users. *Itinerario, 5*, 68–69.

Oshima, T., Hiraguri, Y., & Imano, M. (2014). Geometry reconstruction and mesh generation techniques for acoustic simulations over real-life urban areas using digital geographic information. *Acoust. Sci. Technol., 35*(2), 108–118.

Sowmya, A., & Trinder, J. (2000). Modelling and representation issues in automated feature extraction from aerial and satellite images. *ISPRS J. Photogramm. Remote Sens., 55*(1), 34–47.

Kwak, E., Al-Durgham, M., & Habib, A. (2012). Automatic 3D Building Model Generation From Lidar and Image Data Using Sequential Minimum Bounding Rectangle. *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci., XXXIX-B3*(September), 285–290.

Adamek, K., Vasan, N., Elshaer, A., English, E., & Bitsuamlak, G. (2017). Pedestrian level wind assessment through city development: A study of the financial district in Toronto. *Sustain. Cities Soc., 35*(July), 178–190.

van Druenen, T., van Hooff, T., Montazeri, H., & Blocken, B. (2019). CFD evaluation of building geometry modifications to reduce pedestrian-level wind speed. *Build.*

Environ., 163(July), Article 106293.

Phillips, D., Michael Soligo, R., Phillips, D. A., & Soligo, M. J. (2019). Title: Will CFD ever Replace Wind Tunnels forBuilding Wind Simulations? High-Rise Buildings Will CFD ever Replace Wind Tunnels for Building Wind Simulations? *Int. J. High-Rise Build., 8* (2), 107–116.

van Druenen, T., van Hooff, T., Montazeri, H., & Blocken, B. (2019). CFD evaluation of building geometry modifications to reduce pedestrian-level wind speed. *Build.*

Environ., 163(April), Article 106293.

Janssen, W. D., Blocken, B., & van Hooff, T. (2013). Pedestrian wind comfort around buildings: Comparison of wind comfort criteria based on whole-flow field data for a complex case study. *Build. Environ., 59*, 547–562.

Tong, H., Walton, A., Sang, J., & Chan, J. C. L. (2005). Numerical simulation of the urban boundary layer over the complex terrain of Hong Kong. *Atmos. Environ., 39*(19), 3549–3563.

Tominaga, Y., & Stathopoulos, T. (2010). Numerical simulation of dispersion around an isolated cubic building: Model evaluation of RANS and LES. *Build. Environ., 45*(10), 2231–2239.

B. Blocken, *LES over RANS in building simulation for outdoor and indoor applications: A foregone conclusion?*, vol. 11, no. 5. 2018.

Bitsuamlak, E., Girma, & Simiu. (2010). CFD ’ s potential applications : a wind engineering perspective. In *Proc. Fifth Int. Symp. Comput. Wind Eng. Conf. Chapel Hill, NC* (pp. 23–27). no. 2008.

Su, W., Zhang, Y., Yang, Y., & Ye, G. (2014). Examining the impact of greenspace patterns on land surface temperature by coupling LiDAR data with a CFD model. *Sustain*, *6*(10), 6799–6814.

Kanan, C., & Cottrell, G. W. (2012). Color-to-grayscale: Does the method matter in image recognition? *PLoS One*, *7*(1).

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd Int.*

Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc (pp. 1–15). CD-adapco, S.-C. (2018). STAR-CCM v 11.06.011, 2018. *User Man*.

Richards, P. J., & Hoxey, R. P. (1993). Appropriate boundary conditions for computational wind engineering models using the k-ϵ turbulence model. *J. Wind Eng. Ind. Aerodyn.*, *46–47*(C), 145–153. Aug.

Richards, P. J., & Norris, S. E. (2011). Appropriate boundary conditions for computational wind engineering models revisited. *J. Wind Eng. Ind. Aerodyn.*, *99*(4), 257–266.

Dagnew, A. K., & Bitsuamlak, G. T. (2013). Computational evaluation of wind loads on buildings: A review. *Wind Struct. An Int. J.*, *16*(6), 629–660.

Jorg Franke, Antti Hellsten, Heinke Schlunzen, Bertrand Carissimo, Cost: *Best Practice Guideline For The CFD Simulation Of Flows In The Urban Environment*, 2007.

Chapter 4

Simulating wind flow over complex terrain created by using LiDAR and SRTM

Abstract

Design wind loads on structures and wind power generation depend, among other factors, on the velocity profile and turbulence characteristics of the upcoming wind. These, in turn, depend on the roughness and general configuration of the upstream topography. It has been reported that wind speeds could double in hilly areas at 10 m height above ground. A few national (such as US, Canada, Australia/New Zealand's (AS/NZ)) codes take topography complexities into account to some extent. These codes of practice typically assume simplified upstream topography conditions of a homogeneous roughness or provide explicit corrections in the form of speed-up only for a limited number of specific topographies such as a single hill, valley, or escarpment. In this study accurate CFD models by using high resolution LiDAR data and publicly available Shuttle Radar Topography Mission (SRTM) in Geographic Tagged Image File Format (GeoTIFF) data is developed. Impact of geometric modelling accuracy is discussed. Speed-up factors for cases that are not covered in building codes and standards, such as complex terrain, are generated via a Computational Fluid Dynamics (CFD).

Keywords: Wind speed-up, CFD, LiDAR, SRTM, complex terrain

4.1 Introduction

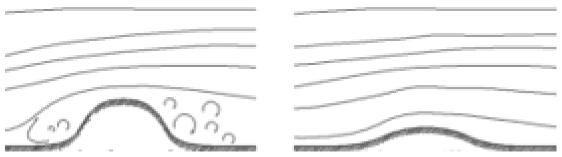
Modelling wind flow over complex terrain is an integral part of wind energy resource assessment and wind load evaluation for structures. The modifications in wind flow due to topography are usually reported in a factor called “speed-up”, which represents the relative increase/decrease in wind speed (mean) in comparison with the incoming wind speed (mean) that is not affected by the terrain, measured at a similar height from the ground. There are several parameters that affect speed-up including geometric parameters such as slope, height, distance from the crest (or bottom), three or two dimensionality, number of hills/valleys; and ground roughness usually presented as roughness length (z_0). Wind flows over hills, for example, are significantly accelerated even when the maximum slopes are quite small as shear in the approaching wind amplifies the wind speed (Jackson and Hunt, 1975). For example, a 20% hill slope could result in a 1.5 speed-up (Belcher and Hunt, 1998). The other parameter that needs to be looked at is the turbulence, which is significantly affected by the terrain, especially in flow separation zones such as at the edges of the mountains and in the wake regions. (Wood, 1995) reported that for two-dimensional hills, 0.31 can be assumed to be the critical slope for flow separation and 0.63 in the three-dimensional case. Such changes in the mean wind speed (i.e. speed-up) and turbulence characteristics must be quantified as accurately as possible for design wind loading calculations in order to design the most economic and safe structures. There have been efforts by several researchers to study the wind flow over hills and complex terrain using theoretical models, wind tunnel studies and more recently on computational fluid dynamics (CFD) models.

Theoretical Models: (Jackson and Hunt, 1975) presented an analytical solution for the flow of a turbulent boundary layer on a uniformly rough surface over a two-dimensional hump with low curvature, e.g. a low hill. (Mason and Sykes, 1979) expanded the two-dimensional theory of (Jackson and Hunt, 1975) for turbulent flow over a shallow ridge to three-dimensional topography. (Walmsley et al., 1980) presented MS3DJH - a computer model for the study of neutrally stratified boundary-layer flow over isolated hills of moderate slope. (Walmsley et al., 1982) studied the effects of small-scale topographic features on their theoretical model prediction. (Taylor et al., 1983) developed the MS3DJH to allow for the use of terrain-dependent length and velocity scales and solutions for the velocity perturbation field. (Taylor et al., 1987) developed guidelines based on linear models which have been adopted by the AS/NZ code. These guidelines were modified by (Weng et al., 2000) to incorporate the effects of surface roughness and non-linearity on speed-up. (Taylor et al., 1986) presented a review of boundary-layer flow over low hills in an attempt to summarize some of the experiments that have been conducted over such terrain. (Lemelin et al., 1988) presented simple empirical approximations for speed-up over hills by using the computer models presented by (Taylor et al., 1986) and augmenting them with wind tunnel data for steep slope cases. These established empirical formulae were the basis of the provisions of the National Building Code of Canada (NBCC, 1995). It should be noted that these provisions have also been used by the American Wind Loading Standard (ASCE 7 – 2002) in all its recent editions (ASCE 7- 2022). In a review of the mechanisms that control neutrally stable turbulent boundary-layer flow over hills and waves, (Belcher and Hunt, 1998) compared calculations based on various analytical and computational models with each other and with relevant experimental data. (Pinard and Wilson, 1999) presented a computer model for wind flow over mesoscale mountainous terrain applied to the yukon. While Microscale models are steady state and are built for small domain sizes, i.e. less than 10 km, mesoscale models are fully time-dependent and are for domains of the order 10 to 2000 km.

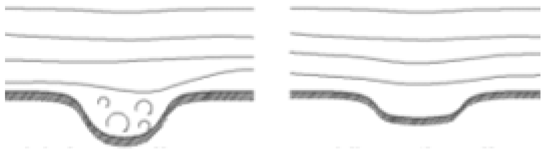
Building codes and standards: Wind standards and codes of practice (e.g., ASCE for the US and NBC of Canada) provide very useful explicit corrections in the form of speed-up for only a limited number of specific topographies, such as single hills, valleys or escarpments. There are no provisions for complex terrain modifications or speed-up factors. For more complex situations, the practitioner is referred to physical simulations in a BLWT. These codes do not include topographically generated features such as: (a) funneling effects in valleys or in between hills, (b) corner effects along the foot of mountains and hills, (c) vortex formation behind steep terrain, and (d) other effects such as wind speed-up above short buildings and wind speed-up between tall buildings.

Wind tunnel and field measurement: (Miller and Davenport, 1998) provided guidelines for the wind speed-up evaluation over complex two dimensional surfaces based on a wind tunnel study. (Ishihara, 1999) presented the results of measurements of wind speed over a circular hill with a maximum slope of about 62.5%. (Cao and Tamura, 2007) studied the roughness blocks effect on the atmospheric boundary layer flow over a two dimensional low hill with and without a sudden roughness change. The effects of the roughness blocks were clarified by comparing the flow characteristics over hill models, with emphasis on wind speed-up and turbulence structure. Adding or removing roughness blocks on the hill surface or inflow area changes the velocity deficit and creates a completely different turbulence structure in the wake. (Lubitz and White, 2007) presented a wind tunnel and field investigation of the effect of local wind direction on speed-up over hills. Other wind tunnel investigations include: (Armitt et al., 1975), (Arya et al., 1987), (Finnigan et al., 1990), (Gong and Ibbetson, 1989), (Snyder and Britter, 1987), (Ferreira et al., 1995), (Carpenter, 1999), (Athanassiadou and Castro, 2001), (Ayotte and Hughes, 2004). Some of the field experiments include (Coppin et al., 1994) and (Castro et al., 2003). (Savory et al., 2008) presented a comparison between field results and a design code for wind-induced transmission tower foundation loads.

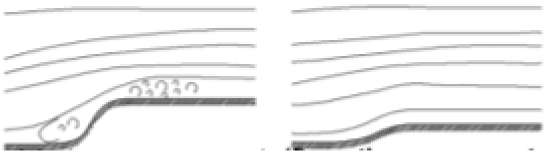
Computational fluid dynamics: Recently, numerical models based on (CFD) principles have been used to evaluate speed-ups for those cases that are not covered by wind design standards and codes of practice. (Bitsuamlak et al., 2004) performed a comparative study of NBCC code, analytical, numerical and experimental (both wind tunnel and field approaches). Analytical models based on linear models were agreeing well with experimental data for shallow hills but deviate for steep hills. Numerical work based on k-epsilon establishing the modifications of wind flow over isolated hills, escarpments, valleys and complex terrains showed good agreement on upstream side, but on the leeward side may lead to problematic predictions (Bitsuamlak et al., 2004). (Bitsuamlak et al., 2006) adopted a CFD based method to evaluate speed-up for various hill and valley, and multiple hill configurations including ground roughness effects, their results comparing well with boundary layer wind tunnel (BLWT) data. (Bitsuamlak et al., 2007) then presented a combined numerical–neural network (NN) approach to provide speed-up ratios for a wide range of topographic features such as single and multiple hills, escarpments, and valleys. The combined approach was able to produce speed-up values were consistent with detail CFD model results, but responsive to simple geometrical inputs and roughness length provided by the user. These studies were limited to evaluating the mean wind speed characteristics (i.e. speed-up). In the proposed study turbulence characteristics will be addressed. A workflow for numerically assessing wind flow over a complex terrain will be developed and the effect of the local geometry modelled both through high resolution LiDAR measurements and publicly available GoeTiff data will be investigated in this study.



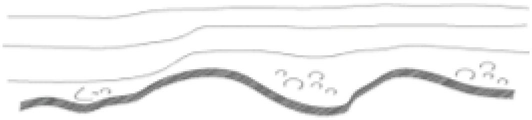
(a) steep hill (b) gentle hill



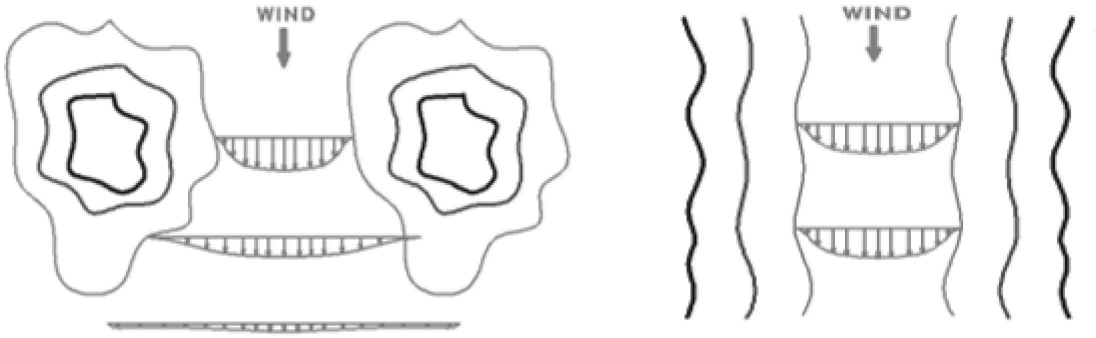
(c) deep valley (d) shallow valley



(e) steep escarpment (f) gentle escarpment



(g) complex terrain, multiple hills



(h) funneling between hills

(i) funneling inside a valley

Figure 4. 1 Wind flow over simple and complex terrains

4.2 Wind speed-up factors

The National Building Code of Canada (NBCC, 2005) defines the speed-up factors in terms of speed-up ratio ΔS as follows:

$$\boldsymbol{\Delta S} = \frac{\boldsymbol{V(z)} - \boldsymbol{V_0(z)}}{\boldsymbol{V_0(z)}} \qquad \text{Eq.4.1}$$

where $V(z)$ is the velocity at height z above the local hill surface and $V_0(z)$ is the upstream reference velocity at same height z as explained in (Fig. 4.2). The following speed-up expression is given in the (NBCC, 2005)

$$\Delta S = \Delta S_{max}(1\text{-}\frac{/x/}{kL})e^{(-\frac{\alpha z}{L})} \qquad \text{Eq.4.2}$$

in which ΔS_{max} is the maximum speed-up ratio (at the crest near the surface), x is the horizontal distance between the position of the consideration and the crest of the hill as shown in (Fig.4.2), k is a constant given in (Table 4.1). L is the horizontal distance upwind between the crest of the hill and a position where the ground elevation is half the height of the hill (H), α is a decay coefficient representing the decrease in the speed-up with height. The values of ΔS_{max} and α depend on the shape of the hill (escarpment) and steepness of the topography.

It is worthy to note that the (NBCC, 2005) has no provisions for complex terrain modifications or speed-up factors. While the National Building Code of Canada (NBCC) 2010 and NBCC 2020 provide guidance on a wide range of topographic and geographic conditions, it is true that there may be complex terrain situations that are not specifically addressed by the code. However, the codes do provide guidance on appropriate design approaches and considerations for these situations.

However, it is important to note that building codes and standards are not intended to cover every possible scenario, and site-specific conditions may require additional design considerations. Architects, engineers, and builders must use their professional judgment and consider site-specific factors when designing and constructing buildings in complex terrain. This may involve conducting site-specific assessments and simulations to ensure that buildings are designed to withstand the expected wind loads and other natural hazards associated with complex terrain. The code provisions are for simple terrain, e.g., simple hills, ridges or escarpments. For more complex situations, the practitioner is referred to physical simulations in a boundary-layer wind tunnel. The standard does not include wind speed effects in rapidly varying terrain where roughness characteristics change significantly over short distances in comparison to typical overhead transmission line spans. This code does not include topographically generated features such as: (1) funnelling effects in valleys or in between hills, (2) corner effects along the foot of mountains and hills, (3) vortex formation behind steep terrain, and (4) other effects, such as, wind speed-up above short buildings and wind speed-up between tall buildings.

Table 4. 1 Parameters for maximum speed-up over hills and escarpments (NBCC, 2005)

shape of hill or escarpment	$\Delta S_{\max}$ (*)	α	k	
			$x<0$	$x>0$
2-dimensional ridges (over valleys with negative H)	2.2 H/L	3	1.5	1.5
2-dimensional escarpments	1.3 H/L	2.5	1.5	4
3-dimensional axi-symmetrical hills	1.6 H/L	4	1.5	1.5

(*) For $H/L > 0.5$, assume that $H/L = 0.5$ and substitute $2H$ for L

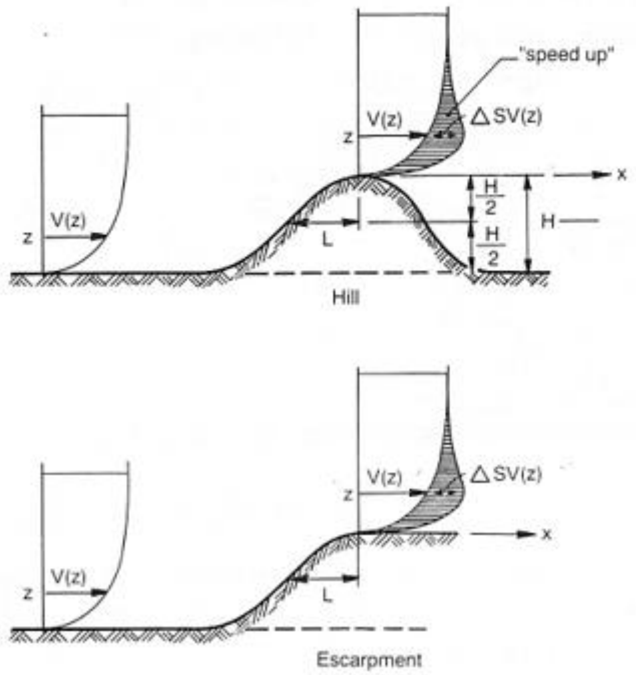


Figure 4. 2 Wind speed-up over hills and escarpments (NBCC, 2005)

The objective of this project has been to develop an application guideline for evaluating wind speed-up due to various types of topographic changes and their effect on transmission line and tower and provide application examples. The guideline is developed based on speed-up information assembled from (i) building codes and standards (including but not limited to ASCE 7-2010, NBCC 2005, AS/NZ, 2002), (ii) published peer reviewed journal papers, (iii) new CFD simulations that are carried out as part of the present study. The guideline includes detailed code application examples on the wind speed-up factors for transmission line and tower configurations specified by WISMIG participant utilities to provide an understanding of the correct usage of the equations, charts, and tables. The application has been developed to cover overarching problems in topography effects such: (a) funneling effects in valleys or in between hills, (b) corner effects along the foot of mountains and hills, (c) vortex formation behind steep terrain, and (d) other 3D effects.

LiDAR: Accuracy of geometric modelling is important to reasonably model the flow by using CFD. To this effect, utilities have started collecting their own 3D maps by deploying the optical remote-sensing method known as LiDAR (light detection and ranging) uses laser energy to intensively sample the earth's topography and generate precise *x*, *y*, and *z* measurements (Wichmann et al., 2015). LiDAR, which is largely utilized in aerial laser plotting applications, is starting to gain popularity as a more affordable option to more established surveying methods like photogrammetry (Curcio et al., 2022). Software like ArcGIS can be used to organize, view, analyze, and distribute large point cloud collections produced by LiDAR. A data collection machine (such as an aeroplane, drone, car, or tripod), a laser scanning system, and GPS (Global Positioning System), are among the main hardware elements of a LiDAR system (inertial navigation system). LiDAR inertial system measures the roll, pitch, and direction of the LiDAR system. LiDAR is an active visual beam that moves along predetermined study paths while transmitting laser rays to a target (Wang et al., 2021). The LiDAR devices pick up and evaluate the beam bouncing from an object. To determine the distance between the device and the intended object, these devices keep an accurate time log from when the beam pulse left the system until it returned (Wang et al., 2022). The measurements of the range are converted into readings of the real three-dimensional points of the reflected object when united with the spatial data GPS. After the point cloud data collecting survey, the LiDAR data is further processed into extremely precise georeferenced 3D coordinates by looking at the beam time distance, beam angle, and GPS location. Point cloud systems' beam pulses bounce off of target bodies on and above the ground, including plants, structures, and other things (Means et al., 2000). There could be one or more returns from a single beam pulse that can be detected by the LiDAR device (Torre-Tojal et al., 2022). Any produced beam pulse that goes to a surface and contacts various bouncing objects is scattered into as many returns as there are reflective surfaces. The most crucial return is the initial beam pulse, which will be connected to the highest point in the topography, such as the top of trees or buildings. Typically, for a ground surface, there will be one return registered by the LiDAR system. In addition to the *x, y,* and *z* spatial values, each point cloud contains intensity, a number of returns, return number, point classification values, GPS time, RGB (red, green and blue) values, scan direction and angle. Intensity is the return signal of the beam pulse that produced the point cloud (Song et al., 2002). The return number indicates the number of beam pulses reflected from various surfaces. It's common to have up to five returns received depending on the reflective surface and the capacity of the beam-scanning device to register such data. After the LiDAR data is post-processed, integer numbers are used to represent the point classification category of a specific point. The numbers can represent which class values like ground, vegetation, water and buildings the point belongs. If imagery data is collected during the point cloud survey, RGB data can be embedded in the point as an additional attribute.

The Shuttle Radar Topography Mission (SRTM) data is a type of satellite data that was collected by a radar instrument aboard the Space Shuttle Endeavour in February 2000. The SRTM data consists of digital elevation models (DEMs) that cover almost all of the Earth's land surface, making it a valuable resource for a wide range of applications, including cartography, geology, hydrology, and environmental monitoring. The SRTM data is available for download from the USGS EarthExplorer website in GeoTIFF format, which is a file format that can store both image data and geospatial metadata. The SRTM GeoTIFF files contain elevation data rather than traditional optical or multispectral satellite imagery, which makes it a valuable resource for generating digital elevation models or other topographic products.

GeoTIFF data: GeoTIFF (Geographic Tagged Image File Format) is an interchange format for extension of the TIFF(Tagged Image File Format) format, to support raster data georeferencing capability (Ritter and Ruth, 1997). The TIFF imagery file format is used to store and transfer digital satellite imagery, scanned aerial images, elevation models, and scanned maps. A TIFF file embedding geographic information (latitude, longitude, map projection etc.) is called GeoTIFF (Mahammad and Ramakrishnan, 2003). GeoTIFF is in wide use in NASA earth science data systems. The geographic content supported in the GeoTIFF tag structure includes its cartographic projection, datum, and ground pixel dimension. Raster data users can save time and effort by using GeoTIFF format due to its platform-interoperability. Data providers benefit from the ease of generating GeoTIFF format imagery products at lower cost.

The GeoTIFF in this research represented the SRTM data which contains elevation data. The United States Geological Survey (USGS) uses GeoTIFF data for satellite images. GeoTIFF is a file format that contains georeferencing information, which allows for the geographic positioning of the image data. This is important for satellite imagery, which is typically collected over large areas and needs to be precisely located in geographic space. The USGS maintains an extensive archive of satellite imagery in GeoTIFF format, which is used for a variety of purposes including land use and land cover mapping, environmental monitoring, and natural resource management. The GeoTIFF format is designed to store both image data and metadata that provides information about the geographic location and projection of the image. The metadata in a GeoTIFF file includes information such as the image's coordinate system, map projection, and spatial resolution.

4.3 Numerical simulation methodology

The following workflow is developed to simulate the wind flow over complex terrain as shown in Fig. 4.3. The overall workflow starts by collecting the necessary LiDAR and SRTM data. Then after creating the geometrical model that is used to define the computational model, proper boundary conditions are applied, and the Navier stokes equations are solved. Finally, the wind flow simulation results are analyzed to generate for example speed-up values. Details of the workflow are discussed in this section further.

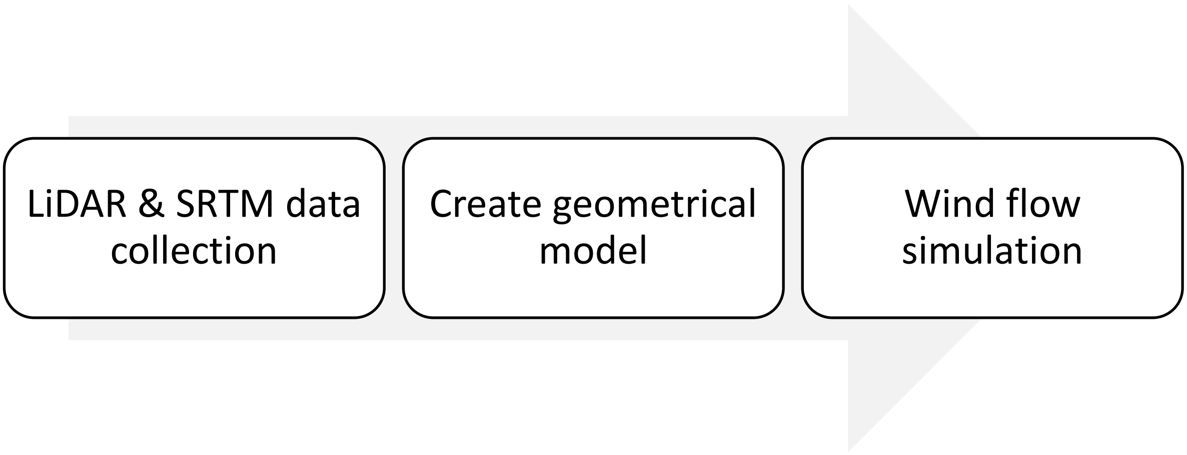


Figure 4. 3 The overall workflow

4.3.1 Topography CAD modelling

The topographic information for the numerical simulation was extracted from satellite imagery and Shuttle Radar Topography Mission (SRTM) elevation data at 30-meter resolution from SRTM and LiDAR. The CAD models are the same size around 50 square KM. The computational domain is divided into millions of polyhedral grids at which the flow equations are solved. Different grid refinement stages were used to maintain computational efficiency while attaining acceptable numerical accuracy. To reduce the computational cost associated with the modelling of such a large domain, different control volumes were used. Northing and easting values are used to represent a location on the Earth's surface (Figure 4.4). Northing and easting values are typically expressed in meters, but they can be converted to kilometers by dividing by 1000. The overall dimensions of the CD are 25 km streamwise, 15 km lateral, and 6 km vertical.

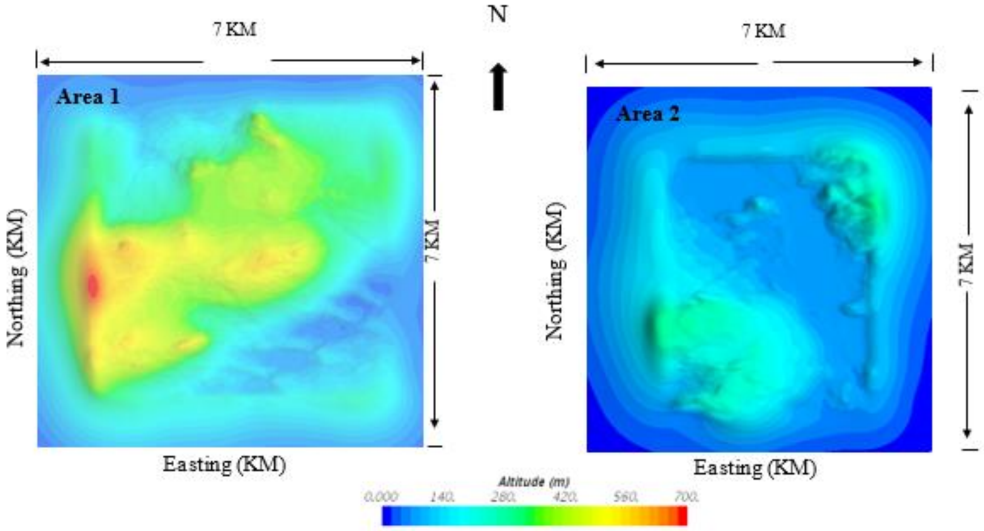


Figure 4. 4 Elevation contours used to generate the computational domain for the CFD simulation.

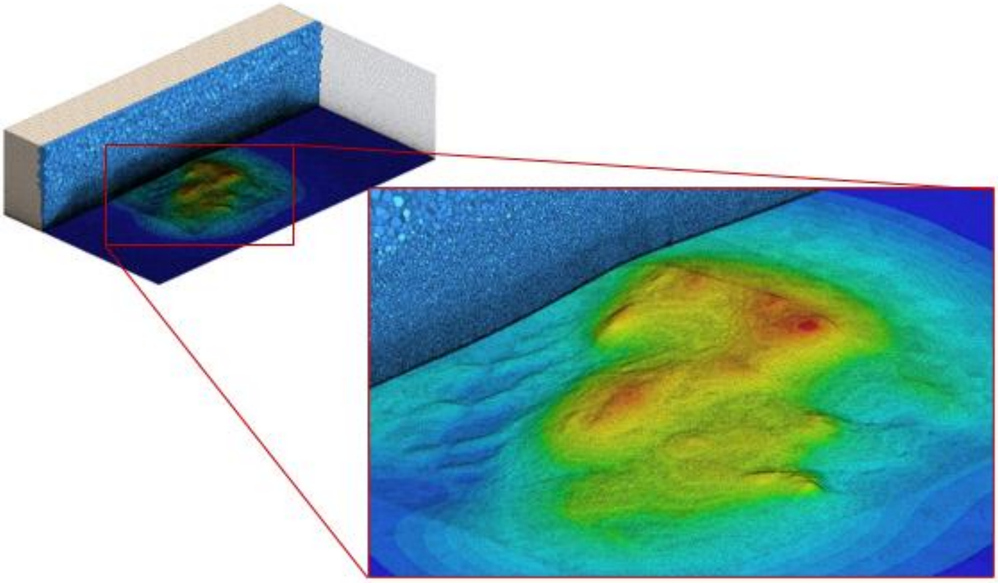


Figure 4. 5 Computational domain with high-resolution grid on the bottom surface of Area 1-LiDAR.

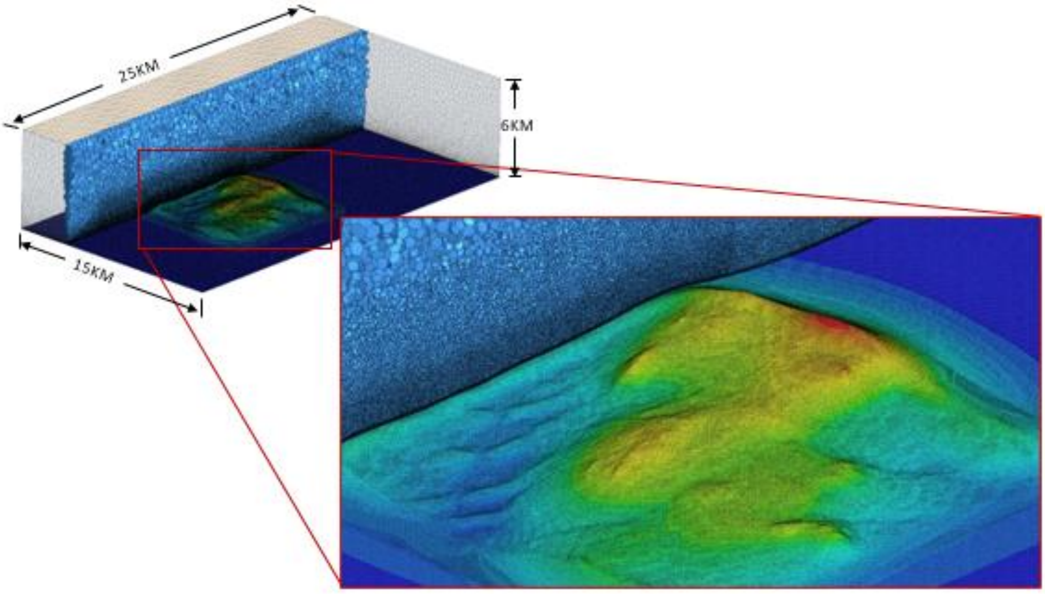


Figure 4. 6 Computational domain with high-resolution grid on the bottom surface of Area 1-SRTM.

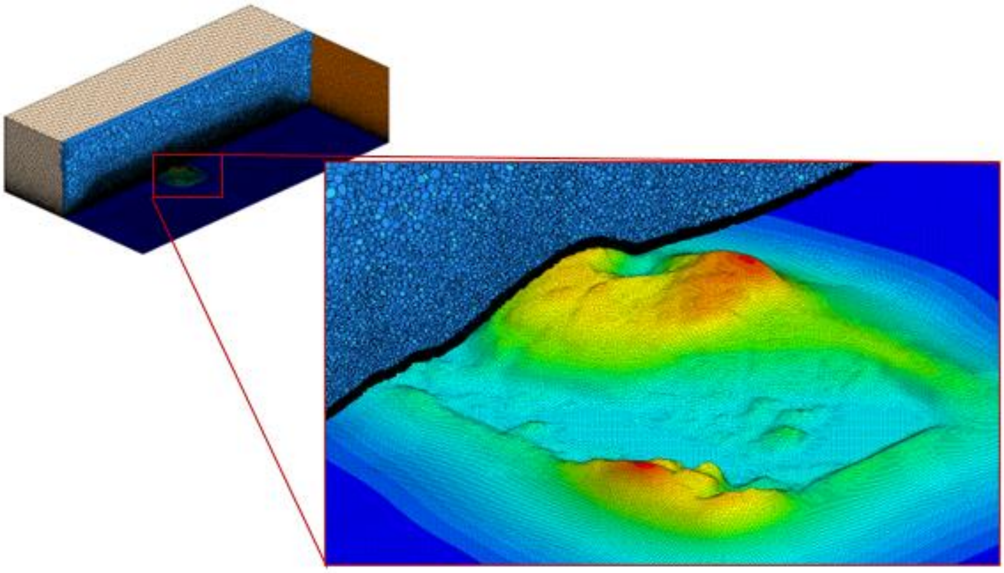


Figure 4. 7 Computational domain with high-resolution grid on the bottom surface of Area 2-LiDAR.

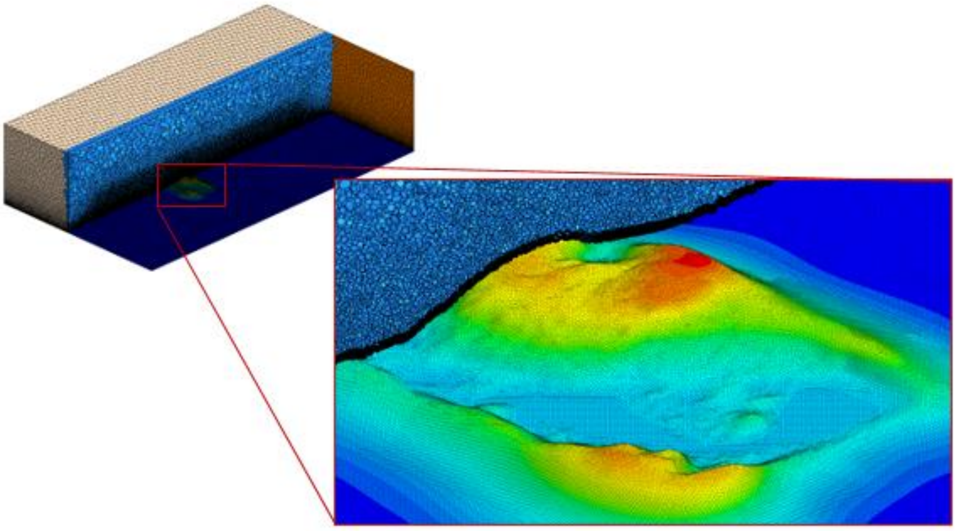


Figure 4. 8 Computational domain with high-resolution grid on the bottom surface of Area 2-SRTM.

4.3.2 Grid sensitivity analysis

The grid sensitivity analysis was performed considering three different mesh grids, G1 (12,639,907 cells), G2(19,339,060), and G3(24,739,056 cells) created for the East (90^0) wind direction. Around the target hill, fine grids were deployed with a resolution of 20 prism layers at a prism stretch of 1.3 and a total thickness set to 60m. The grid refinement ratio between G1 and G2 was 1.53 and between G2 and G3 was 1.25. The grid refinement values are higher than the minimum recommended values (Boache, 1994). A wind velocity magnitude at the high hill of the topography from the ground surface to a height of 10m was chosen as comparing parameter. The relative difference between G1 and G2 is 11.8% and the relative difference between G2 and G3 is 0.91%, thus, G3 is chosen as the final grid solution. In the present simulation, the grid resolution in the vertical direction is 2.5 m for the first 10 m and increases gradually with height above 10m till 60m. To reduce the computational cost, associated with modelling such as a large domain, the horizontal resolution is set to 10 m.

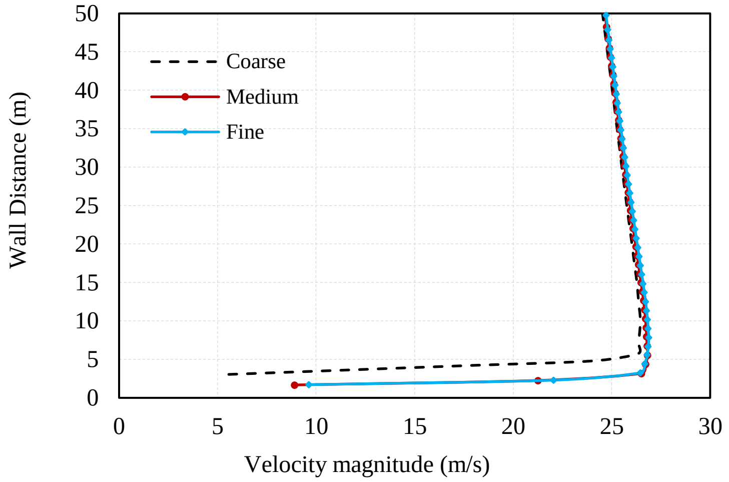


Figure 4. 9 Empty computational domain

4.3.3 Boundary conditions

In this study, the terrain upwind of the target terrain; along with the wind direction, is categorized as flat terrain. The atmospheric boundary layer velocity and turbulence profile based on the Engineering Sciences Data Unit (ESDU) is implemented for the oncoming wind at the inlet corresponding to the wind speed of 10 m/s at 10 m above ground. The two lateral sides and the top surfaces of the computational domain are assigned symmetry conditions. The outlet has zero static pressure boundary and the ground plane as a no-slip rough wall is assigned.

Atmospheric boundary layer (ABL) flow is imposed at the inlet of the domain where the velocity profile is described by the logarithmic law, which constitutes a vertical profile of the mean horizontal wind speed, turbulent kinetic energy K (m^2/s^2) and turbulence dissipation rate ε (m^2/s^3) (Richards and Norris, 2019) as shown the equations.

$$\boldsymbol{u(z) = \frac{u_*}{k} ln\left(\frac{z+z_0}{z_0}\right)} \qquad \text{Eq.4.3}$$

$$\boldsymbol{K = \frac{u_*^2}{\sqrt{C_\mu}}} \qquad \text{Eq.4.4}$$

$$\boldsymbol{\varepsilon} = \frac{\boldsymbol{u_*^3}}{\boldsymbol{k(z+z_0)}} \qquad \text{Eq.4.5}$$

where the constants k=0.42 and C_μ=0.09.

4.3.4 Solver setup

High-resolution, steady Reynolds-Averaged Navier-Stokes (RANS) governing equations with the *Shear Stress Transport* (*SST*) *k- ω* Low Reynolds Number Modelling (LRNM) (Menter, 1994) turbulence closure is deployed to resolve the near-ground airflow. *SST k-ω* turbulence model has been widely used for flow near-wall resolution, adverse pressure gradient, and large flow separations (Kahsay et al., 2019), (Jubayer and Hangan, 2018). 2nd order discretization scheme was chosen for the convection terms. The Gauss-Siedel relaxation scheme was chosen for pressure, velocity, and *k-ω* turbulence parameters. The simulation was carried out at a full-scale 1:1 ratio. The simulations were run using StarCCM+ v13.06.012 commercial package.

4.3.5 Assessing ABL homogeneity in an empty Computational Domain (CD)

This study involves a simulation of an empty CD in order to quantify the extent of ABL homogeneity at the inlet and near the study area where a velocity magnitude and turbulence intensity profiles computed between inlet, 5KM, 10KM, and 15KM as shown in Fig. 4.10.

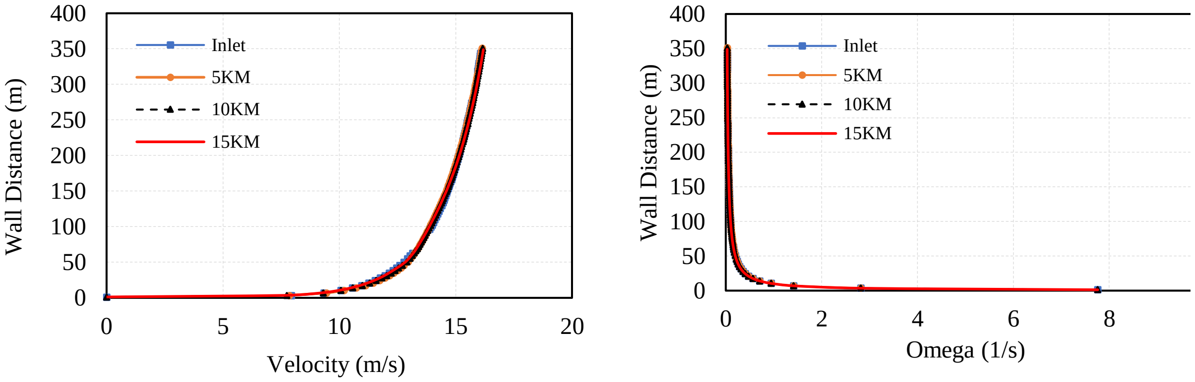


Figure 4. 10 Empty computational domain

4.4 Results and discussion

The speed-up ratio is defined as $(U_t(z)- U_0(z))/U_0(z))$ where $U_t(z)$ is the velocity at the topography at z height above the local ground and $U_0(z)$ is the velocity at the inlet (open profile) at z height above the local ground. Different simulations were performed to investigate the effect of a digital resolution of a CAD model extracted from LiDAR and SRTM. Two wind directions across (0^0) Northly wind and along (90^0) Easterly wind are chosen as comparison wind directions as shown in Fig. 4.11 and 4.15. The wind velocity magnitude at 10 m from the ground at the selected target area was chosen as a parameter for the comparison of the speedup and turbulent intensity as the wind speed at 10 m is widely used for designing structures.

Fig. 4.11 and 4.12 show the velocity contour for the selected Area 1 for a wind direction of 0^0. In these contours the LiDAR image due to its high resolution to capture the high hills detail, higher velocities are observed in the target place, and down the hill of the target area, a large cavity wake is observed.

The selected hill location is presented in topographic speedup contour as shown in Fig. 4.13.

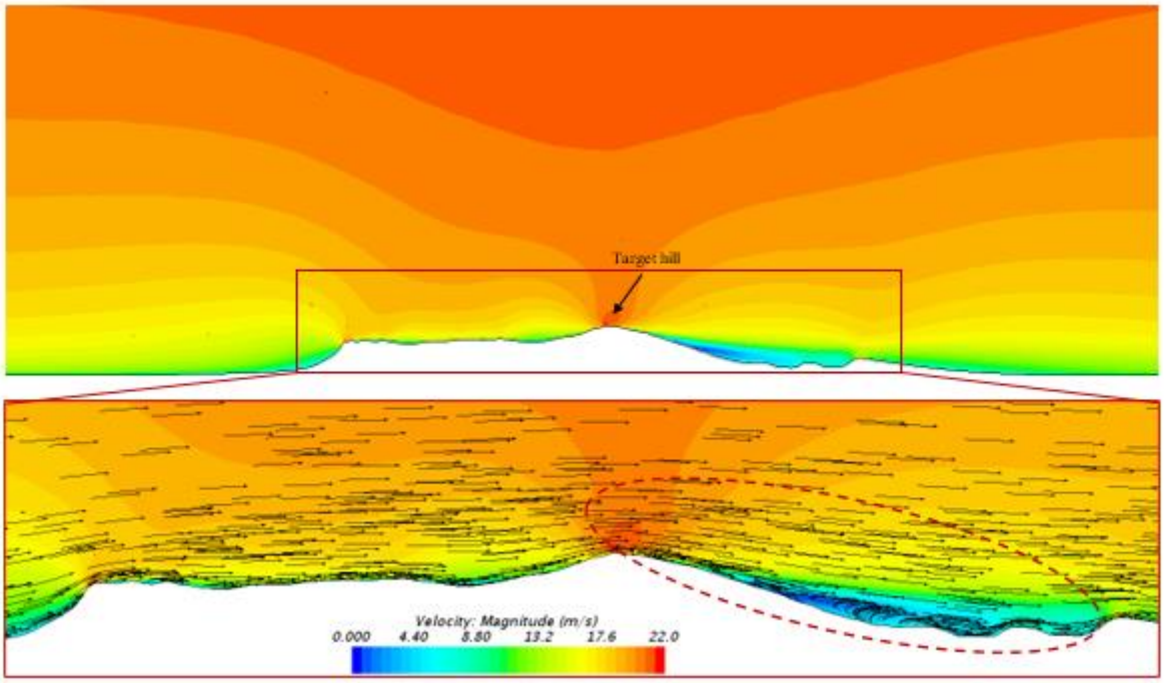


Figure 4. 11 Area1 velocity contour of LiDAR image for a wind direction of 0^0

The airflow velocity pattern near the ground varies between the LiDAR and SRTM images, this is mainly due to the resolution of the image captured and more roughness observed in the LiDAR image as illustrated in Fig. 4.11 and 4.12. Detail roughness near the surface topography affects the wind speed near the ground, which will have an effect on the boundary layer velocity profile. Thus, a higher speedup is observed as shown in Fig. 4.14.

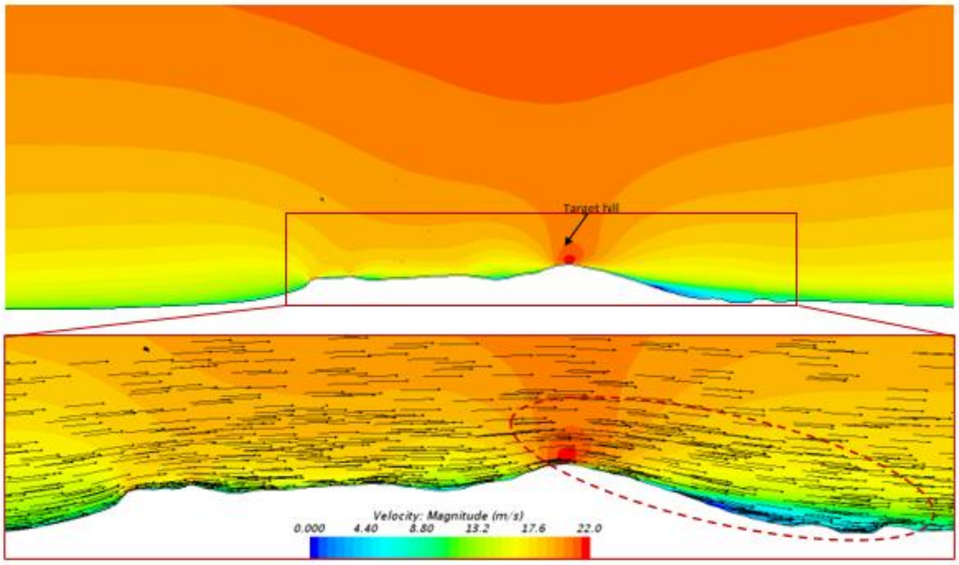
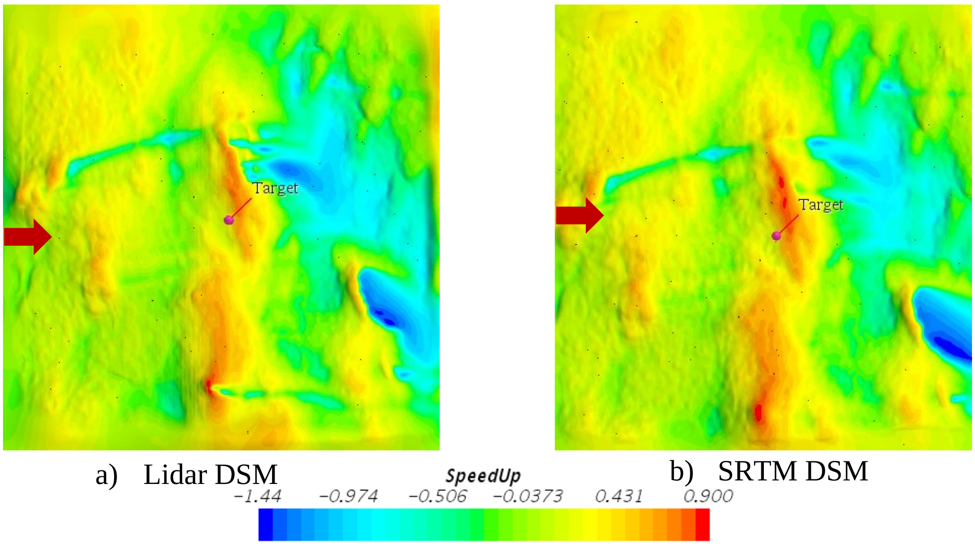


Figure 4. 12 Area1 velocity contour of SRTM image for a wind direction of 0^0



a) Lidar DSM

b) SRTM DSM

Figure 4. 13 Area1 speedup contour at 30 m from the ground for a wind direction of 0^0

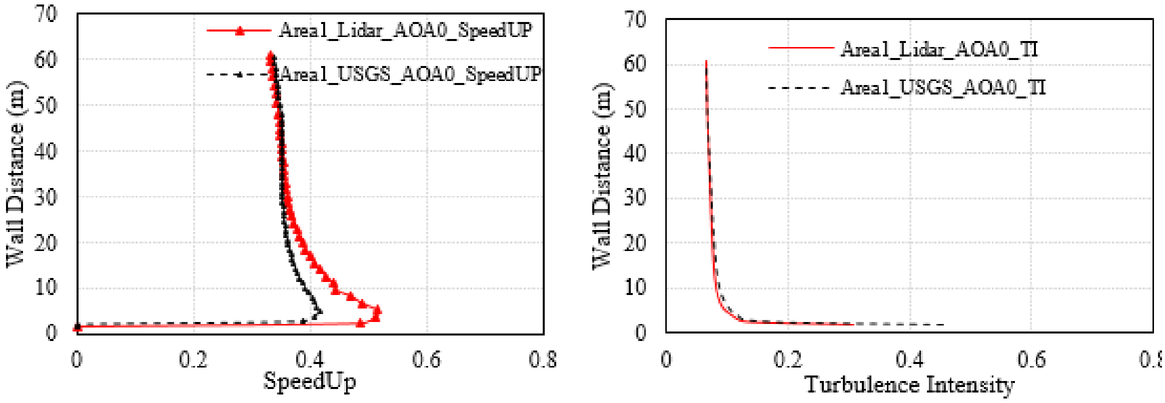


Figure 4. 14 Area1 speedup and turbulence intensity comparison between LiDAR and SRTM for wind direction of 0^0

Speedup comparison at 10 m above the ground on the selected Area1 between LiDAR and SRTM data, the LiDAR shows an increment of 13% as shown in Fig. 4.14. However, the turbulence intensity of the LiDAR shows a decrement of 5%.

LiDAR revealed more topographic details than SRTM. Considering the wind direction of 90^0 towards the upstream a high hill is observed as the selected target hill as shown in Fig. 4.15. However, considering the SRTM data at the same location relatively a flattened hill is observed as shown in Fig. 4.16. Accordingly, a speedup comparison is performed at the selected hills as shown in Fig. 4.17 a topographic speedup contour.

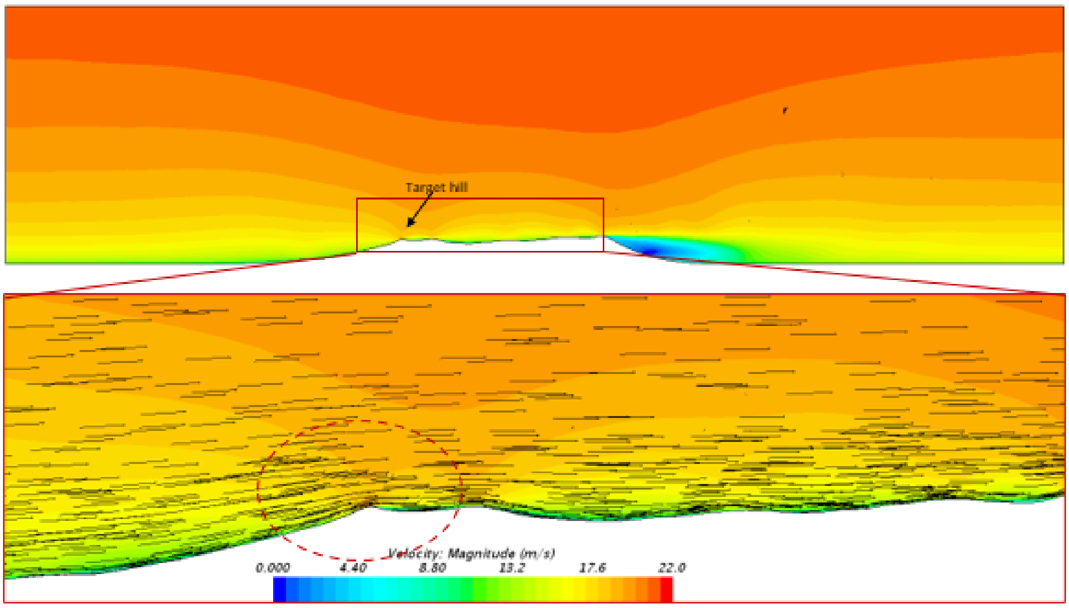


Figure 4. 15 Area1 velocity contour of LiDAR image for a wind direction of 90^0

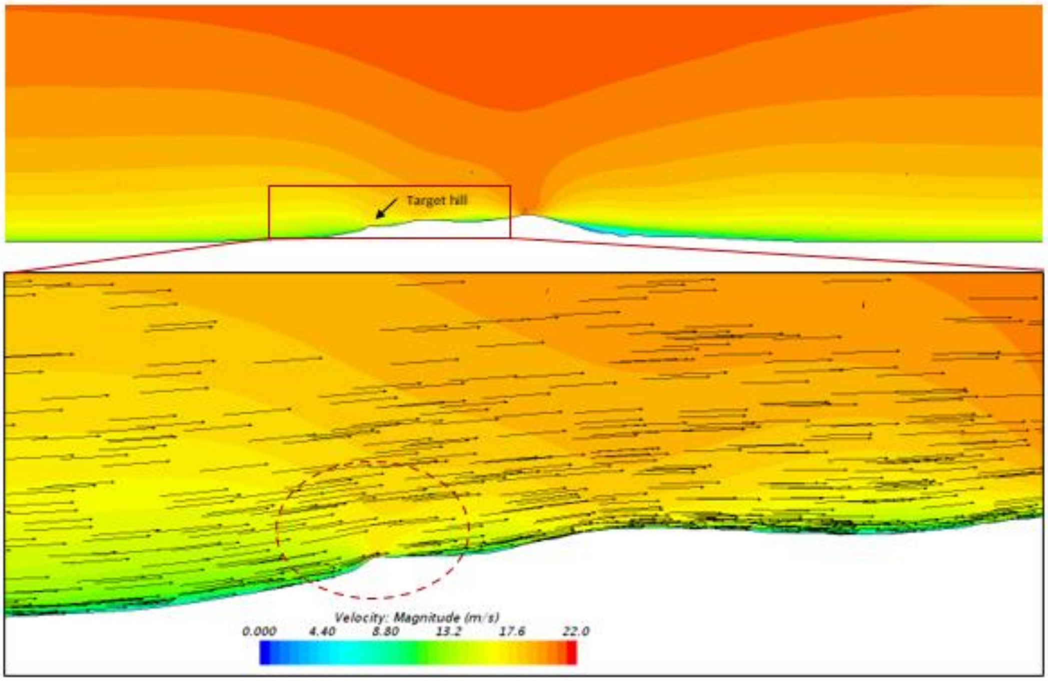
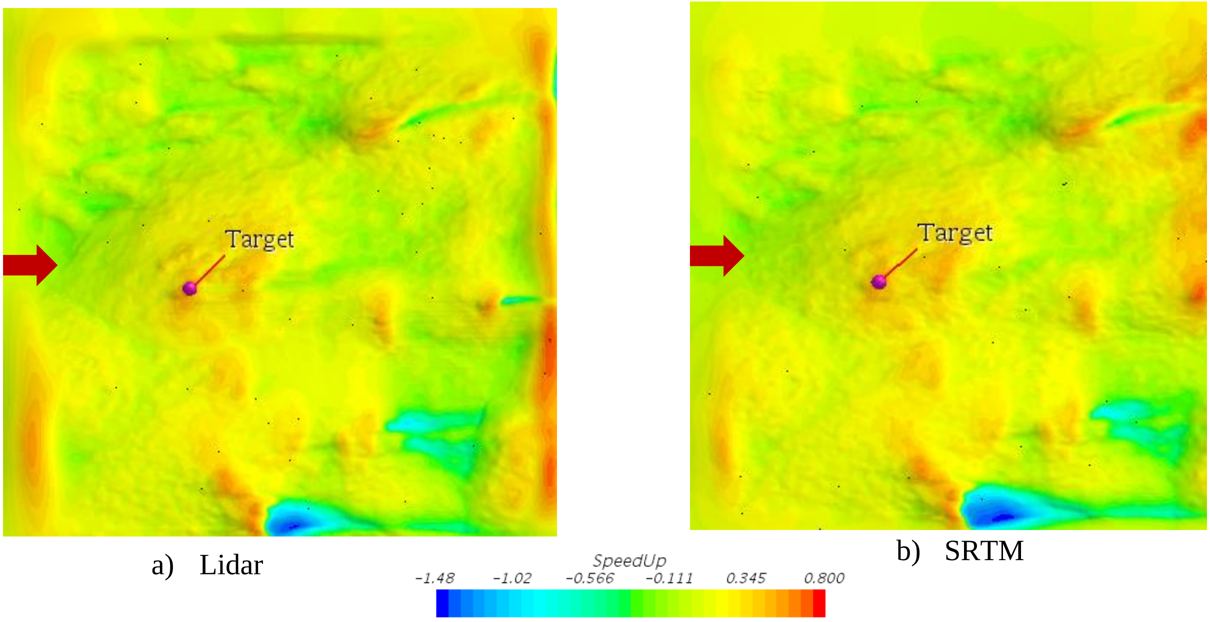


Figure 4. 16 Area1 velocity contour of SRTM image for a wind direction of 90^0



a) Lidar

b) SRTM

Figure 4. 17 Area1 speedup contour at 30 m from the ground for a wind direction of 90^0

For speedup comparison at 10 m above the ground on the selected Area1 between LiDAR and SRTM data, the LiDAR shows an increment of 14% but the turbulence intensity remains the same as shown in Fig. 4.18. Higher speedup is observed near the ground and decreases gradually to a height of 60 m above the ground. However, the turbulence intensity of the LiDAR and the SRTM almost shows the same.

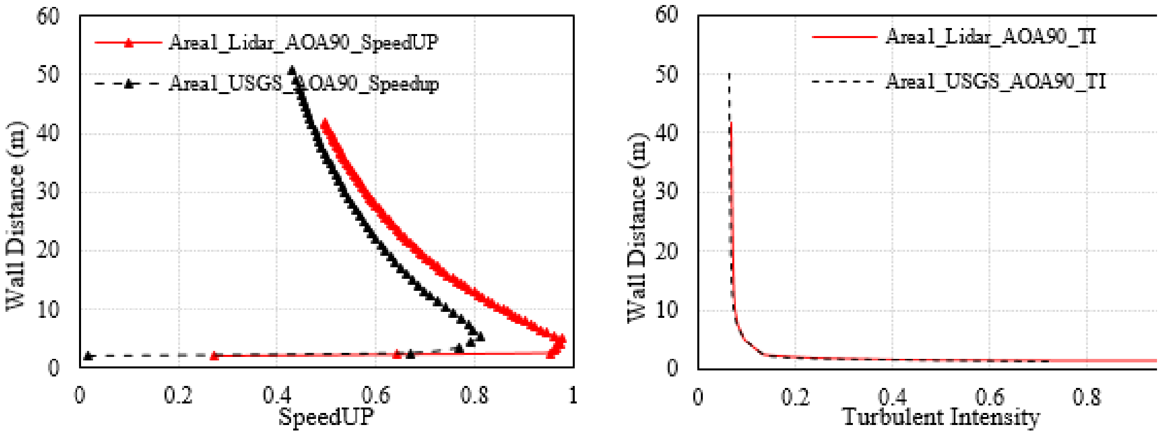


Figure 4. 18 Area1 speedup and turbulent intensity comparison between LiDAR and SRTM for wind direction of 90^0

In the upstream of the selected area, as shown in the velocity contour of Fig. 4.19, 4.20, 4.23 and 4.24, the LiDAR image shows a rougher surface than the SRTM image as shown also in the topographic speedup contour of Fig. 4.21 and 4.25.

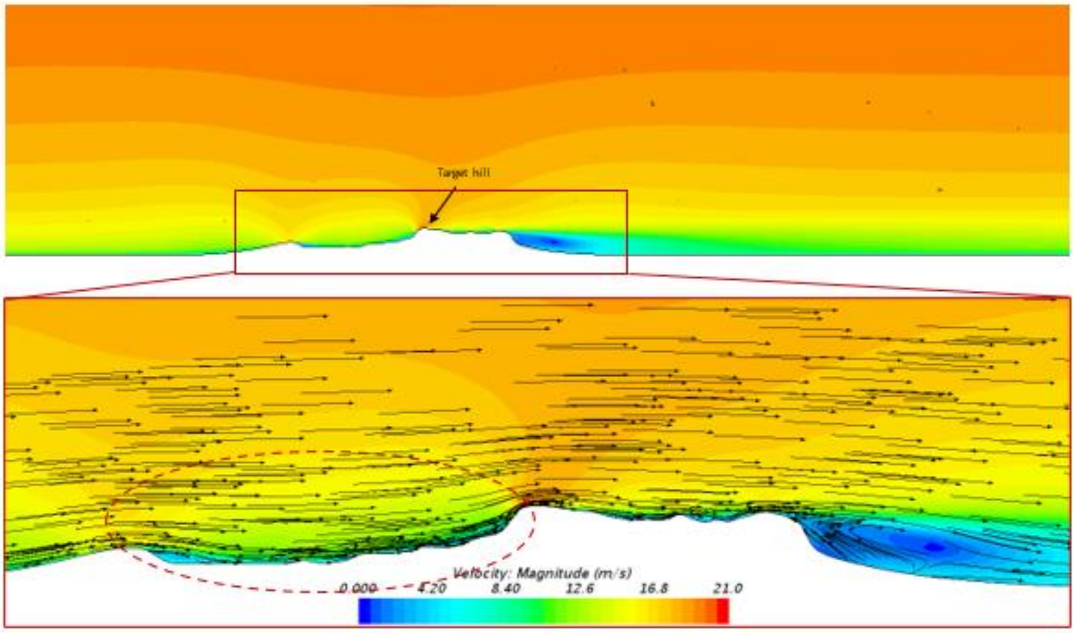


Figure 4. 19 Area2 velocity contour of LiDAR image for a wind direction of 0^0

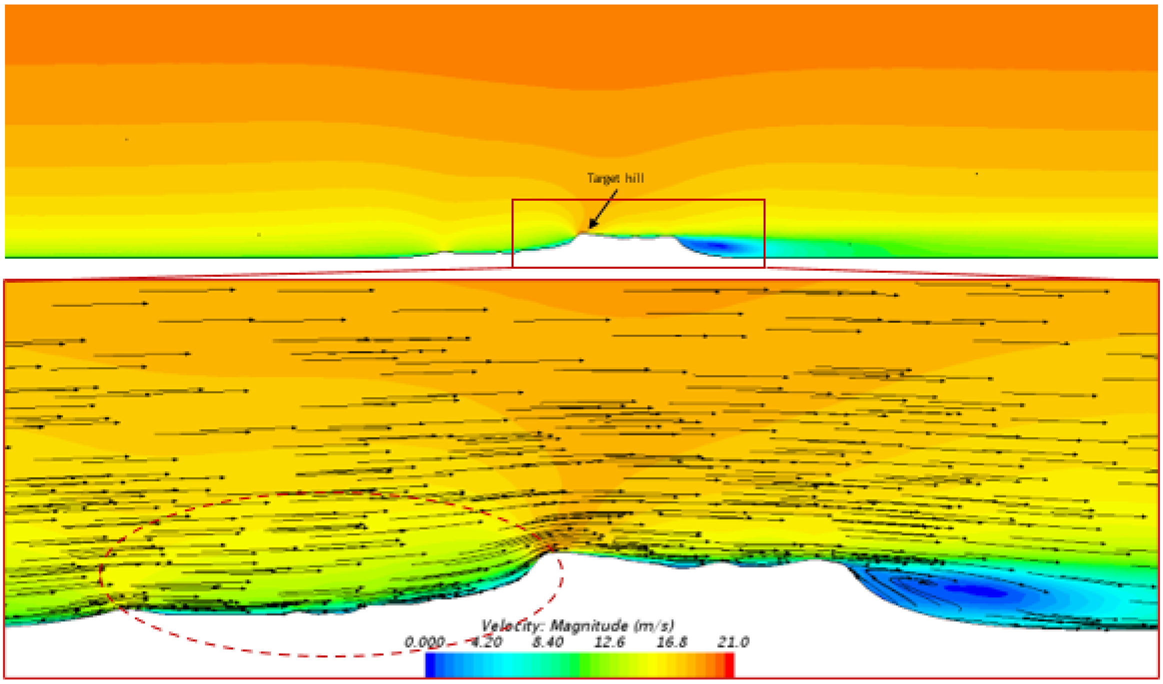
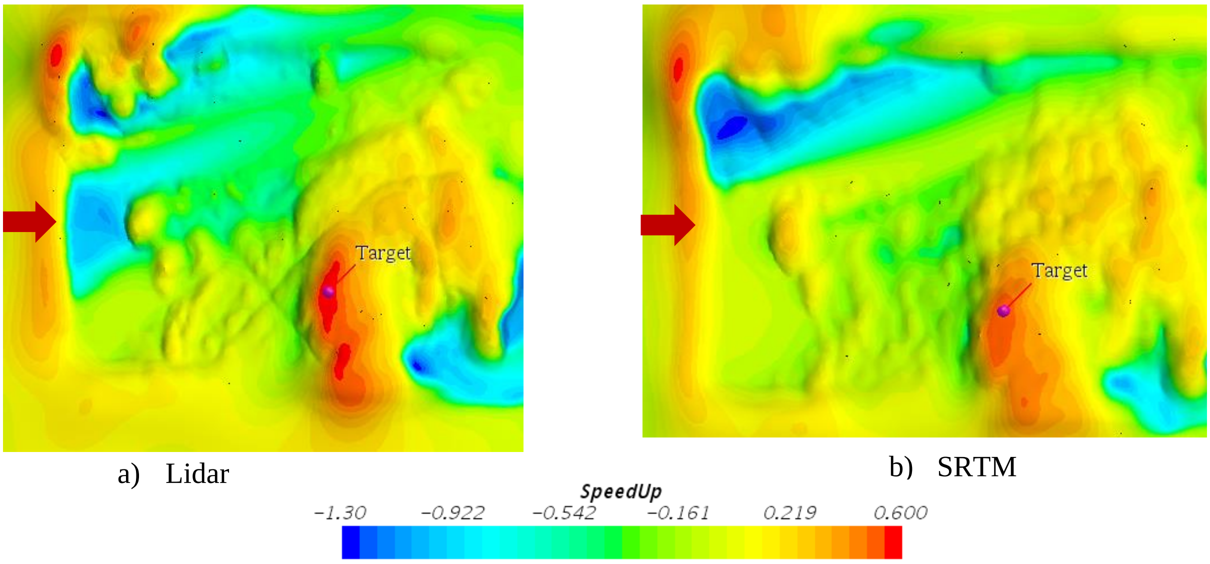


Figure 4. 20 Area2 velocity contour of SRTM image for a wind direction of 0^0


a) Lidar

b) SRTM

Figure 4. 21 Area2 speedup contour at 30 m from ground for a wind direction of 0^0

Speedup comparison at 10 m above the ground on the selected Area2 between LiDAR and SRTM data, the LiDAR shows an increment of 12% as shown in Fig. 4.22. However, the turbulence intensity of the LiDAR shows a decrement of 4%.

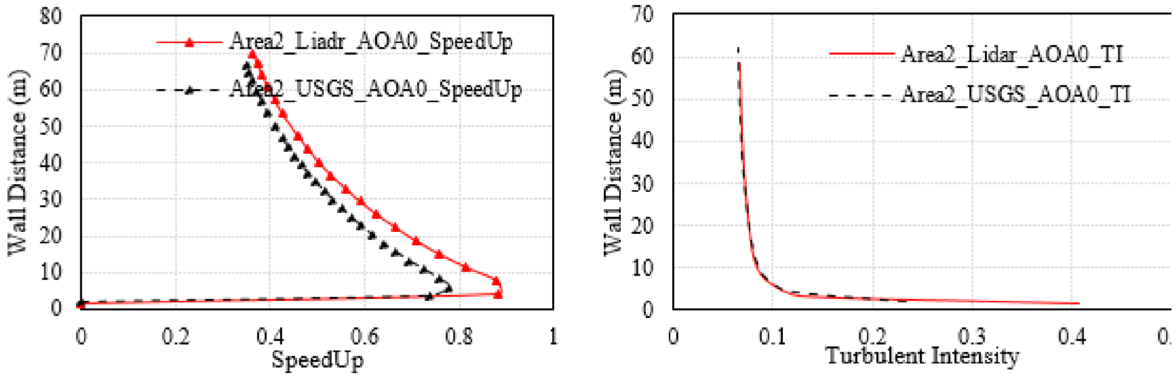


Figure 4. 22 Area2 speedup and turbulent intensity comparison between LiDAR and SRTM for wind direction of 0^0

The topographic speedup contour of Fig. 4.25 shows that upstream of the selected target hill, the LiDAR image reveals rougher ground than the SRTM image, thus having an impact on the speedup of the selected target hill.

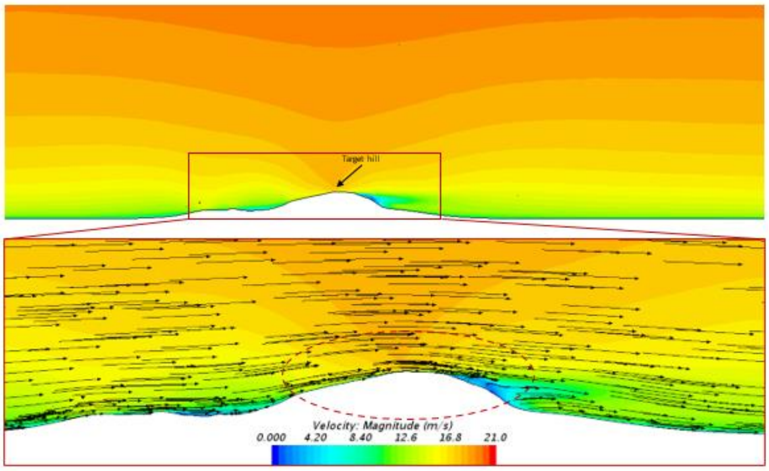


Figure 4. 23 Area2 velocity contour of LiDAR image for a wind direction of 90^0

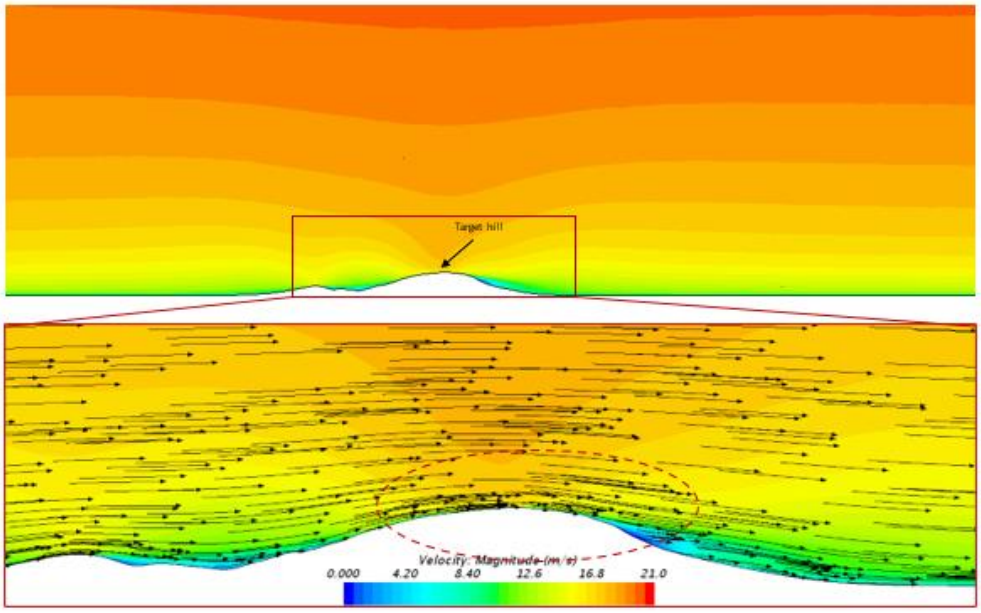
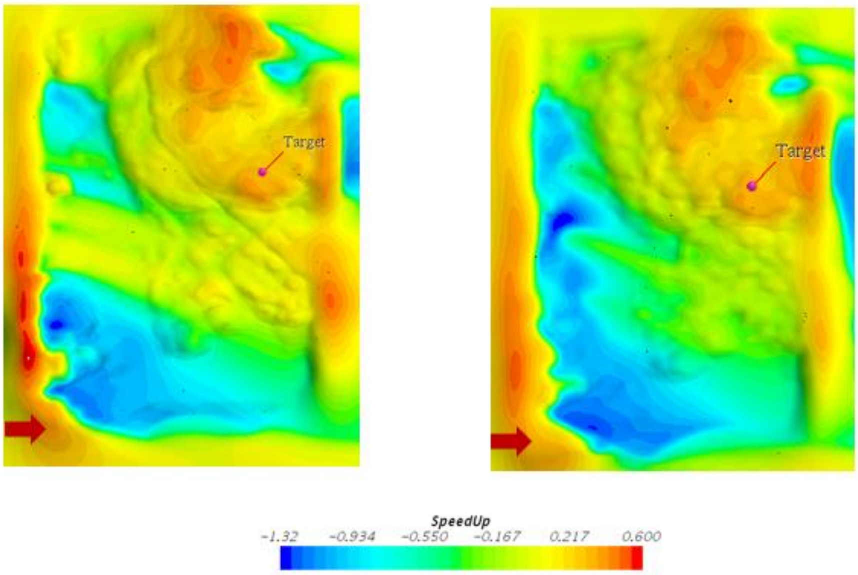


Figure 4. 24 Area2 velocity contour of SRTM image for a wind direction of 90^0



a) Lidar

b) SRTM

Figure 4. 25 Area2 speedup contour at 30 m from the ground for a wind direction of 90^0

Speedup comparison at 10 m above the ground on the selected Area2 between LiDAR and SRTM data, the LiDAR shows an increment of 14% as shown in Fig. 4.26. However, the turbulence intensity of the LiDAR shows a decrement of 10%.

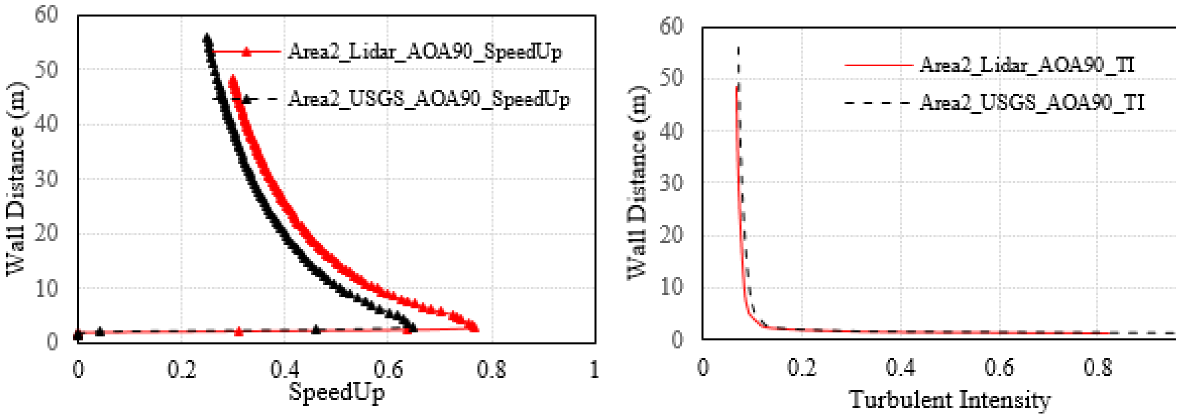


Figure 4. 26 Area2 speedup and turbulence intensity comparison between LiDAR and SRTM for wind direction of 90^0

Conclusion

A workflow for assessing wind flow over a complex terrain that uses advanced geometrical modelling has been developed. Particularly, this study provides a novel investigation of the image accuracy of LiDAR and SRTM to evaluate the speedup ratio. Thus, a detailed CFD simulation has been performed on selected two topographic areas. Considering the size of the topography, a grid sensitivity analysis and ABL flow homogeneity is done. A wind flow across and along the selected target hill is simulated and for all cases, a speedup ratio and turbulent intensity at 10 m from the ground are computed. Therefore, near the ground, the LiDAR topographical image shows a higher speedup ratio than the SRTM image, thus emphasizing the need for the use of more accurate geometrical modelling for assessing flows over complex terrain. Further, the developed 3D complex terrain flow application is useful to address overarching problems in topography effect studies such: (a) funneling effects in valleys or in between hills, (b) corner effects along the foot of mountains and hills, (c) vortex formation behind steep terrain, and (d) other 3D effects. The developed workflows together with the provisions in Codes and Standards are expected to assist engineers/scientists while assessing wind loads and wind energy resource availability.

Reference

Belcher, S.E., Hunt, J.C.R., 1998. Turbulent flow over hills and waves. Annu. Rev. Fluid Mech. 30, 507–538. https://doi.org/10.1146/annurev.fluid.30.1.507

Boache, P.J., 1994. Perspective: A method for uniform reporting of grid refinement studies. J. Fluids Eng. Trans. ASME 116, 405–413. https://doi.org/10.1115/1.2910291

Curcio, A.C., Peralta, G., Aranda, M., Barbero, L., 2022. Evaluating the Performance of High Spatial Resolution UAV-Photogrammetry and UAV-LiDAR for Salt Marshes: The Cádiz Bay Study Case. Remote Sens. 14. https://doi.org/10.3390/rs14153582

Jubayer, C.M., Hangan, H., 2018. A hybrid approach for evaluating wind flow over a complex terrain. J. Wind Eng. Ind. Aerodyn. 175, 65–76. https://doi.org/10.1016/j.jweia.2018.01.037

Kahsay, M.T., Bitsuamlak, G.T., Tariku, F., 2019. CFD simulation of external CHTC on a high-rise building with and without façade appurtenances. Build. Environ. 165, 106350. https://doi.org/10.1016/j.buildenv.2019.106350

Mahammad, S.S., Ramakrishnan, R., 2003. GeoTIFF - A standard image file format for GIS applications Map India 2003 Image Processing & Interpretation. Map India Conf.

Means, J.E., Acker, S.A., Fitt, B.J., Renslow, M., Emerson, L., Hendrix, C.J., 2000. Predicting forest stand characteristics with airborne scanning lidar. Photogramm. Eng. Remote Sensing 66, 1367–1371.

Menter, F.R., 1994. Two-equation eddy-viscosity turbulence models for engineering applications. AIAA J. 32, 1598–1605. https://doi.org/10.2514/3.12149

Richards, P.J., Norris, S.E., 2019. Appropriate boundary conditions for computational wind engineering: Still an issue after 25 years. J. Wind Eng. Ind. Aerodyn. 190, 245–255. https://doi.org/10.1016/j.jweia.2019.05.012

Ritter, N., Ruth, M., 1997. The GeoTIFF data interchange standard for raster geographic images. Int. J. Remote Sens. 18, 1637–1647. https://doi.org/10.1080/014311697218340

Song, J.-H., Han, S.-H., Yu, K., Kim, Y.-I., 2002. Assessing the possibility of land-cover classification using lidar intensity data. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch. 34.

Torre-Tojal, L., Bastarrika, A., Boyano, A., Lopez-Guede, J.M., Graña, M., 2022. Above-ground biomass estimation from LiDAR data using random forest algorithms. J. Comput. Sci. 58. https://doi.org/10.1016/j.jocs.2021.101517

Wang, X., Burt, C., Zarkesh-Ha, P., Neumann, A., Brueck, S.R.J., 2022. Indoor Lidar Plenoptic Array Based on Time-of-Flight Reflection. IEEE Photonics J. 14, 1–7. https://doi.org/10.1109/JPHOT.2022.3215482

Wang, Z., Zhang, J., Gao, H., 2021. Impacts of laser beam divergence on lidar multiple scattering polarization returns from water clouds. J. Quant. Spectrosc. Radiat. Transf. 268, 107618. https://doi.org/10.1016/j.jqsrt.2021.107618

Wichmann, V., Bremer, M., Lindenberger, J., Rutzinger, M., Georges, C., Petrini-Monteferri, F., 2015. Evaluating the potential of multispectral airborne lidar for topographic mapping and land cover classification. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. 2, 113–119. https://doi.org/10.5194/isprsannals-II-3-W5-113-2015

Wood, N., 1995. The onset of separation in neutral, turbulent flow over hills. Boundary-Layer Meteorol. 76, 137–164. https://doi.org/10.1007/BF00710894

AS/NZ1170.2, (2002), “Australian/New Zealand standard, structural design actions, part 2: wind actions,” Standards Australia & Standards New Zealand.

ASCE 7-10, (2010), “Minimum Design Loads for Buildings and Other Structures,” American Society of Civil Engineers / 12-May-2010 / 658 pages, ISBN: 9780784410851.

ASCE 7-02, (2002), “ASCE standard: minimum design loads for buildings and other structures,” American Society of Civil Engineers.

Armitt, J., Cojan, M., Manuzio, C. and Nicolini, P. (1975), “Calculation of wind loadings on components of overhead lines”, *Proceedings of the Institution of Electrical Engineers,* **122**(11), 1247-52.

Arya, S.P.S., Capuano, M.E. and Fagen, L.C. (1987), “Some fluid modelling studies of flow and dispersion over two-dimensional low hills”, *Atmospheric Environment - Part A General Topics,* **21**(4), 753-764.

Athanassiadou, M. and Castro, I.P. (2001), “Neutral flow over a series of rough hills: a laboratory experiment”, *Bound. -Layer Meteorol.,* **101**(1), 1-30.

Ayotte, K.W. and Hughes, D. (2004), “Observations of boundary-layer wind-tunnel flow over isolated ridges of varying steepness and roughness”, *Bound. -Layer Meteorol.,* **112**(3), 525-56.

Belcher, S. E. and Hunt, J. C. R. (1998), “Turbulent flow over hills and waves,” *Annu. Rev. Fluid Mech.* 30:507–38

Bitsuamlak, G., Stathopoulos, T. and Bedard, C. (2006), “Effects of upstream two-dimensional hills on design wind loads: A computational approach”, Wind and Structures, An International Journal, 9(1), 37-58.

Bitsuamlak, G.T., Bedard, C. and Stathopoulos, T. (2007), “Modelling the effect of topography on wind flow using a combined numerical-neural network approach”, *J. Comput. Civ. Eng.,* **21**(6), 384-392.

Bitsuamlak, G.T., Stathopoulos, T. and Bedard, C. (2004), “Numerical evaluation of wind flow over complex terrain: Review”, *J. Aerospace Eng.,* **17**(4), 135-145.

Cao, S. and Tamura, T. (2007), "Effects of roughness blocks on atmospheric boundary layer flow over a two-dimensional low hill with/without sudden roughness change", *J. Wind Eng. Ind. Aerodyn,* **95**(8), 679-95.

Carpenter, P. and Locke, N. (1999), "Investigation of wind speeds over multiple two-dimensional hills", *J. Wind Eng. Ind. Aerodyn.,* **83**(1-4), 109-120.

Castro, F.A., Palma, J.M.L.M. and Lopes, A.S. (2003), "Simulation of the askervein flow. Part 1: Reynolds averaged navier-stokes equations (k - turbulence model)", *Bound. -Layer Meteorol.,* **107**(3), 501-530.

Coppin, P.A., Bradley, E.F. and Finnigan, J.J. (1994), "Measurements of flow over an elongated ridge and its thermal stability dependence: the mean field", *Bound. -Layer Meteorol.,* **69**(1-2), 173-99.

Ferreira, A. D., Lopes, A. M. G., Viegas, D. X., and Sousa, A. C. M., (1995), "Experimental and numerical simulation of flow around two-dimensional hills," *J. Wind Eng. Ind. Aerodyn*, 54/55, 173–181.

Finnigan, J.J., Raupach, M.R., Bradley, E.F. and Aldis, G.K. (1990), "A wind tunnel study of turbulent flow over a two-dimensional ridge", *Bound. -Layer Meteorol.,* **50**(1-4), 277-317.

Gong, W. and Ibbetson, A. (1989), "A wind tunnel study of turbulent flow over model hills", *Bound. -Layer Meteorol.,* **49**(1-2), 113-148.

Ishihara, T., Hibi, K. and Oikawa, S. (1999), "A wind tunnel study of turbulent flow over a three-dimensional steep hill", *J. Wind Eng. Ind. Aerodyn.,* **83**(1-4), 95-107.

Jackson, P.S. and Hunt, J.C.R. (1975), "Turbulent wind flow over a low hill", *Q. J. R. Meteorol. Soc.,* **101**(430), 929-55.

Mason P. J. and Sykes R. I., (1979), "Flow over an isolated hill of moderate slope," Q. J. R. Meteorol. Soc. 105:383–95.

Miller, C.A., and Davenport, A. G., (1998), "Guidelines for the Calculation of Wind Speed-ups in Complex Terrain," *J. Wind Eng. Ind. Aerodyn.,* 74-76, 189-197.

Lemelin, D.R., Surry, D. and Davenport, A.G. (1988), "Simple approximations for wind speed-up over hills", *J. Wind Eng. Ind. Aerodyn.,* **28 pt 1**(1-3), 117-127.

Lubitz, W.D. and White, B.R. (2007), "Wind-tunnel and field investigation of the effect of local wind direction on speed-up over hills", *J. Wind Eng. Ind. Aerodyn,* **95**(8), 639-61.

National Building Code of Canada (NBCC), (1995), "National building code user's guide, Structural commentaries (Part 4)," Canadian Commission on Building and Fire Codes, National Research Council of Canada, Ottawa.

National Building Code of Canada (NBCC), (2005), Canadian Commission on Building and Fire Codes, National Research Council of Canada, Ottawa.

Mason, P.J. and Sykes, R.I. (1979), "Flow over an isolated hill of moderate slope", *Q. J. R. Meteorol. Soc.,* **105**(444), 383-95.

Savory, E., Parke, G.A.R., Disney, P. and Toy, N. (2008), "Wind-induced transmission tower foundation loads: A field study-design code comparison", *J. Wind Eng. Ind. Aerodyn.,* **96**(6-7), 1103-1110.

Snyder, W.H. and Britter, R.E. (1987), "A wind tunnel study of the flow structure and dispersion from sources upwind of three-dimensional hills", *Atmospheric Environment - Part A General Topics,* **21**(4), 735-751.

Pinard, J. P. and Wilson, J. D., (1999), "Computer models for wind flow over mesoscale mountainous terrain applied to the yukon," Final Report in Forestry.

Taylor, P.A., Mason, P.J. and Bradley, E.F. (1987), "Boundary-layer flow over low hills", *Bound. -Layer Meteorol.,* **39**(1-2), 107-32.

Taylor, P.A., Walmsley, J.L. and Salmon, J.R. (1986), "Guidelines and models for estimating wind speeds at wecs sites in complex terrain.", *Intersol 85*, Proceedings of the Ninth Biennial Congress of the International Solar Energy Society.

Taylor, P.A., Walmsley, J.L. and Salmon, J.R. (1983), "A simple model of neutrally stratified boundary-layer flow over real terrain incorporating wavenumber-dependent scaling", *Bound. -Layer Meteorol.,* **26**(2), 169-89.

Walmsley, .I. L., Taylor, P. A., and Mok, R., (1980), "MS3DJH - a computer model for the study of neutrally stratified boundary-layer flow over isolated hills of moderate slope," Rep. AQRB-80-008-L, Atmos. Environ. Service, Downsview, Ontario.

Walmsley, J.L., Salmon, J.R. and Taylor, P.A. (1982), "On the application of a model of boundary-layer flow over low hills to real terrain", *Bound. -Layer Meteorol.,* **23**(1), 17-46.

Weng, W., Taylor, P.A. and Walmsley, J.L. (2000), "Guidelines for airflow over complex terrain: Model developments", *J. Wind Eng. Ind. Aerodyn.,* **86**(2-3), 169-186.

Wood, N. (1995), "The onset of separation in neutral, turbulent flow over hills", *Bound. -Layer Meteorol.,* **76**(1-2), 137-64.

Chapter 5

Conclusions and Recommendations

5.1 Overview

In this chapter, the main contributions, findings, and limitations of the research work are summarized. This research presented three studies that contribute to the autonomous geometry modelling and its application for micro-climate simulations. The main objective of this research is to develop automated site-specific 3D urban and complex terrain topology and micro-climate modelling for wind engineering and building science applications. The thesis explored different methods before eventually choosing the optimum technique for remote sensing data analysis. Machine learning and deep learning techniques were tested for building footprint feature extraction. The deep learning model was selected for further analysis due to its high efficiency. The resulting building footprint polygons were used to generate digital elevation model by integration with point cloud data for solar power potential assessment. LiDAR was also combined with 2D polygons to create 3D building models for PLW CFD analysis. Also, a comparison of a satellite image-based model and LiDAR based complex terrain wind flow analysis was carried out.

5.2 Chapter 2: A deep learning model-based building footprint polygon extraction for a GIS-based Solar power potential estimation

In this chapter of the research, autonomous building footprint polygon extraction method was successfully developed. The deep learning technique achieved an improved result over traditional machine learning algorithms such as random trees, SVM object-based classification, maximum likelihood. The advantage of the deep learning model is that it can be applied to a new set of input image data to extract building footprint polygons for autonomous applications once it’s trained. In addition, the model can be improved over time with minimum adjustments when quality data is available. Although it is recommended to deep learning and provides a higher accuracy result over machine learning techniques, it still requires upgrading and to continuously learn with new data to predict unique building forms that are unusual. In addition, when the number of buildings being predicted at a single moment surge, the model struggles to capture specifics. This is mainly due to the limitation of the training set and computational requirements and hence it will perform better with more training data.

By combining the extracted residential building footprint polygons with LiDAR generated DSM, it was possible to automate the calculation of household solar panel estimation. Constraints such as removing rooftops with a slope of more than 45˚, north-facing buildings were implemented. The solar power potential estimation can be scaled to include entire neighborhoods as long as high-density LiDAR data is available. But most importantly, the application of machine learning to extract the residential house's polygon area used for the solar panel calculation paved the way for a more robust and automated workflow.

5.3 Chapter 3: Autonomous urban topology generation for urban flow modelling

In this part, a new framework is developed that integrates the autonomous urban topology generator with urban flow modelling. A deep learning model for building footprints extraction from satellite imagery is developed and used to generate 3D building models by integrating it with Light Detection and Ranging (LiDAR) data to generate 3D building models. The 3D models are meshed and used in the CFD modelling modules. The entire process is explained through an application example of urban flow modelling encountered during a pedestrian-level wind assessment. The U-Net-based deep learning algorithm achieved a 3D building model with a LOD3 value. The image segmentation part of the building 3D modelling workflow is critical to the autonomous concept since the deep learning model can be reused to predict a new set of images. The model can be adapted to image samples that are significantly different from the images on which it was trained with minimum training samples. The combination uses of images and point cloud data addressed the issues of roof structure modelling and most of the building facades are also captured with the workflow used.

Since the buildings are classified one at a time, it would be more convenient if the collected testing images are georeferenced beforehand to make the workflow quicker. The resolution of the LiDAR data should always be of higher quality, as lower quality will cause some structures to not be complete. Especially, the roof structures are more affected by lower-quality point cloud resolution. The predicted building outline polygons can further be processed to achieve more refined edges. But this will be limited to the type of application that these polygons will be used as in some cases the extra processing seems to chop off critical details from the polygons. Although the model can accurately identify different building shapes, it still requires improvement to predict unique building shapes that are uncommon. The model needs additional training to make a more accurate prediction of buildings which have various extra details placed on them. The accuracy of the final 3D model outcome of the buildings is also affected by the density of the LiDAR data. The absence of dense point cloud data has affected some details of the building models generated in this thesis. The autonomously generated 3D models are seamlessly integrated with the physics modelling to simulate turbulence flow in urban areas that are used among other applications for pedestrian-level wind assessment, thus, reducing the Engineering and Tech time-intensive manual computational model generation process.

In chapter 2, the deep learning model was trained on satellite images that represent the top view of residential buildings. The parameters used for the neural network training were adjusted and fine tuned based on the type of buildings that are predicted. A separate deep learning model was used in Chapter 3 that was trained on satellite images of high rise buildings in a downtown urban setting. The model underwent various modifications through various parameter tuning procedures to fit the type of input data. While the 2D polygons from the deep learning model in Chapter 2 were used for the solar power potential estimation, the model from Chapter 3 was used to generate 3D building models as part of an autonomous workflow for pedestrian level wind assessment.

5.4 Chapter 4: Modelling wind flow over complex terrain generated by using LiDAR and SRTM

In this section, a workflow for assessing wind flow over a complex terrain that uses advanced geometrical modelling has been developed. The chapter investigated the data accuracy of LiDAR and SRTM to evaluate the speedup ratio. A CFD simulation has been performed on selected two topographic areas. A grid sensitivity analysis and ABL flow homogeneity is also done. A wind flow across and along the selected target hill is simulated and for all cases, a speedup ratio and turbulent intensity at 10 m from the ground are computed.

The LiDAR topographical image shows a higher speedup ratio than the SRTM image near the ground. This emphasized the need for the use of more accurate geometrical modelling for assessing flows over complex terrain. The developed 3D complex terrain flow application is useful to address funneling effects in valleys or in between hills, corner effects along the foot of mountains and hills, vortex formation behind steep terrain, and other 3D effects.

5.5 Limitations of the thesis

Although the deep learning model can accurately identify different building shapes, it still requires improvement to predict unique building shapes that are uncommon. The model needs additional training to make a more accurate prediction of buildings which have various extra details placed on them. Also, when the number of buildings being predicted at a single moment increases, the model struggles to capture more details. This is mainly due to the limitation of the training set and computational capacity requirements. But the workflow described in this thesis can be considered as a starting point for accurate and improved building model predictions at a city scale when quality training set is provided along with sufficient computational resources. The accuracy of the final 3D model outcome of the buildings is also affected by the density of the LiDAR data. The absence of dense point cloud data has affected some details of the building models generated in this thesis. Overall, computational capacity limitation was one of the main challenges throughout the thesis.

5.6 Future Research Directions

- The deep learning approach has the potential to be applied to different data types in addition to urban geometric information, such as material types to include modelling of energy and thermal performance of buildings and neighbourhood.

  Deep learning can be applied for modelling the energy and thermal performance of buildings in several ways.

  Predicting energy consumption: Deep learning can be used to predict the energy consumption of a building by analyzing historical energy consumption data and other relevant parameters such as outdoor temperature, occupancy, and building characteristics. This can help building managers optimize energy usage and reduce costs.

  Fault detection and diagnosis: Deep learning can be used to detect and diagnose faults in building systems, such as HVAC systems, by analyzing sensor data from the building. This can help identify inefficiencies and prevent energy waste.

  Building envelope optimization: Deep learning can be used to optimize the design of building envelopes, such as walls, roofs, and windows, for thermal performance. This can help reduce energy consumption and improve indoor comfort.

  Indoor air quality prediction: Deep learning can be used to predict indoor air quality based on various parameters such as occupancy, outdoor air quality, and ventilation rates. This can help building managers optimize ventilation rates and maintain a healthy indoor environment.

- Similar approaches like autonomously generated 3D models integrated with more complex transient physics modelling to simulate turbulence effects in urban areas and for handling urban heat transfer modelling.

  Autonomously generated 3D building models can be used for complex transient physics modelling to simulate turbulence effects in urban areas and for handling urban heat transfer modelling in the following ways:

  Creating accurate geometry: Autonomously generated 3D building models can provide accurate representations of the geometry of urban areas, including building heights, shapes, and orientations. This information is critical for modelling complex transient physics such as turbulence effects and urban heat transfer.

  Generating meshes: The 3D building models can be used to generate meshes for numerical simulations, which are essential for solving the equations that govern fluid flow and heat transfer. The meshes can be generated automatically, reducing the time and effort required for meshing.

  Simulating turbulence effects: The 3D building models can be used to simulate turbulence effects in urban areas. This includes modelling the interactions between the airflow and the buildings, which can have a significant impact on the urban heat transfer and energy consumption.

  Handling urban heat transfer modelling: The 3D building models can be used to model the heat transfer in urban areas, including radiation, conduction, and convection. This allows for the analysis of urban heat island effects and the development of strategies to mitigate them.

- Interior scans can also be developed for natural ventilation applications.

Assessing natural ventilation potential: Interior scans can provide detailed information about the layout and features of a building, such as the location and size of windows, doors, and vents. This information can be used to assess the potential for natural ventilation, by simulating airflows and identifying areas with the best potential for air movement.

Optimizing ventilation strategies: Interior scans can also be used to optimize natural ventilation strategies. For example, by simulating airflows in a building, it is possible to determine the most effective locations for vents and windows to maximize air movement and ventilation efficiency.

Designing HVAC systems: Interior scans can be used to design HVAC systems that complement natural ventilation strategies. By simulating airflows and analyzing ventilation patterns, it is possible to design HVAC systems that provide additional air movement where natural ventilation is insufficient, or to optimize the use of HVAC systems to work in conjunction with natural ventilation.

Predicting indoor air quality: Interior scans can also be used to predict indoor air quality based on natural ventilation strategies. By simulating airflows and analyzing the movement of pollutants, it is possible to predict the impact of natural ventilation on indoor air quality and identify areas that may require additional ventilation or air purification measures.

Appendix

Appendix A: LiDAR and image processing

A1: Morphological filter algorithm (Phyton Scripts)

```
Import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('png.png', cv2.IMREAD_GRAYSCALE)
_, mask = cv2.threshold(img, 100, 200, cv2.THRESH_BINARY_INV)
kernel = np.ones((2,2), np.uint8)
erosion = cv2.erode(mask, kernel, iterations=1)
dilation = cv2.dilate(mask, kernel, iterations=20)
titles = ['image', 'mask', 'erosion', 'dilation']
images = [img, mask, erosion, dilation]
for i in range(4):
    plt.subplot(2, 2, i+1), plt.imshow(images[i], 'gray')
    plt.title(titles[i])
    plt.xticks([]), plt.yticks([])
plt.show()
```

A2: The clip LiDAR to polygon algorithm

```
import os
from WBT.whitebox_tools import WhiteboxTools
wbt = WhiteboxTools()
wbt.work_dir = os.path.dirname(os.path.abspath(__file__))
wbt.clip_lidar_to_polygon ( i="TD2.las",  polygons="TD.shp",
    output="buildings.las", )
```

A3: Point cloud ground filtering algorithm

Python function:

```
wbt.lidar_ground_point_filter(
    i,
output,
            radius=2.0,
            min_neighbours=0,
            slope_threshold=45.0,
            height_threshold=1.0,
            classify=True,
            slope_norm=True,
            height_above_ground=False,
            callback=default_callback
        )
```

Command-line interface:

```
>>./whitebox_tools -r=LidarGroundPointFilter -v ^
--wd="/path/to/data/" -i="input.las" -o="output.las" ^
--radius=10.0 --min_neighbours=10 --slope_threshold=30.0 ^
--height_threshold=0.5 --classify --slope_norm
```

A4: Software used

OSS – Open Source Software

OAS – Open Access Software

whitebox-1.3.0 version tool package is installed on Python 3.8

Scholars GeoPortal: is a search and discovery tool for extracting geospatial data. Ontario University students, faculty, and researchers can easily search, discover, map, share, and download geospatial data.

A5: VGG Image Annotator(VIA)

VGG Image Annotator is a simple and standalone manual annotation tool for images, audio and video. This is a light weight, standalone and offline software package that does not require any installation or setup and runs solely in a web browser. The VIA software allows human annotators to define and describe spatial regions in images or video frames, and temporal segments in audio or video. These manual annotations can be exported to plain text data formats such as JSON and CSV and therefore are amenable to further processing by other software tools. VIA also supports collaborative annotation of a large dataset by a group of human annotators. The BSD open source license of this software allows it to be used in any academic project or commercial application.

A6: Code for converting JSON files into image masks

```
Import  json
import os
import numpy as np
import PIL.Image
import cv2
import matplotlib.pyplot as plt
with open("eight.json", "r") as read_file:
    data = json.load(read_file)
all_file_names=list(data.keys())
Files_in_directory = []
for root, dirs, files in os.walk("eight"):
    for filename in files:
        Files_in_directory.append(filename)
for j in range(len(all_file_names)):
    image_name=data[all_file_names[j]]['filename']
    if image_name in Files_in_directory:
         img = np.asarray(PIL.Image.open('eight/'+image_name))
    else:
        continue
    if data[all_file_names[j]]['regions'] != {}:
        #cv2.imwrite('images/%05.0f' % j +'.jpg',img)
        print(j)
        try:
            shape1_x=data[all_file_names[j]]['regions']['0']['shape_attributes']['all_points_x']
            shape1_y=data[all_file_names[j]]['regions']['0']['shape_attributes']['all_points_y']
        except :
            shape1_x=data[all_file_names[j]]['regions'][0]['shape_attributes']['all_points_x']
            shape1_y=data[all_file_names[j]]['regions'][0]['shape_attributes']['all_points_y']
        fig = plt.figure()
        plt.imshow(img.astype(np.uint8))
        plt.scatter(shape1_x,shape1_y,zorder=2,color='red',marker = '.', s= 55)
        ab=np.stack((shape1_x, shape1_y), axis=1)
        img2=cv2.drawContours(img, [ab], -1, (255,255,255), -1)
        mask = np.zeros((img.shape[0],img.shape[1]))
        img3=cv2.drawContours(mask, [ab], -1, 255, -1)
        cv2.imwrite('eightmask/%05.0f' % j +'.png',mask.astype(np.uint8))
import os
path = os.chdir("D:\\All\\phd research\\THESIS\\thesis
2\\binary_mask_from_json\\eightmask")
```

```
i = 13
for file in os.listdir(path):
    new_file_name = "{}.png".format(i)
    os.rename(file, new_file_name)
    i=i+1
import PIL
import os
from PIL import Image
f = r"D:\\All\\phd research\\THESIS\\thesis 2\\binary_mask_from_json\\eightmask"
os.listdir(f)
for file in os.listdir(f):
    f_img = f+"/"+file
    img = Image.open(f_img)
    img = img.resize((256,256))
    img.save(f_img)
f = r"D:\\All\\phd research\\THESIS\\thesis 2\\binary_mask_from_json\\eight"
os.listdir(f)
for file in os.listdir(f):
    f_img = f+"/"+file
    img = Image.open(f_img)
    img = img.resize((256,256))
    img.save(f_img)
```

A7: RGB to grayscale

```
OutputFolder='D:\All\phdresearch\THESIS\thesis2\matlab_deep
learning\wall_segmentation\data\traingray';  % Set as needed [EDITED]

dinfo = dir('*.png');% image extension

for K = 1 : length(dinfo)

  thisimage = dinfo(K).name;

  Img   = imread(thisimage);

  Y     = imshow(Img);

  Gray  = rgb2gray(Img);

  imwrite(Gray, fullfile(OutputFolder, thisimage));  % [EDITED]

end
```

A8: Deep learning image segmentation code

```
dataFolder  = fullfile('data');

imageDir = fullfile(dataFolder,'traingray');

labelDir = fullfile(dataFolder,'masks');
```

```
numObservations = 4;

trainImages = repelem({imageDir},numObservations,1);

trainLabels = repelem({labelDir},numObservations,1);

classNames = ["building","background"];

labelIDs   = [255 0];

imds = imageDatastore(trainImages);

pxds = pixelLabelDatastore(trainLabels,classNames,labelIDs);

trainingData = combine(imds,pxds);

data = read(trainingData);

I = data{1};

C = data{2};

B = labeloverlay(I,C);

imshow(B)

augmentedTrainingData = transform(trainingData,@jitterImageColorAndWarp);

size(alldata)

data = readall(augmentedTrainingData);

rgb = cell(numObservations,1);

for k = 1:numObservations

   I = data{k,1};

   C = data{k,2};

   rgb{k} = labeloverlay(I,C);

end

montage(rgb)

imageSize = [256 256];

numClasses = 2;

lgraph = unetLayers(imageSize, numClasses)

lgraph.Layers
```

```
targetSize= [144 144];
preprocessedTrainingData = transform(augmentedTrainingData,...
    @(data)centerCropImageAndLabel(data,targetSize));
data=readall(preprocessedTrainingData);
rgb = cell(numObservations,1);
for k = 1:numObservations
    I = data{k,1};
    C = data{k,2};
    rgb{k} = labeloverlay(I,C);
end
montage(rgb)
alldata=combine(trainingData,augmentedTrainingData);
val_img="data\valimggray";
val_mask="data\val_masks";
valimgids=imageDatastore(val_img);
valclassNames = ["building","background"];
vallabelIDs   = [255 0];
valpxds = pixelLabelDatastore(val_mask,valclassNames,vallabelIDs);
valData = combine(valimgids,valpxds);
checkpointPath = pwd;
options = trainingOptions('adam', ...
    'InitialLearnRate',1e-3, ...
    'MaxEpochs',30, ...
    'Shuffle','every-epoch', ...
    'MiniBatchSize',4, ...
    'VerboseFrequency',50, ...
    'Plots','training-progress', ...
    'ValidationData',valData, ...
    'CheckpointPath',checkpointPath);


net = trainNetwork(alldata,lgraph,options)
alldata9694=net
save alldata9694
netaugment = trainNetwork(augmentedTrainingData,lgraph,options)
netaugment_30e_3lr_adam=netaugment
save netaugment_30e_3lr_adam
netaugmented2 = trainNetwork(preprocessedTrainingData,lgraph,options)
net.Layers
a=dir(["data\traingray\*.png"])
out=size(a,1)
load('net_checkpoint__2700__2021_09_20__22_17_01.mat','net')
checkpointPath = pwd;
options = trainingOptions('adam', ...
    'InitialLearnRate',1e-3, ...
    'MaxEpochs',1, ...
    'Shuffle','every-epoch', ...
    'MiniBatchSize',4, ...
    'VerboseFrequency',1, ...
    'Plots','training-progress', ...
    'ValidationData',valData, ...
    'CheckpointPath',checkpointPath);
net1 = trainNetwork(trainingData,layerGraph(net),options)
Burbax=net;
save Burbax
```

```
testImage=imread('18_gs.jpg');


testImage=imresize(testImage,[256 256])
imwrite(testImage,'wow.png')
testImage=imread('wow.PNG');
imshow(testImage)
C=semanticseg(testImage,net);
B=labeloverlay(testImage,C);
imshow(B)
testImage=imread('predictbuilding.PNG');
testImage=imresize(testImage,[256 256])
imwrite(testImage,'wow1.png')
testImage=imread('wow1.PNG');
imshow(testImage)
C=semanticseg(testImage,net);
B=labeloverlay(testImage,C);
imshow(B)
```

A9: Testing results for the deep learning model

```
cd'D:\All\phdresearch\THESIS\thesis2\matlab_deeplearning\wall_segmentation\data\single_color'
OutputFolder='D:\All\phdresearch\THESIS\thesis2\matlab_deeplearning\wall_segmentation\data\singletest';
dinfo = dir('*.png');% image extension
for K = 1 : length(dinfo)
  thisimage = dinfo(K).name;
  Img   = imread(thisimage);
  Y     = imshow(Img);
  Gray  = rgb2gray(Img);

  imwrite(Gray, fullfile(OutputFolder, thisimage));

end

cd 'D:\All\phd research\THESIS\thesis 2\matlab_deep learning\wall_segmentation'

dataFolder  = fullfile('data');

testImagesDir = fullfile(dataFolder,'singletest');

testLabelsDir = fullfile(dataFolder,'singlemask');

imds = imageDatastore(testImagesDir);

classNames = ["building","background"];

labelIDs   = [255 0];

pxdsTruth = pixelLabelDatastore(testLabelsDir,classNames,labelIDs);

net = load('Burbax9688.mat');

net = net.net;

pxdsResults = semanticseg(imds,net,"WriteLocation",'data\single_output')

Running semantic segmentation network
-------------------------------------
* Processed 10 images.

pxdsResults =

  PixelLabelDatastore with properties:

                       Files: {10×1 cell}
                  ClassNames: {2×1 cell}
                    ReadSize: 1
                     ReadFcn: @readDatastoreImage
    AlternateFileSystemRoots: {}

metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTruth);

Evaluating semantic segmentation results
----------------------------------------
* Selected metrics: global accuracy, class accuracy, IoU, weighted IoU, BF score.
*Processed 10 images.
*Finalizing... Done.
*Data set metrics:

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
```

0.98838	0.98069	0.97053	0.97699	0.92372

A10: Prediction results samples

```
load Burbax9688.mat
testImage=imread('data\single_color\4.PNG');
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\5.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\6.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
```

```
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\7.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\8.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\9.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
```

```
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\10.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\11.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\12.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
```

```
BW = C == 'building';
figure
imshow(BW)
testImage=imread('data\single_color\13.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,Burbax9688);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
```

A11: Binary mask extraction code for lower height buildings

```
import json
import os
import numpy as np
import PIL.Image
import cv2
import matplotlib.pyplot as plt
with open("homeval.json", "r") as read_file:
    data = json.load(read_file)
all_file_names=list(data.keys())
Files_in_directory = []
for root, dirs, files in os.walk("valimggray"):
    for filename in files:
        Files_in_directory.append(filename)
for j in range(len(all_file_names)):
    image_name=data[all_file_names[j]]['filename']
    if image_name in Files_in_directory:
         img = np.asarray(PIL.Image.open('valimggray/'+image_name))
    else:
        continue
    if data[all_file_names[j]]['regions'] != {}:
        #cv2.imwrite('images/%05.0f' % j +'.jpg',img)
        print(j)
        try:
            shape1_x=data[all_file_names[j]]['regions']['0']['shape_attributes']['all_points_x']
            shape1_y=data[all_file_names[j]]['regions']['0']['shape_attributes']['all_points_y']
        except :
            shape1_x=data[all_file_names[j]]['regions'][0]['shape_attributes']['all_points_x']
            shape1_y=data[all_file_names[j]]['regions'][0]['shape_attributes']['all_points_y']
        fig = plt.figure()
        plt.imshow(img.astype(np.uint8))
        plt.scatter(shape1_x,shape1_y,zorder=2,color='red',marker = '.', s= 55)
        ab=np.stack((shape1_x, shape1_y), axis=1)
        img2=cv2.drawContours(img, [ab], -1, (255,255,255), -1)
        mask = np.zeros((img.shape[0],img.shape[1]))
        img3=cv2.drawContours(mask, [ab], -1, 255, -1)
        cv2.imwrite('valimggraymask/%05.0f' % j +'.png',mask.astype(np.uint8))
import os
path = os.chdir("D:\\All\\phd_research\\THESIS\\thesis2\\binary_mask_from_json\\valimggraymask")
i = 13
for file in os.listdir(path):
    new_file_name = "{}.png".format(i)
    os.rename(file, new_file_name)
    i=i+1
import PIL
import os
from PIL import Image
f = r"D:\\All\\phd_research\\THESIS\\thesis2\\binary_mask_from_json\\valimggraymask"
os.listdir(f)
for file in os.listdir(f):
    f_img = f+"/"+file
    img = Image.open(f_img)
    img = img.resize((256,256))
    img.save(f_img)
```

A12: Resizing images to fit the neural network

```
import PIL
import os
from PIL import Image
f = r"D:\\All\\phd_research\\THESIS\\thesis2\\binary_mask_from_json\\valimggray"
os.listdir(f)
for file in os.listdir(f):
    f_img = f+"/"+file
    img = Image.open(f_img)
    img = img.resize((256,256))
    img.save(f_img)
```

A13: Renaming image files to make them manageable in code lines

```
import os
```

```
path =
os.chdir("D:\\All\\phd_research\\THESIS\\thesis2\\binary_mask_from_json\\valimggraymask
")

i = 1

for file in os.listdir(path):

    new_file_name = "{}.png".format(i)

    os.rename(file, new_file_name)

    i=i+1
```

APPENDIX B: Machine learning accuracy results

Table B. 1 Accuracy assessment of the SVM method

OBJECTID	ClassValue	C_0	C_1	Total	U_Accuracy	Kappa
1	C_0	15	0	15	1	0
2	C_1	24	11	35	0.314286	0
3	Total	39	11	50	0	0
4	P_Accuracy	0.384615	1	0	0.52	0
5	Kappa	0	0	0	0	0.215686

Table B. 2 Accuracy assessment of the random trees method

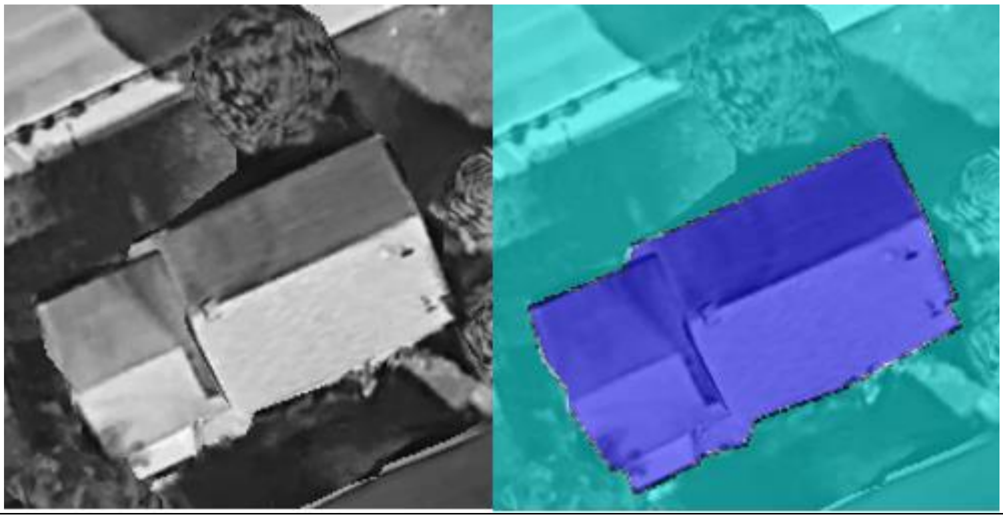
OBJECTID	ClassValue	C_0	C_1	Total	U_Accuracy	Kappa
1	C_0	20	2	22	0.909091	0
2	C_1	17	11	28	0.392857	0
3	Total	37	13	50	0	0
4	P_Accuracy	0.540541	0.846154	0	0.62	0
5	Kappa	0	0	0	0	0.281392

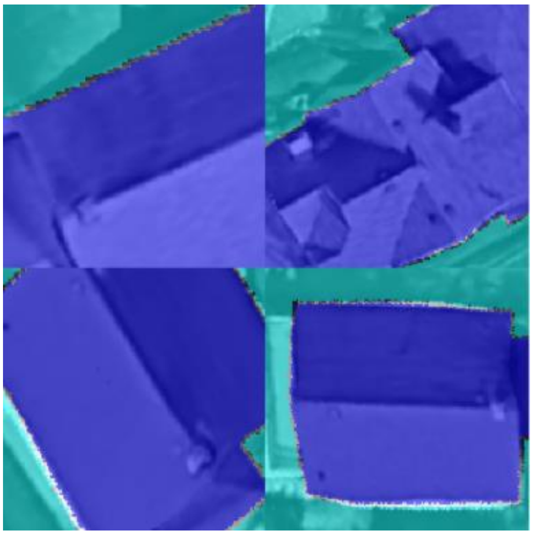
Table B. 3 Accuracy assessment of the maximum likelihood method

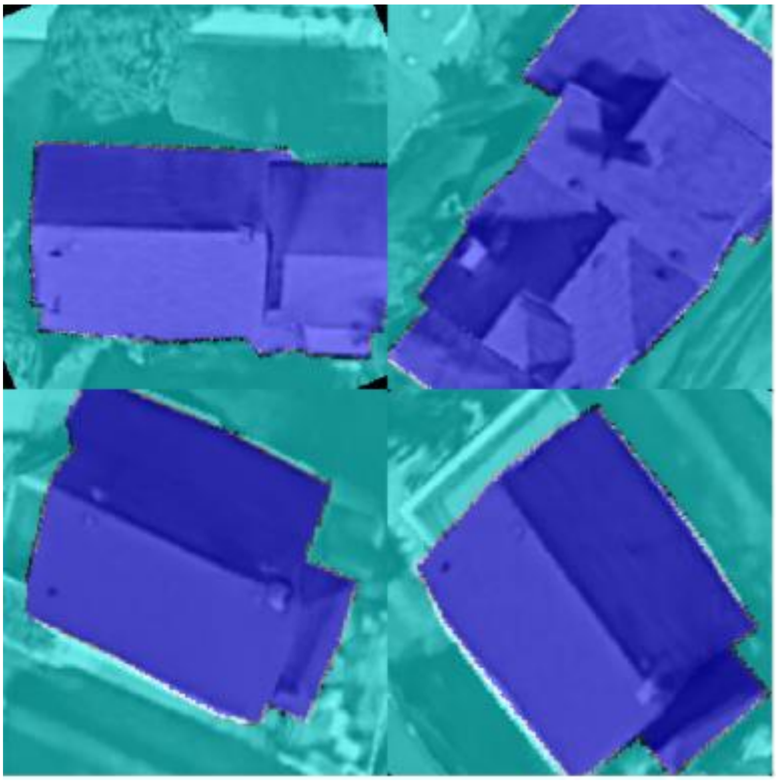
OBJECTID	ClassValue	C_0	C_1	Total	U_Accuracy	Kappa
1	C_0	10	0	10	1	0
2	C_1	30	10	40	0.25	0
3	Total	40	10	50	0	0
4	P_Accuracy	0.25	1	0	0.4	0
5	Kappa	0	0	0	0	0.117647

APPENDIX C: Deep learning algorithm and solar potential analysis miscellaneous details

C1: Training data preparation and data augmentation for lower height buildings







C2: Training workflow for a deep learning model on lower height buildings

```
% when executing a section code always make sure you are in the correct

% directory that the code is trying to execute

OutputFolder = 'D:\All\phd research\THESIS\thesis 2\matlab_deep
learning\home_segmentation\data\traingray';  % Set as needed [EDITED]

dinfo = dir('*.png');% image extension

for K = 1 : length(dinfo)

  thisimage = dinfo(K).name;

  Img   = imread(thisimage);

  Y     = imshow(Img);

  Gray  = rgb2gray(Img);

  imwrite(Gray, fullfile(OutputFolder, thisimage));  % [EDITED]
```

```
end
OutputFolder = 'D:\All\phd research\THESIS\thesis 2\matlab_deep learning\home_segmentation\data\valimggray';  % Set as needed [EDITED]
dinfo = dir('*.png');% image extension
for K = 1 : length(dinfo)
  thisimage = dinfo(K).name;
  Img   = imread(thisimage);
  Y     = imshow(Img);
  Gray  = rgb2gray(Img);
  imwrite(Gray, fullfile(OutputFolder, thisimage));  % [EDITED]
end
dataFolder  = fullfile('data');
imageDir = fullfile(dataFolder,'traingray');
labelDir = fullfile(dataFolder,'train_masks');
numObservations = 4;
trainImages = repelem({imageDir},numObservations,1);
trainLabels = repelem({labelDir},numObservations,1);
classNames = ["building","background"];
labelIDs   = [255 0];
imds = imageDatastore(trainImages);
pxds = pixelLabelDatastore(trainLabels,classNames,labelIDs);
trainingData = combine(imds,pxds);
data = read(trainingData);
I = data{1};
C = data{2};
B = labeloverlay(I,C);
imshow(B)
```

```
augmentedTrainingData = transform(trainingData,@jitterImageColorAndWarp);
data = readall(augmentedTrainingData);
rgb = cell(numObservations,1);
for k = 1:numObservations
    I = data{k,1};
    C = data{k,2};
    rgb{k} = labeloverlay(I,C);
end
montage(rgb)
imageSize = [256 256];
numClasses = 2;
lgraph = unetLayers(imageSize, numClasses)
lgraph.Layers
targetSize= [144 144];
preprocessedTrainingData = transform(augmentedTrainingData,...
    @(data)centerCropImageAndLabel(data,targetSize));
data=readall(preprocessedTrainingData);
rgb = cell(numObservations,1);
for k = 1:numObservations
    I = data{k,1};
    C = data{k,2};
    rgb{k} = labeloverlay(I,C);
end
montage(rgb)
alldata=combine(trainingData,augmentedTrainingData);
val_img="data\valimggray";
val_mask="data\val_masks";
```

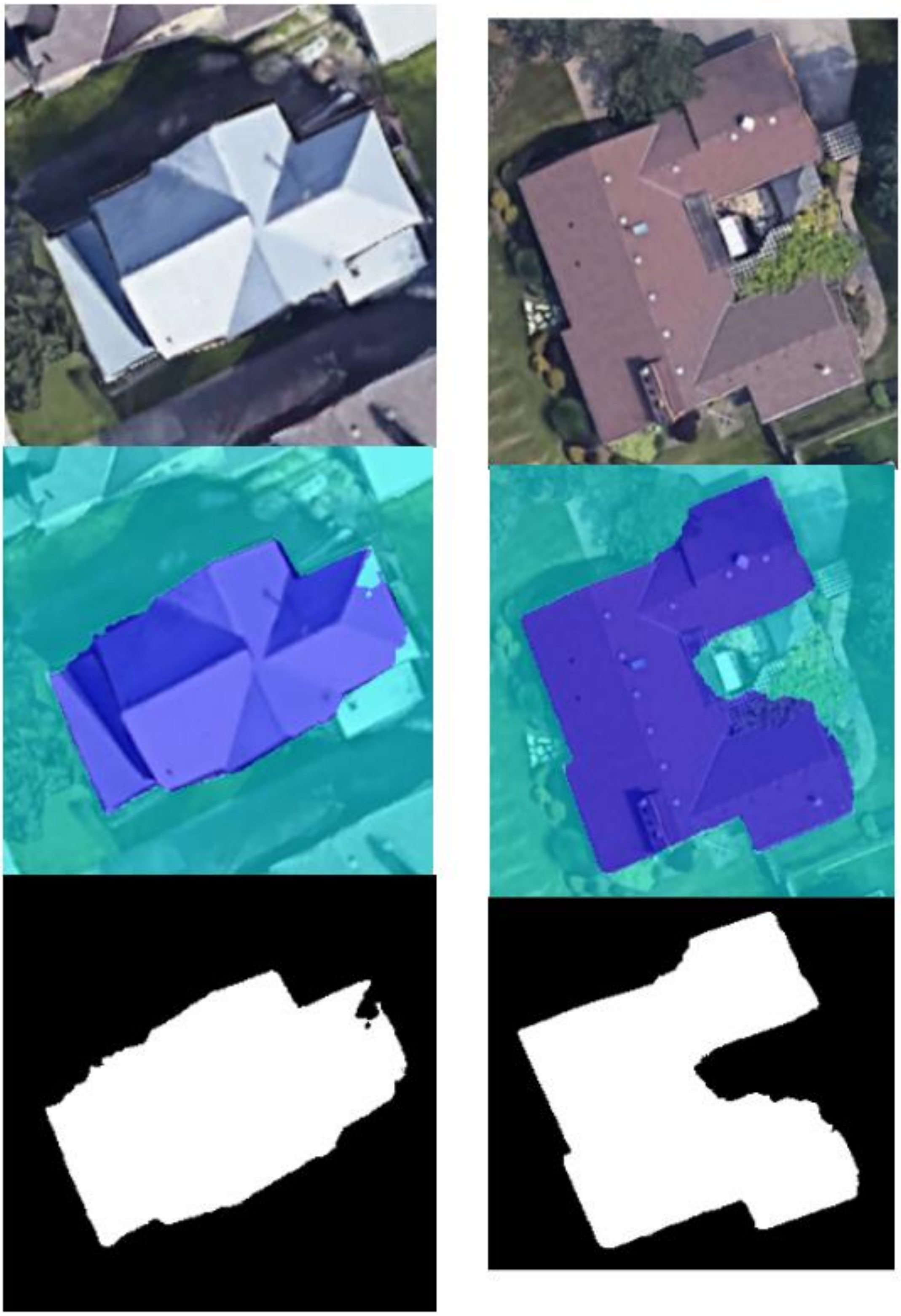
```
valimgids=imageDatastore(val_img);
valclassNames = ["building","background"];
vallabelIDs   = [255 0];
valpxds = pixelLabelDatastore(val_mask,valclassNames,vallabelIDs);
valData = combine(valimgids,valpxds);
checkpointPath = pwd;
options = trainingOptions('adam', ...
    'InitialLearnRate',1e-3, ...
    'MaxEpochs',30, ...
    'Shuffle','every-epoch', ...
    'MiniBatchSize',4, ...
    'VerboseFrequency',50, ...
    'Plots','training-progress', ...
    'ValidationData',valData, ...
    'CheckpointPath',checkpointPath);
net = trainNetwork(alldata,lgraph,options)
```

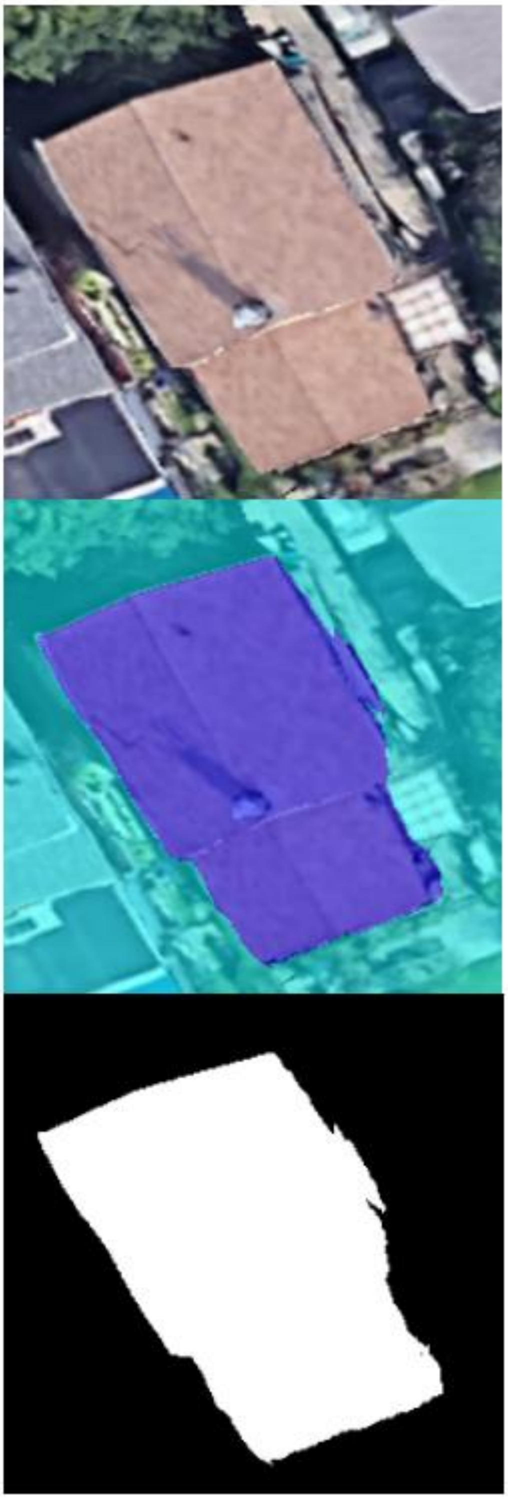
C3: Testing results for sample lower height buildings

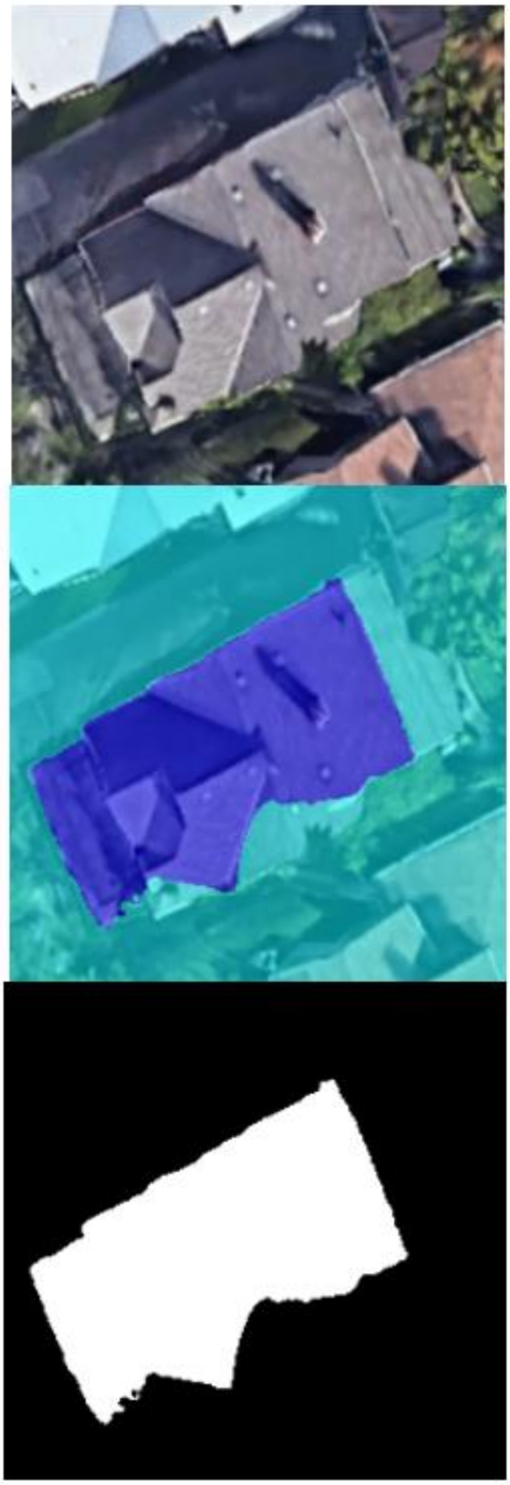
```
load house.mat
testImage=imread('data\single_color\1.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,house);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
```

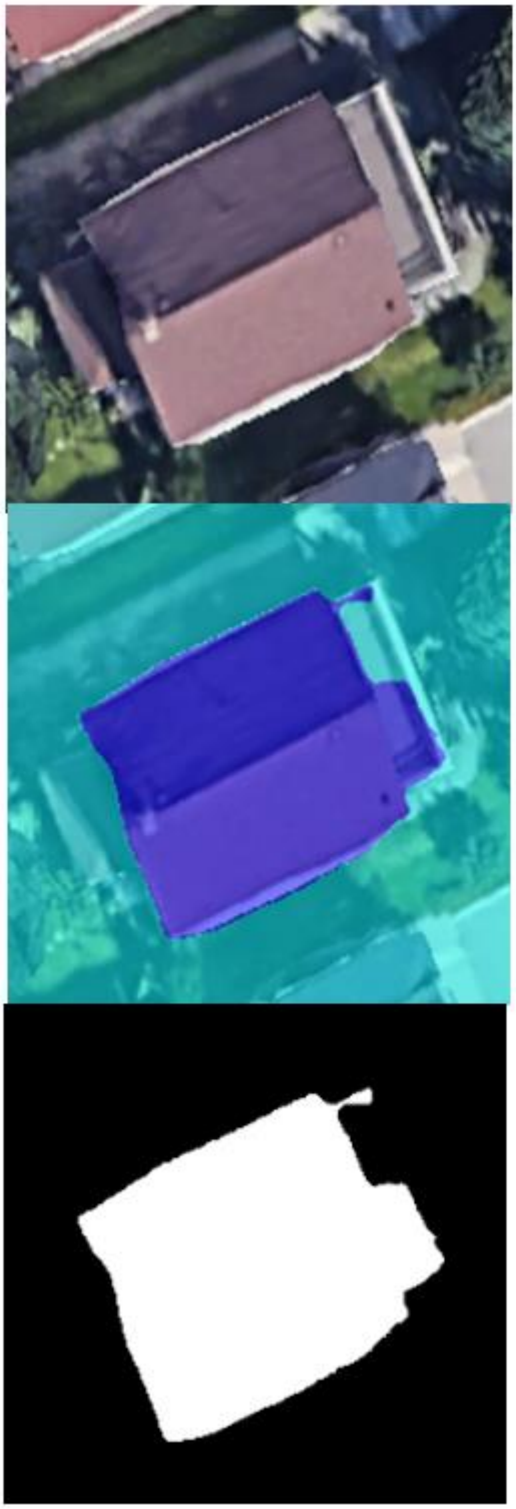
```
figure
imshow(BW)
testImage=imread('data\single_color\2.PNG');
testImage=imresize(testImage,[256 256]);
imshow(testImage)
C=semanticseg(testImage,house);
B=labeloverlay(testImage,C);
imshow(B)
BW = C == 'building';
figure
imshow(BW)
```

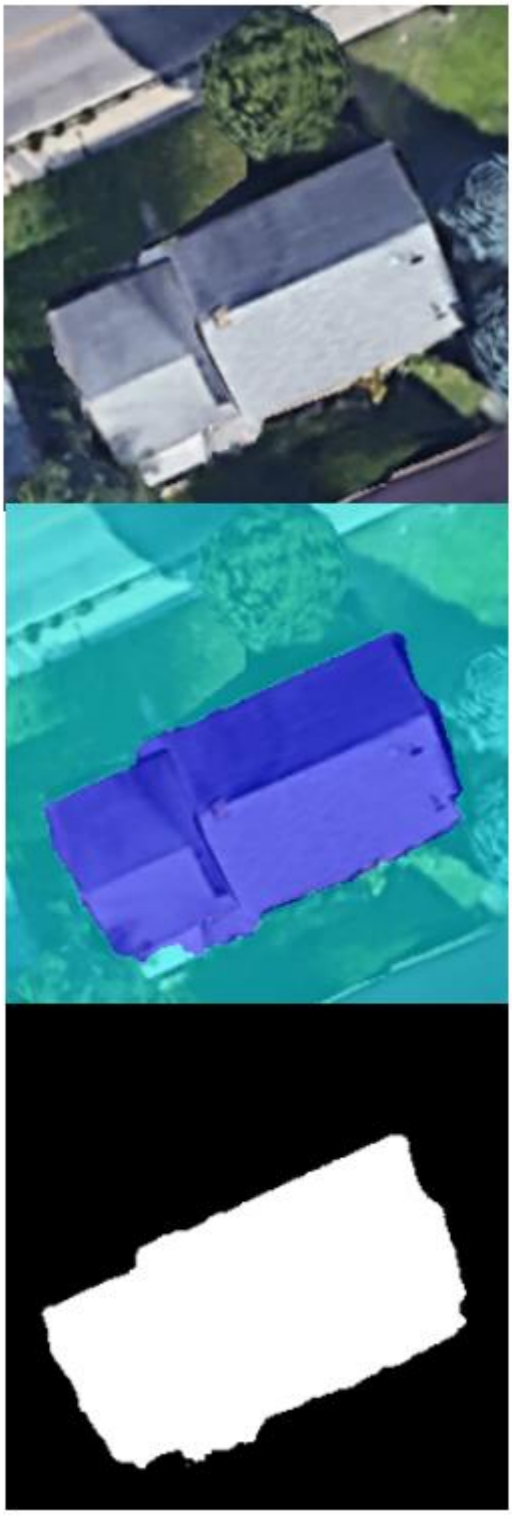
C4: Residential houses footprint deep learning prediction results sampled











C5: Solar radiation mean of the residential houses

BUILDING_ID	COUNT	AREA(m^2)	MEAN (kWh/m^2)
1	266	66.5	988.8369909
2	333	83.25	1056.711608
3	332	83	936.5872871
4	348	87	957.9884345
5	372	93	962.2721126
6	296	74	1042.54831
7	304	76	995.4115508
8	327	81.75	981.9803405
9	326	81.5	952.9742458
10	282	70.5	1062.344189
11	334	83.5	999.5745477
12	340	85	972.8445828
13	302	75.5	1003.151627
14	339	84.75	976.1442025
15	261	65.25	1026.033071
16	397	99.25	1000.459264
17	264	66	1027.970638
18	365	91.25	979.8183973
19	6	1.5	967.7018433
20	302	75.5	1013.077562
21	252	63	1052.572982
22	309	77.25	1025.647193
23	361	90.25	1037.211949
24	388	97	1010.631286
25	371	92.75	978.789964
26	375	93.75	996.0474754
27	348	87	1077.878938
28	374	93.5	1009.786548
29	311	77.75	1013.584016
30	348	87	1026.308742
31	344	86	1040.859261
32	400	100	1015.52737
33	306	76.5	1007.152456
34	357	89.25	1034.993336
35	258	64.5	1012.449771
36	301	75.25	1025.155251
37	342	85.5	1017.833461
38	389	97.25	1023.477873
39	352	88	1011.754621
40	10	2.5	1043.687122
41	327	81.75	986.6344584
42	144	36	959.1864772
43	333	83.25	1012.21599
44	273	68.25	1043.983437
45	320	80	1024.306415
46	1	0.25	1158.915649
47	222	55.5	996.5717806
48	6	1.5	865.3701681
49	255	63.75	941.6319532
50	234	58.5	986.0729641
51	236	59	986.7512414

52	80	20	949.279628
53	251	62.75	959.0423051
54	293	73.25	1056.779065
55	279	69.75	995.1344701
56	4	1	842.4700317
57	156	39	956.7949622
58	403	100.75	957.8380559
59	51	12.75	910.0591897
60	285	71.25	1073.400558
61	382	95.5	954.7063557
62	241	60.25	955.9124057
63	368	92	1055.080315
64	43	10.75	882.6895468
65	320	80	1006.742312
66	298	74.5	981.2009185
67	151	37.75	999.5060085
68	345	86.25	968.5060869
69	79	19.75	981.3067944
70	91	22.75	986.2216354
71	9	2.25	914.1350505
72	439	109.75	912.2813929
73	108	27	980.3053905
74	247	61.75	1060.924001
75	312	78	943.2689238
76	299	74.75	1023.421744
77	278	69.5	1010.332966
78	239	59.75	1054.122962
79	344	86	1002.849332
80	4	1	942.9744415
81	323	80.75	1003.971871
82	209	52.25	1033.088993
83	333	83.25	943.5936109
84	33	8.25	898.5093735
85	157	39.25	1055.452286
86	303	75.75	1000.448003
87	152	38	926.6145064
88	2	0.5	829.006012
89	169	42.25	1004.646621
90	288	72	1013.349299
91	372	93	1020.290147
92	76	19	934.6854842
93	558	139.5	1056.123254
94	13	3.25	907.9842999
95	13	3.25	1082.355952
96	7	1.75	988.3387102
97	81	20.25	935.5186963
98	516	129	1027.106713
99	1	0.25	846.2277222
100	2	0.5	886.6378784
101	1	0.25	1075.067505
102	366	91.5	1058.682854
103	198	49.5	953.6445655
104	324	81	963.5123809
105	164	41	1023.186926

106	57	14.25	885.8026637
107	703	175.75	1038.960867
108	616	154	1026.252602
109	279	69.75	1010.609367
110	5	1.25	970.4407471
111	7	1.75	933.6768101
112	6	1.5	996.659729
113	2	0.5	926.555542
114	249	62.25	1032.243668
115	1	0.25	826.8411255
116	267	66.75	972.4266504
117	550	137.5	1024.554231
118	222	55.5	946.017089
119	3	0.75	1056.70933
120	230	57.5	1006.089633
121	13	3.25	855.0880925
122	52	13	1026.906413
123	201	50.25	1035.841791
124	111	27.75	934.2385259
125	193	48.25	966.4478216
126	14	3.5	927.5419399
127	17	4.25	942.364624
128	98	24.5	893.3508182
129	212	53	915.7102627
130	238	59.5	1007.090441
131	5	1.25	1130.16615
132	182	45.5	955.2526815
133	116	29	973.2285567
134	5	1.25	914.3980469
135	20	5	1027.827664
136	3	0.75	968.8831991
137	377	94.25	1033.507919
138	272	68	977.2423477
139	29	7.25	884.5816377
140	17	4.25	996.020971
141	5	1.25	990.1759033
142	12	3	899.9884338
143	10	2.5	1042.437872
144	213	53.25	1031.194427
145	3	0.75	1007.688619
146	13	3.25	992.7176326
147	44	11	862.8684845
148	358	89.5	994.78858
149	177	44.25	996.6053474
150	129	32.25	1046.684374
151	5	1.25	862.6032104
152	195	48.75	1025.795829
153	211	52.75	957.5626429
154	84	21	939.2804493
155	26	6.5	884.1019804
156	1	0.25	972.9595337
157	130	32.5	912.9184519
158	164	41	987.7323642
159	129	32.25	1042.636638

160	192	48	994.1622483
161	146	36.5	1004.062704
162	2	0.5	1077.397308
163	137	34.25	944.3491697
164	68	17	1001.426736
165	95	23.75	1113.841627
166	1	0.25	966.9815674
167	161	40.25	1062.08104
168	177	44.25	979.5981997
169	3	0.75	878.0691325
170	125	31.25	982.2826777
171	1	0.25	840.0734253
172	82	20.5	1010.213207
173	84	21	949.1892133
174	415	103.75	1002.996472
175	425	106.25	1001.306811
176	305	76.25	1009.428563
177	149	37.25	1013.134115
178	352	88	1015.590036
179	322	80.5	1020.649554
180	350	87.5	988.1620785
181	126	31.5	1001.108539
182	9	2.25	998.0908
183	217	54.25	992.5074044
184	430	107.5	1023.022682
185	170	42.5	954.3976756
186	9	2.25	934.9113702
187	241	60.25	1003.627402
188	197	49.25	971.3191203
189	333	83.25	1024.185583
190	372	93	963.5298564
191	96	24	962.0187225
192	495	123.75	985.7198386
193	127	31.75	1036.438488
194	269	67.25	1074.069491
195	400	100	989.6771736
196	336	84	1023.695968
197	468	117	1005.955873
198	331	82.75	1032.372832
199	232	58	1028.373936
200	308	77	1006.175679
201	9	2.25	956.6376885
202	261	65.25	928.5789301
203	258	64.5	1010.483595
204	321	80.25	988.8846816
205	96	24	1071.673859
206	314	78.5	1012.624152
207	351	87.75	1010.19283
208	317	79.25	964.4030563
209	1	0.25	922.9816284
210	632	158	1048.072157
211	4	1	825.6678314
212	157	39.25	1010.190968
213	347	86.75	1070.550078

214	357	89.25	968.2890316
215	278	69.5	970.4394621
216	276	69	935.0516077
217	216	54	968.4878331
218	323	80.75	1082.675257
219	423	105.75	1058.726439
220	3	0.75	1100.092407
221	388	97	1039.658036
222	276	69	980.8422321
223	269	67.25	1026.07255
224	402	100.5	1059.223017
225	4	1	916.616272
226	259	64.75	1028.137894
227	307	76.75	1084.754839
228	108	27	964.7623285
229	12	3	953.4170481
230	217	54.25	950.4575749
231	172	43	946.1943743
232	259	64.75	1040.316811
233	273	68.25	1054.302935
234	12	3	1074.232183
235	9	2.25	985.4574653
236	311	77.75	1056.89311
237	273	68.25	1020.034446
238	268	67	1050.439585
239	338	84.5	981.0788861
240	3	0.75	1108.588867
241	352	88	1028.933488
242	14	3.5	1019.880253
243	4	1	968.2856445
244	225	56.25	980.2949368
245	307	76.75	1041.274696
246	274	68.5	1037.618757
247	345	86.25	978.2957371
248	206	51.5	976.1482215
249	7	1.75	931.1154349
250	1	0.25	841.4387207
251	15	3.75	917.9375936
252	2	0.5	1076.01178
253	114	28.5	920.5690902
254	249	62.25	1078.244036
255	8	2	915.8704224
256	7	1.75	886.2780762
257	4	1	933.4315033
258	37	9.25	886.4593753
259	297	74.25	992.1429053
260	287	71.75	970.42896
261	172	43	976.9168424
262	125	31.25	982.5156553
263	389	97.25	996.8104372
264	42	10.5	1029.248324
265	1	0.25	857.9434814
266	4	1	916.0022278
267	272	68	987.9665453

268	287	71.75	1027.774037
269	1	0.25	865.6409912
270	312	78	1020.644312
271	294	73.5	1017.365283
272	77	19.25	985.3631719
273	2	0.5	871.5733337
274	6	1.5	997.3743083
275	250	62.5	1080.220816
276	174	43.5	974.3534307
277	218	54.5	1030.441443
278	45	11.25	917.7486152
279	1	0.25	901.1488647
280	351	87.75	1064.265117
281	22	5.5	927.2997825
282	262	65.5	989.4158139
283	197	49.25	1044.722153
284	263	65.75	981.2891247
285	85	21.25	920.3993193
286	8	2	928.1192017
287	387	96.75	1038.825817
288	207	51.75	1032.77275
289	142	35.5	1036.19313
290	219	54.75	1010.080933
291	265	66.25	996.004974
292	213	53.25	1024.971029
293	231	57.75	1053.652138
294	11	2.75	1003.702354
295	252	63	1032.533827
296	37	9.25	939.7830464
297	4	1	920.1133575
298	593	148.25	1037.505736
299	5	1.25	897.2580322
300	236	59	1033.97086
301	181	45.25	976.4362169
302	118	29.5	1048.216091

Curriculum Vitae

Name: Tewodros Alemayehu

Education

Western University

London, ON, Canada
2018-2023 Ph.D.

Leadstar Univsersity College

Addis Ababa, Ethiopia
2014-2016 MBA

Adama University

Nazreth, Ethiopia
2012-2015 M.Sc.

Hawasssa University

Hawassa, Ethiopia
2006-2010 B.Sc.

Related Work Experience

Research Assistant
Western University - London, ON
2018-2023

Assistant Analyst
Science & Technology Information Center

Addis Ababa, Ethiopia
2013-2018

Publications

Alemayehu, T.F. and Bitsuamlak, G.T., 2022. Autonomous urban topology generation for urban flow modelling. Sustainable Cities and Society, 87, p.104181.