

# **Point Cloud-based Deep Learning and UAV Path Planning for Surface Defect Detection of Concrete Bridges**

**Neshat Bolourian**

A Thesis  
In the Department  
of  
Building, Civil, and Environmental Engineering

Presented in Partial Fulfillment of the Requirements  
For the Degree of  
Doctor of Philosophy (Building, Civil, and Environmental Engineering) at  
Concordia University  
Montreal, Quebec, Canada

June 2022

**© Neshat Bolourian, 2022**

**CONCORDIA UNIVERSITY  
SCHOOL OF GRADUATE STUDIES**

This is to certify that the thesis prepared

By: *Neshat Bolourian*

Entitled: *Point Cloud-based Deep Learning and UAV Path Planning for Surface Defect Detection of Concrete Bridges*

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (*Building, Civil, and Environmental Engineering*)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____	Chair
<i>Dr. Sebastien Le Beux</i>	
_____	External Examiner
<i>Dr. Yelda Turkan</i>	
_____	External to Program
<i>Dr. Gerald J. Gouw</i>	
_____	Examiner
<i>Dr. Fuzhan Nasiri</i>	
_____	Examiner
<i>Dr. Ashutosh Bagchi</i>	
_____	Thesis Supervisor
<i>Dr. Amin Hammad</i>	

Approved by \_\_\_\_\_  
*Dr. Mazdak Nik-Bakht, Graduate Program Director*

Month/day/2022 \_\_\_\_\_  
*Dr. Mourad Debbabi, Dean, Gina Cody School of Engineering and Computer Science*

# ABSTRACT

## Point Cloud-based Deep Learning and UAV Path Planning for Surface Defect Detection of Concrete Bridges

Neshat Bolourian, Ph.D.  
Concordia University, 2022

Over the past decades, several bridges have collapsed, causing many losses due to the lack of proper monitoring and inspection. Although several new techniques have been developed to detect bridge defects, annual visual inspection remains the main approach. Visual inspection, using naked eyes, is time-consuming and subjective because of human errors. Light Detection and Ranging (LiDAR) scanning is a new technology to collect 3D point clouds. The main strength of point clouds over 2D images is collecting the third dimension of the scanned objects. Deep Learning (DL)-based methods have attracted the researchers' attention for concrete surface defect detection. However, no point cloud-based DL method is currently available for semantic segmentation of bridge surface defects without converting the raw point cloud dataset into other representations, which results in increasing the size of the dataset and leads to some challenges regarding storage capacity, cost, and training time. Some promising point cloud-based semantic segmentation methods (i.e., PointNet and PointNet++) have been applied in segmenting bridge components (i.e., slabs, piers), but not for segmenting surface defects (i.e., cracks, spalls). Moreover, most of the current point cloud-based concrete surface defect detection methods focus on only one type of defects. On the other hand, in DL, a dataset plays a key role in terms of variety, diversity, accuracy, and size. The lack of publicly available point cloud datasets for bridge surface defects is one of the reasons of the lack of studies in the area of point cloud-based methods.

Furthermore, compared with terrestrial LiDAR scanning, LiDAR-equipped Unmanned Aerial Vehicle (UAV) is capable of scanning the inaccessible surfaces of the bridges at a closer distance with higher safety. Although the UAV flying path can be controlled using remote controllers, automating and optimizing UAV path planning is preferable for being able to trace a collision-free path with minimum flight time. To increase the efficiency and accuracy of this approach, it is crucial to scan all parts of the bridge with a near perpendicular view. However, in the case of obstacle existence (e.g., bridge piers), achieving full coverage with near perpendicular view may not be possible. To provide more accurate results, using overlapping views is recommended. However, this method could result in increasing the inspection cost and time. Therefore, overlapping views should be considered only for surface areas where defects are expected.

Addressing the above issues, this research aims to: (1) create a publicly available point cloud dataset for concrete bridge surface defect semantic segmentation, (2) develop a point cloud-based semantic segmentation DL method to detect different types of concrete surface defects, and (3) propose a novel near-optimal path planning method for LiDAR-equipped UAV with respect to the minimum path length and maximum coverage considering the potential locations of defects.

On this premise, a point cloud-based DL method for semantic segmentation of concrete bridge surface defects (i.e., cracks and spalls), called SNEPointNet++, is developed. To have a network with high-performance, SNEPointNet++ focuses on two main characteristics related to surface defects (i.e., normal vector and depth) and takes into account the issues related to the point cloud dataset (i.e., small size and imbalanced dataset). Sensitivity analysis is applied to capture the best

combination of hyperparameters and investigate their effects on network performance. The dataset, which was collected from four concrete bridges, was annotated, augmented, and classified into three classes: cracks, spalls, and no defect. This dataset is made available for other researchers. The model was trained and evaluated using 60% and 20% of the dataset, respectively. Testing on the remaining part of the dataset resulted in 93% recall (69% IoU) and 92% recall (82.5% IoU) for cracks and spalls, respectively. Moreover, the results show that the spalls of the segments deeper than 7 cm (severe spalls) can be detected with 99% recall.

On the other hand, this research proposes a 3D path planning method for using a UAV equipped with a LiDAR for bridge inspection to have efficient data collection. The method integrates a Genetic Algorithm (GA) and A\* algorithm to solve the Traveling Salesman Problem (TSP), considering the potential locations of bridge surface defects such as cracks. The objective is to minimize the time of flight while achieving maximum visibility. The method provides the potential locations of surface defects to efficiently achieve perpendicular and overlapping views for sampling the viewpoints. Calculating the visibility with respect to the level of criticality leads to giving the priority to covering the areas with higher risk levels. Applying the proposed method on a 3-span bridge in Alberta, the results reveal that considering overlapping views based on the level of criticality of the zones and perpendicular views for all viewpoints leads to accurate and time-efficient data collection.

# ACKNOWLEDGEMENT

My great appreciation goes to my supervisor, Dr. Amin Hammad, for his patience and support in overcoming numerous obstacles I have been facing through my research. His guidance, advice and criticism was my most valuable asset during my studies.

I would like to thank the members of my thesis committee, Dr. Yelda Turkan, Dr. Ashutosh Bagchi, Dr. Fuzhan Nasiri, and Dr. Gerald Gouw for spending their valuable time for reading my thesis.

I would like to acknowledge the contributions of Mr. Majid Nasrollahi for implementing adapted PointNet++ and point cloud data collection. His enthusiasm in conducting research was a great asset in our contribution.

I gratefully appreciate my friend, Mr. Ali Ghelmani, who helped me through some challenges in implementing defect semantic segmentation using his knowledge about programming and computer vision.

I am thankful to Mr. Mohammad Akbarzadeh, Dr. Khaled El-Ammari, and Dr. Fariddodin Vahdatikhaki for their valuable comments and sharing their knowledge as much as they could. I would, also, like to thank Dr. Ameen Albahri for introducing Unity which was used in path planning implementation.

I would like to thank my husband, Roozbeh Talebi, my parents, Simin Izadian and Akbar Bolourian, and my sister, Nafis Bolourian, for their spiritual support and unconditional love in my life. I am also grateful to my friend, Dr. Behrang Talebi, for encouraging me to start my Ph.D. study and helping me through it. I would like to thank my uncle, Dr. Jalal Izadian, for all his positivity and guidance.

Last but not least, I am thankful to all my colleagues and friends, especially Ms. Leila Rafati, Dr. Negar Salimzadeh, and Dr. Shide Salimi, who filled this chapter of my life with a lot of joy and incredible memories.

*Dedicated to my lovely family and my beloved niece, Delara.*

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS.....</b>	<b>vii</b>
<b>LIST OF FIGURES.....</b>	<b>ix</b>
<b>LIST OF TABLES.....</b>	<b>xi</b>
<b>LIST OF ABBREVIATIONS.....</b>	<b>xii</b>
<b>LIST OF SYMBOLS.....</b>	<b>xiv</b>
<b>CHAPTER 1 INTRODUCTION.....</b>	<b>1</b>
1.1 Background.....	1
1.2 Problem Statement.....	2
1.3 Research Objectives.....	4
1.4 Overview of the Research Methodology.....	4
1.5 Thesis Organization.....	6
<b>CHAPTER 2 LITERATURE REVIEW.....</b>	<b>7</b>
2.1 Introduction.....	7
2.2 Bridge Inspection.....	7
2.3 Visual Inspection Methods.....	9
2.3.1 Camera-based Methods.....	9
2.3.2 LiDAR-based Methods.....	9
2.3.3 Comparison between Camera- and LiDAR-based Methods.....	9
2.4 LiDAR Scanning Platforms.....	10
2.5 Defect Detection Using Point Cloud and ML.....	11
2.5.1 Machine Learning and Deep Learning.....	12
2.5.2 Point Cloud Datasets.....	13
2.5.3 Point Cloud-based Semantic Segmentation Using DL.....	14
2.5.4 Concrete Surface Defect Detection Using Point Cloud.....	21
2.5.5 Comparison between Image and Point Cloud-based Defect Semantic Segmentation	28
2.6 UAV Path Planning.....	28
2.6.1 Obstacle-free Path Planning Algorithms.....	30
2.6.2 Coverage Path Planning Algorithms.....	34
2.6.3 Traveling Salesman Problem.....	35
2.6.4 UAV Path Planning for Inspection.....	37
2.7 Summary.....	43
<b>CHAPTER 3 POINT CLOUD-BASED CONCRETE SURFACE DEFECT SEMANTIC SEGMENTATION.....</b>	<b>44</b>

3.1	Introduction.....	44
3.2	Proposed Method.....	44
3.2.1	Aspects Considered in the Method.....	44
3.2.2	Performance Metrics.....	51
3.2.3	Framework of Adjustments Made to PointNet++.....	52
3.3	Implementation and Case Study.....	52
3.3.1	Data Collection.....	54
3.3.2	Data Preparation.....	56
3.3.3	Training and Testing.....	60
3.4	Discussion.....	75
3.4.1	Training, Evaluations, and Testing Results.....	75
3.4.2	Testing Results Based on Segment Depth.....	78
3.4.3	Comparison with Image-Based Methods.....	78
3.4.4	Comparison with Point Cloud-Based Methods.....	79
3.5	Summary and Conclusions.....	81
<b>CHAPTER 4 LIDAR-EQUIPPED UAV PATH PLANNING CONSIDERING POTENTIAL LOCATIONS OF DEFECTS FOR BRIDGE INSPECTION.....</b>		<b>83</b>
4.1	Introduction.....	83
4.2	Considerations, Requirements, and Constraints.....	83
4.3	Developing Path Planning Considering Potential Locations of Defects.....	85
4.3.1	Assign IVs to Cells.....	88
4.3.2	Select Initial VPIs.....	90
4.3.3	Calculate the Optimum Path.....	93
4.4	Implementation and Case Study.....	95
4.5	Conclusions.....	100
<b>CHAPTER 5 SUMMARY, CONTRIBUTIONS, AND FUTURE WORK.....</b>		<b>101</b>
5.1	Summary of Research.....	101
5.2	Research Contributions and Conclusions.....	101
5.3	Limitations and Future Work.....	102
<b>REFERENCES.....</b>		<b>104</b>
<b>APPENDIX A. DATA AUGMENTATION.....</b>		<b>117</b>
<b>APPENDIX B. DATA ARRANGEMENT.....</b>		<b>121</b>
<b>APPENDIX C. APPLICATION FOR FLIGHT OPERATION CERTIFICATION.....</b>		<b>124</b>
<b>APPENDIX D. LIST OF PUBLICATIONS.....</b>		<b>135</b>



## LIST OF FIGURES

Figure 1-1. Using a cherry picker to inspect a bridge (Facelift Ltd., 2017).	1
Figure 1-2. Failure of a cherry picker during under-bridge inspection (The Vertikal Press, 2015).	1
Figure 1-3. Example of highway bridges (Montreal Turcot Interchange) (FOTOimage Montreal, 2017).	2
Figure 1-4. Overview of research methodology.	5
Figure 2-1. Level of performance vs. time relationship for various levels of maintenance (Mirza, 2006).	7
Figure 2-2. Visual inspection using naked eyes (Ma and Sacks, 2016).	9
Figure 2-3. Terrestrial laser scanning and mobile laser scanning.	10
Figure 2-4. Airborne laser scanning.	11
Figure 2-5. Binary classification using ReLU (Montufer et al., 2014)	13
Figure 2-6. PointNet architecture (Qi et al., 2017a).	15
Figure 2-7. PointNet++ architecture for single scale point grouping (Qi et al., 2017b).	16
Figure 2-8. (a) Multi-scale grouping (MSG); (b) Multiresolution grouping (MRG) (Qi et al., 2017b)	16
Figure 2-9. Adapted PointNet for defect detection (Nasrollahi et al., 2019).	17
Figure 2-10. (a) Overview of scan points, (b) angle deviation, and (c) distance deviation (Kim et al., 2015).	22
Figure 2-11. Framework for bridge inspection using TLS (Truong-Hong et al., 2016).	23
Figure 2-12. Crack detection at an abutment of Lungsdorf Bridge (Truong-Hong et al., 2016).	23
Figure 2-13. Sample of crack measurement (Valenca et al., 2017).	24
Figure 2-14. General process of UAV model-based path planning method.	29
Figure 2-15. Example of Bug 1 algorithm (Lumelsky and Stepanov, 1987).	30
Figure 2-16. The schematic diagram of the A* algorithm path search (Fu et al., 2018).	31
Figure 2-17. Pseudocode of RRT algorithm.	32
Figure 2-18. Extended operation (Kuffner and LaValle, 2000).	32
Figure 2-22. Dual RRT (LaValle and Kuffner Jr, 2001).	32
Figure 2-20. Near neighbor search and rewiring operations in RRT* (Noreen et al., 2016).	33
Figure 2-21. Pseudocode of RRT* algorithm (Nasir et al., 2013).	34
Figure 2-22. Coverage path planning using wavefront algorithm (Galceran and Carreras, 2013).	35
Figure 2-23. Coverage path planning using spanning trees algorithm (Galceran and Carreras, 2013).	35
Figure 2-24. Framework of a PSO algorithm (Goldberg et al., 2006).	37
Figure 2-25. Finding boundaries of the sampling space (Bircher et al., 2015).	38
Figure 2-26. Path planning using LKH and RRT* (Bircher et al., 2015).	38
Figure 2-30. Generating VPIs (Phung et al., 2017a).	39
Figure 2-28. Sampling viewpoints (Cao et al., 2020).	39
Figure 2-29. Path planning of a bridge (Cao et al., 2020).	40
Figure 3-1. Cross-section of a sample defect with the normal vector and depth of a point.	45
Figure 3-2. Sample of blocks of points in a segment.	46
Figure 3-3. Defining the values for sensitivity analysis for number of points per block and block size.	48
Figure 3-4. Sample of stride.	48

Figure 3-5. Sensitivity analysis framework. ....	50
Figure 3-6. Example of TP, TN, FP, and FN for the crack class. ....	51
Figure 3-7. Overall framework of the proposed defect semantic segmentation ....	52
Figure 3-8. Implementation and case study steps of adapted PointNet++ and SNEPointNet++. .	53
Figure 3-9. Data collection locations in Montreal. ....	55
Figure 3-10. Samples of scanning positions on the western side of Bridge 1. ....	56
Figure 3-11. Example of segmentation and annotation processes. ....	57
Figure 3-12. Sample of the fitted segment. ....	58
Figure 3-13. Distribution of dataset segments based on density. ....	59
Figure 3-14. Architecture of adapted PointNet++ of the performance. ....	63
Figure 3-15. Testing results and training time of cases A1-A9. ....	65
Figure 3-16. Effect of stride and number of points on crack semantic segmentation using adapted PointNet++. ....	66
Figure 3-17. Effect of stride and number of points on spall semantic segmentation using adapted PointNet++. ....	67
Figure 3-18. Architecture of SNEPointNet++ with the best performance (Case M4). ....	70
Figure 3-19. Effect of stride on crack semantic segmentation using SNEPointNet++. ....	73
Figure 3-20. Effect of stride on spall semantic segmentation using SNEPointNet++. ....	74
Figure 3-21. Three samples of semantic segmentation. ....	77
Figure 3-22. Classifying testing results based on depth of segments. ....	78
Figure 4-1. Scanner position on top of the UAV in (a) and under (b). ....	83
Figure 4-2. Six degrees of freedom of a UAV (Getbestcopter, 2016). ....	84
Figure 4-3. LiDAR specifications. ....	85
Figure 4-4. Proposed framework for path planning. ....	87
Figure 4-5. Calculating minimum cell size. ....	88
Figure 4-6. The meshed bridge surface. ....	88
Figure 4-7. Determining the level of criticality of each zone. ....	89
Figure 4-8. Assigning IVs to cells. ....	90
Figure 4-9. Scanning defects. ....	91
Figure 4-10. Scanning defects in case of an obstacle. ....	91
Figure 4-11. Maximum visible area from VPI located at minimum distance from surface. ....	92
Figure 4-12. Selecting VPIs based on criticality levels. ....	93
Figure 4-13. Path length matrix calculation pseudocode. ....	94
Figure 4-14. Inspected bridge in Alberta, Canada. ....	95
Figure 4-15. Relation between flight time and payload for Matrix 100. ....	95
Figure 4-16. Modeled bridge structure. ....	96
Figure 4-17. Results of bridge structural analysis. ....	96
Figure 4-18. Risk zones and sampled VPIs (bottom view of bridge). ....	97
Figure 4-19. Importing the model into Unity. ....	97
Figure 4-20. Meshed surface of the bridge. ....	97
Figure 4-21. Four optimization results. ....	98
Figure 4-22. Visual representation for the calculated path. ....	98
Figure 4-23. Percentage of cells in each zone at any given incidence angle. ....	99
Figure B-1. A segment of a concrete bridge including defects in Cloud Compare. ....	121
Figure B-2. Arrangement of dataset. ....	122

## LIST OF TABLES

Table 2-1. Defects severity based on depth (d), width (w), and height (h) of the defect (Ontario Ministry of Transportation, 2008) .....	8
Table 2-2. Examples of 3D datasets for semantic segmentation (Garcia-Garcia et al., 2017). ....	14
Table 2-3. Usage of point cloud-based methods in inspection (construction industry). ....	19
Table 2-4. Summary of the most related point cloud-based concrete surface defect detection studies. ....	26
Table 2-5. Image-based semantic segmentation of surface defect using DL. ....	28
Table 2-6. Summary of the literature review of UAV path planning for inspection. ....	41
Table 3-1. FARO Focus3D LiDAR specifications (FARO Technologies Inc., 2011). ....	54
Table 3-2. Scanning information. ....	56
Table 3-3. Statistics of the annotated dataset before augmentation. ....	59
Table 3-4. List of number of points per block, block sizes, and their densities. ....	61
Table 3-5. Training and evaluation results of adapted PointNet++ for various stride sizes and numbers of points per block. ....	64
Table 3-6. Testing results of adapted PointNet++ for various stride sizes and numbers of points per block. ....	64
Table 3-7. Hyperparameter values in each round of sensitivity analysis. ....	68
Table 3-8. Training and evaluation results of the top five combinations of sublayers and sampling sizes. ....	69
Table 3-9. Testing results of the top five combinations of sublayers and sampling sizes. ....	69
Table 3-10. Testing results for different block sizes. ....	71
Table 3-11. Testing results for different numbers of points per block. ....	71
Table 3-12. Testing results for various stride sizes and numbers of points per block. ....	72
Table 3-13. Comparison between PointNet++ and adjusted networks (best configurations). ....	76
Table 3-14. Training and evaluation results. ....	76
Table 3-15. Testing results. ....	76
Table 3-16. Classifying the testing results based on the segment depth. ....	78
Table 3-17. Comparison between SNEPointNet++ and image-based DL semantic segmentation methods. ....	79
Table 3-18. Comparison between SNEPointNet++ and similar point cloud-based methods. ....	81
Table 4-1. Determine three ranges for shear and bending moment. ....	89
Table 4-2. Takeoff weight and flight time for Matrix 100. ....	95
Table 4-3. Path planning results for three cases. ....	100

## LIST OF ABBREVIATIONS

2D	2-Dimensional
3D	3-Dimensional
4D	4-Dimensional
AASHTO	American Association of State Highway and Transportation Officials
ACO	Ant Colony Optimization
ADAM	Adaptive Moment Optimization
AGP	Art Gallery Problem
AI	Artificial Intelligence
ALS	Airborne Laser Scanning
BIM	Building Information Model
BSMP	Bridge Safety Management Program
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
CSA	Canadian Standards Association
DGCNN	Dynamic Graph Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DoF	Degree of Freedom
FC	Fully Connected
FBX	Film Box
FHWA	Federal Highway Administration
FN	False Negative
FoV	Field of View
FoV <sub>H</sub>	Horizontal Field of View
FoV <sub>V</sub>	Vertical Field of View
FP	False Positive
FPPM	Fast Path Planning Method
GA	Genetic Algorithm
GPS	Global Positioning System
HDF	Hierarchical Data Format
IFC	International Foundation Class
IoU	Intersection over Union
IV	Importance Value
LiDAR	Light Detection and Ranging
LKH	Lin-Kernighan Heuristic
mIoU	mean Intersection over Union
ML	Machine Learning
MLP	Multilayer Perceptrons
MLS	Mobile Laser Scanning

MVCNN	Multi-View Convolutional Neural Network
NBIS	National Bridge Inspection Standards
NDT	Non-Destructive Testing
NP	Nondeterministic Polynomial
OA	Overall Accuracy
OM	Operation and Maintenance
OSIM	Ontario Structure Inspection Manual
PS	Phase-Shift
PSO	Particle Swan Optimization
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RMSProp	Root Mean Square Propagation
RRT	Rapidly-exploring Random Tree
RRT*	Rapidly-exploring Random Tree Star
SLAM	Simultaneous Localization And Mapping
SNEPointNet++	Surface Normal Enhanced PointNet++
TLS	Terrestrial Laser Scanning
TN	True Negative
ToF	Time-of-Flight
TP	True Positive
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
VPI	View Point of Interest

## LIST OF SYMBOLS

$\vec{N}$	Normal vector
$d_b$	Laser beam diameter
$\alpha_1$	Incidence angle
$\alpha_R$	Angular resolution
$d_{max}$	Maximum distance
$d_{min}$	Minimum distance

# CHAPTER 1 INTRODUCTION

## 1.1 Background

Several bridges have collapsed over the last few decades, causing many fatalities, injuries, and damages. For example, There are several bridges such as I-35W Highway Bridge, Minneapolis, U.S. (NTSB, 2008), which collapsed in 2007 because of insufficient inspection. Collapses could be prevented through efficient management and regular inspection. In Canada, most bridges have exceeded more than half of their expected service life, leaving a heavy burden of investment towards rehabilitation and maintenance (Gagnon et al., 2008). Despite the development and use of several new techniques for detecting bridge defects, visual inspection remains the main approach in detecting surface defects, such as cracks, spalling, and corrosion (Laefer et al., 2014). Conditions highlighted in previous inspections, combined with high traffic volume might demand higher inspection frequency (Transports Canada, 2012), leading many researchers to explore new method for increasing visual inspection efficiency, safety, and accuracy. The main problem of visual inspection is its reliance for the most part on manual data collection using non-equipped eyes, which is subjective and time-consuming (Dorafshan and Maguire, 2018). Also, some approaches require inspectors to access different bridge components using certain tools (e.g., cherry picker (Figure 1-1), ropes), which may affect traffic on the bridge and related project cost. According to Michigan Department of Transportation reports, closing off a four-lane highway bridge in a metropolitan area for 10 hours costs approximately US\$14,600 (Mcful, 2018). Moreover, using cherry pickers and ropes could be unsafe for the inspectors. Figure 1-2 shows the failure of a cherry picker over the edge of the bridge on I-84 road of West Hartford in 2015, which led to the inspector's death (The Vertikal Press, 2015).



Figure 1-1. Using a cherry picker to inspect a bridge (Facelift Ltd., 2017).



Figure 1-2. Failure of a cherry picker during under-bridge inspection (The Vertikal Press, 2015).

For many bridges (e.g., Montreal Turcot bridge (Figure 1-3)), the geometry is problematic, which complicates the use of supplementary devices while also raising the risk of falling for inspectors trying to access all components (Kim et al., 2015b). Therefore, to ensure safer, more efficient, and accurate bridge inspection, an automated data acquisition system should be developed (Liu et al., 2014).



Figure 1-3. Example of highway bridges (Montreal Turcot Interchange) (FOTOimage Montreal, 2017).

In the visual inspection of concrete bridges, surface defects (e.g., cracks) can be inspected using 3D Light Detection and Ranging (LiDAR) scanners as a Non-Destructive Testing (NDT) method. However, the commonly used terrestrial LiDAR is limited to stationary data collection, which reduces the accessibility to some components of the bridge. An unmanned system that supports the inspector's role provides a safer and more efficient inspection (Dorafshan and Maguire, 2018). For instance, an Unmanned Aerial Vehicle (UAV) can fly to all parts of a large structure to collect data using a mounted LiDAR or/and camera (Barazzetti and Scaioni, 2009). This technology saves time and money. For example, inspecting the second largest bridge in Minnesota with traditional methods, using three snoopers inspection vehicles, costs approximately US\$59,000 in eight days; while using a UAV would amount to around US\$20,000 in five days (Hampel and Maas, 2009). A UAV can scan inaccessible surfaces from closer distance, improving safety, accuracy, and efficiency. Both the scanner and the UAV have limitations and specifications which should be considered to achieve efficient results.

## 1.2 Problem Statement

The key problems of point cloud-based methods for concrete bridge surface defect detection can be attributed to three main issues as follows:

(1) The first issue is regarding the methods for detecting bridge surface defects using point clouds. According to the comparison between point clouds and images, which will be discussed in Section 2.3, using point clouds is more beneficial for surface defect detection of bridges. Several point-cloud based methods only focus on the point cloud features ( $x$ ,  $y$ ,  $z$ ), while there are also some features (i.e., colors and normal vectors), which may be useful for detecting the surface defects (Teza et al., 2009; Olsen et al., 2010; Kim et al., 2015b).

On the other hand, Deep Learning (DL)-based methods have attracted the researchers' attention for concrete surface defect detection (Mohammed Abdelkader et al., 2021; Wang et al., 2022). In computer vision, DL-based methods can be used for applications such as classification, semantic segmentation, and instance segmentation. In terms of detecting the defects, classification models



are able to identify the existence of a defect in the input, while semantic segmentation refers to labeling each point individually (Qi et al., 2017a). Instance segmentation provides an even more detailed analysis by distinguishing between different instances with the same label, in addition to labeling each point (Jiang et al., 2020). Compared to classification, semantic segmentation, which can be the initial step for instance segmentation, is more beneficial for surface defect detection due to providing detailed information about each point.

Despite the tremendous improvements in this area, no DL method is currently available for semantic segmentation of bridge surface defects without converting the raw point cloud dataset into images or voxels, which results in increasing the size of the dataset and leads to some challenges regarding storage capacity, cost, and training time.

Some point cloud-based semantic segmentation methods such as PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) are promising. These methods have been applied in segmenting bridge components (i.e., slabs, piers), but not for segmenting surface defects (i.e., cracks, spalls).

Furthermore, in visual inspection, inspectors try to detect different types and sizes of surface defects simultaneously. However, most of the current point cloud-based concrete surface defect detection methods focus on only one type of defects (Kim et al., 2015b; Turkan et al., 2018).

(2) The second issue is related to the data collection method. The reliability of DL-based methods depends on the size and quality of the dataset. Therefore, data collection process plays an important role in bridge inspection. During bridge inspection, depending on the type of visual sensor mounted on/under the UAV (e.g., camera, LiDAR), different types of data (e.g., images, 3D point clouds) can be collected. Although the UAV flying path can be controlled using remote controllers (Xu and Turkan, 2019), automating and optimizing UAV path planning is preferable for being able to trace a collision-free path with minimum flight time (Bircher et al., 2016a).

Covering all bridge surfaces by the UAV is of utmost importance. In case of using LiDAR, the data accuracy depends on several parameters, such as the incidence angle of the laser beam and scanner-structure distance. To increase the efficiency and accuracy of this approach, in addition to optimizing the flight path of the UAV considering the path length and full coverage of the inspected area, it is crucial to scan all parts of the bridge with a near perpendicular view. However, in the case of obstacle existence (e.g., bridge piers), achieving full coverage with near perpendicular view may not be possible. To provide more accurate results, using overlapping views (i.e., scanning the surface from more than one view) is recommended. However, this method could result in increasing the inspection cost and time. Therefore, overlapping views should be considered only for surface areas where defects are expected. It should be mentioned that at the time of developing this research, the technology for LiDAR-equipped UAV is not readily available to be practically applied for bridge inspection due to the lack of an affordable light high-quality 3D LiDAR capable of being mounted on UAV.

(3) The third issue is regarding lack of available point cloud dataset for concrete surface defect detection. Recently, the use of DL-based methods has been increasing in the construction industry (i.e., activity recognition (Torabi et al., 2021), building element classification (Koo et al., 2021)). In DL, a dataset plays a key role in terms of variety, diversity, accuracy, and size. In other words, a large high-quality dataset leads to better learning and can help to increase the performance and generalization of the trained model. Many image-based DL methods have been applied on the

publicly available datasets (i.e., MS-COCO (Lee et al., 2019), SDNET2018 (Maguire et al., 2018)) for surface defect detection. However, the lack of publicly available point cloud datasets for bridge surface defects is one of the reasons of the lack of studies in the area of point cloud-based methods.

### 1.3 Research Objectives

The long-term goal of this study is to achieve an effective and efficient bridge visual inspection using a LiDAR-equipped UAV. Thus, the following objectives are set for this research.

- (1) Creating a publicly available point cloud dataset for concrete bridge surface defect semantic segmentation.
- (2) Developing a point cloud-based semantic segmentation DL method to detect different types of concrete surface defects.
- (3) Proposing a novel near-optimal path planning method for LiDAR-equipped UAV with respect to the minimum path length and maximum coverage considering the potential locations of defects.

### 1.4 Overview of the Research Methodology

As shown in the overview of the research methodology in Figure 1-4, this research consists of two main modules, which are proposed to meet the main research objectives. Module 1 aims to create a publicly available dataset and detect concrete surface defects (i.e., spalls and cracks) using an adjusted network based on PointNet++ (Qi et al., 2017b). This network is called Surface Normal Enhanced PointNet++ (SNEPointNet++). The following aspects are considered to adjust the model: (1) increasing the size of the dataset by augmentation (i.e., flipping), (2) addressing the issue of the imbalanced dataset for priority classes, which have smaller parts of the dataset (i.e., cracks and spalls) (3) focusing on the most relevant features of defects by considering the color, normal vector, and depth as input features, and (4) applying sensitivity analysis to capture the best combinations of the hyperparameters for an accurate model. As shown in Figure 1-4, the point cloud dataset is collected using LiDARs and then prepared for training in four main steps: (1) normal vectors calculation, (2) annotation: annotating and labeling the point clouds into the classes (i.e., cracks, spalls, no defects), (3) augmentation: flipping the dataset horizontally and vertically, and (4) data preprocessing and adding the normalized depth value. The dataset is publicly available at SNEPointNet++ Dataset (Bolourian, 2022). Sensitivity analysis is used to select the values of hyperparameters of the network. The output of this module is a trained model, which is validated and tested on the unseen damaged concrete surfaces. This model can be used for the automatic semantic segmentation of the surface defects (i.e., cracks and spalls). This method is explained in detail in Chapter 3.

Module 2 focus is on path planning of LiDAR-equipped UAV in two main steps: (1) defining View Points of Interest (VPIs), and (2) finding the optimal path.

Since the model-based method is used for the proposed path planning, the 3D bridge models (e.g., CAD file, Bridge Information Model (BrIM)) should be provided. Those models can be developed based on the available design documents or the 3D scans of the structure. In the first step, considering several factors (i.e., Fields of View (FoVs) of the LiDAR, minimum distance between the LiDAR and the bridge surface, overlapping views, and the levels of the criticality) in defining the VPIs improves the accuracy and efficiency of data collection. During scanning the bridge, the laser beams may not detect a defect for different reasons: large incidence angle, non-perpendicular view, the existence of obstacles, and relatively large size of beam steps. Scanning the defects from

more than one VPI and providing overlapping views better estimate the defect size. On the other hand, the critical sections are located at the areas, which are expected to have a higher potential of the existence of defects. The bending moment and shear diagrams can identify the location of these critical sections. Then, an Importance Value (IV) is assigned to each section corresponding to its level of critically.

In the second step, a cost matrix is calculated based on the shortest collision-free path length between each two VPIs using A\*, which is a search-based path planning algorithm. Then, a Traveling Salesman Problem (TSP) is defined and solved using a Genetic Algorithm (GA) to find a near-optimal flight path with a minimum flight time passing through all VPIs based on the calculated cost matrix.

Finally, the bridge surface coverage is calculated considering the level of criticality. If the coverage is not acceptable, new VPIs are added, and the process should be repeated. This module is explained in detail in Chapter 4. The UAV flies based on the final path automatically to collect the 3D point clouds of the bridge surface.

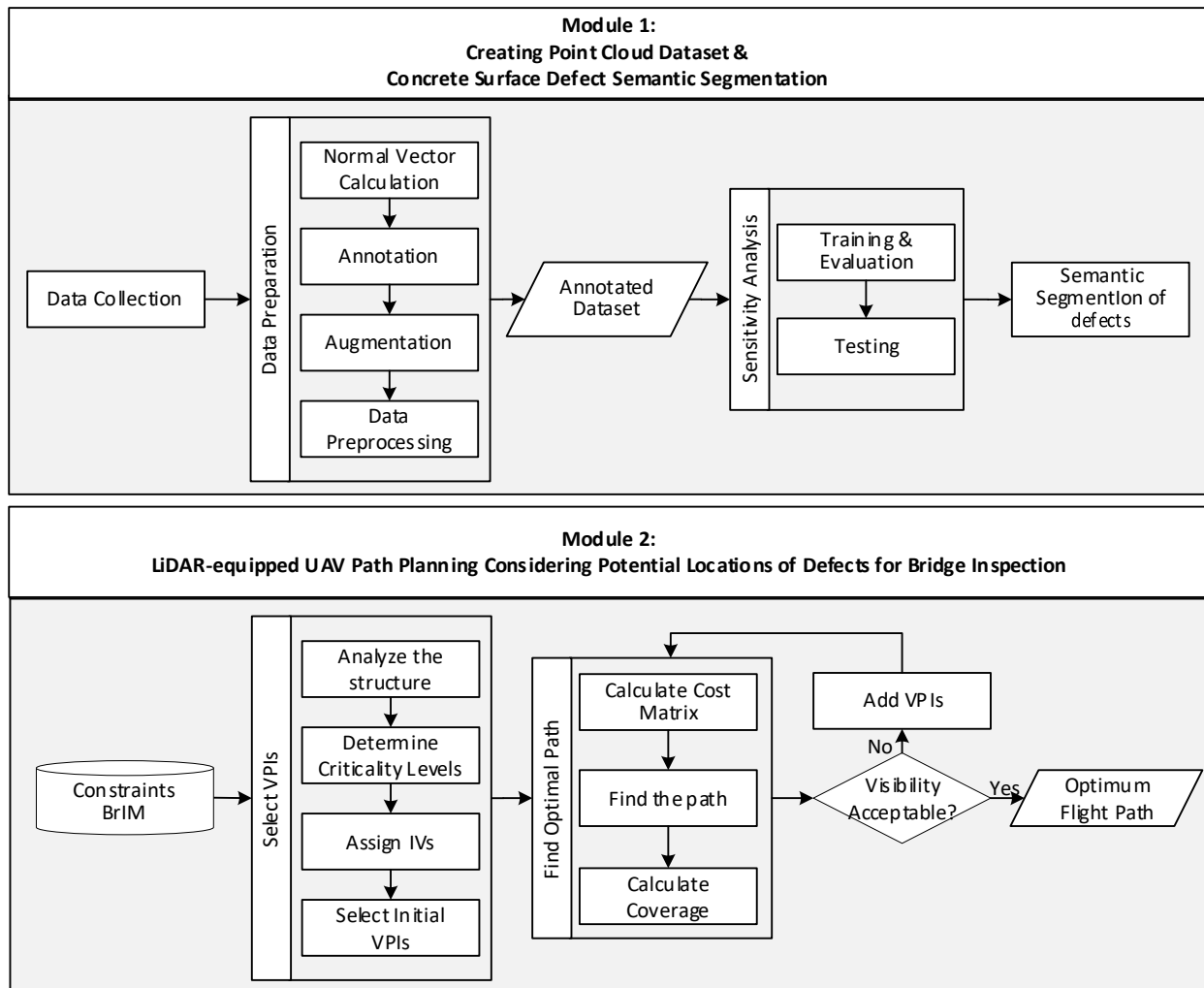


Figure 1-4. Overview of research methodology.

## 1.5 Thesis Organization

The remaining of this proposal is divided into five chapters, which are structured as follows:

### *Chapter 2: Literature Review*

This chapter starts with introducing two popular methods, camera- and LiDAR-based methods, in bridge visual inspection and comparing them based on their advantages and shortcomings. Afterward, the chapter reviews the existing studies related to defect detection using point cloud and ML and UAV path planning. First, the literature review comprises the concepts of DL-based semantic segmentation methods, followed by introducing point cloud-based semantic segmentation methods and their applications in inspection. Then, the relevant literature on point cloud-based bridge surface defect detection methods, including Machine Learning (ML) and non-ML methods, is reviewed to find the gaps in this area. The remaining part of this chapter is allocated to UAV path planning including the review of the most promising obstacle-free path planning algorithms, coverage path planning algorithms, and TSP solver methods. Finally, the most relevant UAV path planning studies are reviewed.

### *Chapter 3: Point Cloud-based Concrete Surface Defect Semantic Segmentation*

This chapter covers two objectives of this research: creating a point cloud dataset and concrete bridge surface defect semantic segmentation. It starts by explaining the main aspects considered in SNEPointNet++ to adjust the original PointNet++ and improve the network performance. This is followed by the framework of the two adjusted networks: adapted PointNet++ and SNEPointNet++. Adapted PointNet++ is applied to evaluate and ensure the feasibility of using PointNet++ for surface defect detection. Later, in SNEPointNet++, normal vectors are used as additional input to improve the performance of the network. Then, the publicly available dataset is created using the collected point clouds from four bridges in Montreal, which are used later in the case study to validate the proposed method and reach the best SNEPointNet++ model for semantic segmentation of cracks and spalls.

### *Chapter 4: LiDAR-Equipped UAV Path Planning Considering Potential Locations of Defects for Bridge Inspection*

This chapter presents the research method for LiDAR-equipped UAV path planning. First, the considerations, requirements, and constraints were introduced, followed by explaining the developed path planning method considering the potential location of defects. Afterward, the case study focused on planning a flying path for a LiDAR-equipped UAV to scan the lower surface of a three-span bridge deck.

### *Chapter 5: Summary, Contributions, and Future Work*

The work and the conclusions are summarized in this chapter. Moreover, the contributions and the limitations of the proposed methods are highlighted. Finally, some suggestions are recommended to overcome the limitations and expand this research in the future.

## CHAPTER 2 LITERATURE REVIEW

### 2.1 Introduction

This chapter presents the literature review in the areas related to the scope of this research, including bridge inspection, path planning, point-cloud semantic segmentation, and defect detection. The main purpose of this review is to identify the gaps in previous studies and determine the most appropriate techniques to overcome those shortcomings. To this end, first, bridge inspection and the most common methods of visual bridge inspection are introduced in Sections 2.2 and 2.3, respectively, followed by the comparison of LiDAR scanning platforms. Section 2.5 reviews point cloud-based semantic segmentation methods, which are used in inspection as well as the point cloud-based ML and non-ML methods, which are used in concrete surface defect detection. Finally, Section 2.6 provides extensive reviews of UAV path planning and its application in inspection.

### 2.2 Bridge Inspection

Structural inspection is essential to improve the sustainability and safety of infrastructure systems (e.g., bridges). Aging and environmental conditions are some of the inevitable causes of bridge deterioration (Le et al., 2017). A good maintenance system should be undertaken to keep the infrastructures in a fully functioning or operating condition. Figure 2-1 illustrates the quantitative relationship between the deterioration of Canada's infrastructure and the level of maintenance (Mirza, 2006). The quality of the infrastructure performance is rated on a scale from 0 to 1, where 1 is the perfect performance, and four levels of maintenance are considered, 0%, 1%, 1.5%, and 2%. The maintenance level is corresponding to the percentage of the infrastructure construction cost. It can be seen that a higher level of maintenance leads to longer anticipated service life and a higher level of performance. To have effective maintenance, efficient inspection is required.

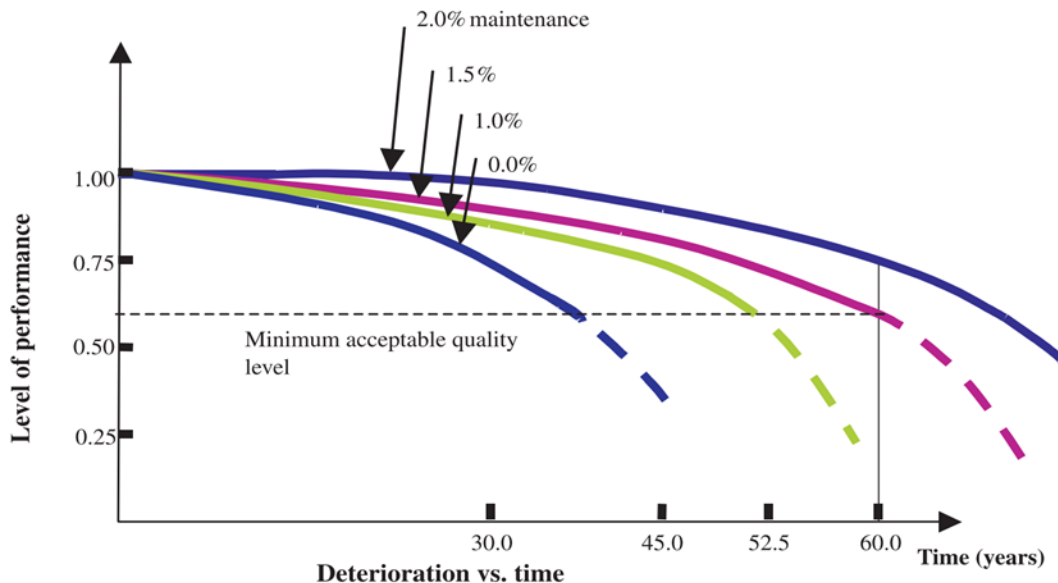


Figure 2-1. Level of performance vs. time relationship for various levels of maintenance (Mirza, 2006).

Bridge inspection has been developed over many decades. National Bridge Inspection Standards (NBIS) were established in 1971 by the Federal Highway Administration (FHWA), including regulations, policies, and guidelines to standardize inspection methods and qualifications.

Although there is no national Canadian bridge inspection standard, a part of the Railway Transportation Safety Guideline is related to bridge inspection (Government of Canada, 2021).

Each bridge inspection standard includes a different classification for bridge inspection (Hsien-Ke et al., 2017). However, there are five main types, namely:

- *Initial inspection*: It is the inspection done after finishing the construction and should be repeated after the rehabilitation of the bridge or any other changes related to the bridge.
- *Routine inspection*: This intermediate-level inspection is scheduled based on the bridge condition. According to the Ontario Structure Inspection Manual (OSIM) (2008), all bridges in good condition shall be inspected every two years. However, the inspection interval may be decreased, and the bridge may be inspected more frequently after a while because of aging or poor condition. Inspecting the bridges regularly helps the inspector to track the propagation of defects.
- *Damage inspection*: In case of unexpected damages, damage inspection is done to assess the necessity of urgent bridge load restriction or bridge closure, as well as an urgent repair action
- *In-depth inspection*: This inspection is done to inspect the important components or the ones susceptible to defects. Moreover, in-depth inspection makes an effort to cover the missed or improbably inspected elements over routine inspection.
- *Interim/Special inspection*: It can be defined as a specific inspection that should be scheduled at the discretion of the inspection responsible authorities.

According to Transport Canada Guideline, “each Bridge Safety Management Program (BSMP) should include a visual inspection for each bridge in service at least once each calendar year with not more than 540 days between any successive inspections”. While more frequency may be required considering the condition noted on the previous inspection, the type and configuration of the bridge, and volume of traffic (Transports Canada, 2012).

Reinforced concrete structures experience loss of integrity during their service life because of corrosion, salt and acid actions, the influence of high temperature, shrinkage, or water (Bien and Zwolski, 2007). Concrete deck deterioration can be classified into two categories based on the location of the defect: surface or subsurface defects (Ahlborn et al., 2010). Visual inspection is a common method to detect surface defects. As shown in Table 2-1, four levels of severity can be determined for some defects (e.g., scaling, spalling, delamination) based on their sizes (i.e., depth, width, and height of the affected area).

Table 2-1. Defects severity based on depth (*d*), width (*w*), and height (*h*) of the defect (Ontario Ministry of Transportation, 2008)

Type of defects	<i>Light</i> *	<i>Medium</i> *	<i>Severe</i> *	<i>Very Severe</i> *
<i>Scaling</i>	$d < 5$	$5 \leq d < 10$	$10 \leq d < 20$	$20 \leq d$
<i>Delamination</i>	$w, h < 150$	$150 \leq w, h < 300$	$300 \leq w, h < 600$	$600 \leq w, h$
<i>Spalling</i>	$w, h < 150$ $d < 25$	$150 \leq w, h < 300$ $25 \leq d < 50$	$300 \leq w, h < 600$ $50 \leq d < 100$	$600 \leq w, h$ $100 \leq d$

\* All dimensions are in mm

## 2.3 Visual Inspection Methods

Visual inspection is a common structural inspection method. Using naked eyes in bridge inspection is unsafe, time-consuming, expensive, and subjective because of human errors (Guldur et al., 2015). As shown in Figure 2-2, the inspectors may use their experience and manual measurements during the inspection, which results in different outcomes depending on the level of inspector experience (Morgan and Falkner, 2001). Moreover, reaching all parts of the surface takes time and puts the inspector at the risk of falling hazards during inspection (Kim et al., 2015a). To this end, several technologies (i.e., cameras, LiDARs) have been recently developed to detect surface defects automatically and accurately.



Figure 2-2. Visual inspection using naked eyes (Ma and Sacks, 2016).

### 2.3.1 Camera-based Methods

Camera-based methods using images are common and cost-efficient. These methods are used in detecting surface defects such as air pockets (Zhu and Brilakis, 2008), potholes (Koch and Brilakis, 2011), cracks (Hutchinson and Chen, 2006), and determining some characteristics such as the length and width of cracks (Barazzetti and Scaioni, 2009; Hampel and Maas, 2009; Adhikari et al., 2014). In terms of concrete surface defect detection and assessment, many improvements are needed to overcome the questionable noisy data and increase the accuracy, which is related to camera pose, camera distance, and environmental conditions (e.g., lighting and shading at different locations) (Koch et al., 2015).

### 2.3.2 LiDAR-based Methods

LiDAR-based methods focus on a 3D point cloud dataset containing geometrical information of the sparse captured points collected by a LiDAR. The geometrical information ( $x, y, z$ ) of points is calculated based on the measured distances of the scanned objects or surfaces from the scanner location (Rashidi et al., 2020). Many researchers have studied how these methods apply to surface defect detection (Laefer et al., 2014; Kim et al., 2015b; Guldur and Hajjar, 2016) and mass losses (Teza et al., 2009).

### 2.3.3 Comparison between Camera- and LiDAR-based Methods

Both camera- and LiDAR-based methods are much more time-efficient than manual measurements (Rashidi et al., 2020). The necessity of providing supplementary information (i.e., camera lens, focal length) before analyzing the images and its high sensitivity level to environmental conditions (e.g., lighting and shading) are the main shortcomings of camera-based methods (Laefer et al., 2014). Although LiDAR-based methods can work properly regardless of any information related

to the equipment or target, determining that information may be helpful in improving the inspection method.

Furthermore, most camera-based methods are defined for simple curved or flat concrete surfaces. Consequently, they may fail at analyzing more complex structures, geometries, and materials (Koch et al., 2015). For instance, Zhang et al. (2014) proposed an image processing method to detect the cracks of simple subway tunnels and concluded that although the images were practical in their case study, LiDAR scanning is a better technique for inspecting complex structures.

Having the third dimension is the main advantage of point clouds over images. Although images collected by a depth camera can represent the surface depth, several images with different setups must be collected to cover the whole part of a structure depth due to its narrow FoV. Moreover, the registration process of these images is time-consuming (Maru et al., 2021), which further limits their applicability. Another approach to extract the depth of concrete surface defects (i.e., spalling distress) is a regression analysis model based on computer vision and situ measurements (Dawood et al., 2017). Computer vision using cameras may detect the boundaries of defects more efficiently than LiDAR (Demir and Baltasvias, 2012).

Despite the high initial cost of LiDARs, using them may be more profitable and economical in the long term. Some researchers focused on improving the resolution of image-based reconstructed models (Khaloo et al., 2018). Khaloo et al. (2018) generated a 3D point cloud bridge model using 2,000 high-resolution images, which resulted in three times higher local noise and lower precision compared to the scanned bridge model using a LiDAR. Moreover, the quality of the image-based reconstructed model is lower than the LiDAR-based model due to the complexity of detecting the interaction of the background and the inspected bridge, especially in case of low contrast.

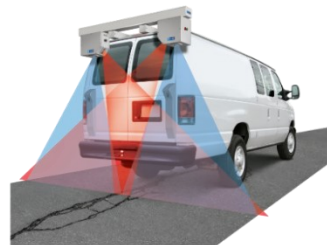
To consider the colors of images as well as the three dimensions of point clouds, several methods are developed using a dataset, which includes all these features (Olsen et al., 2010; Guldur and Hajjar, 2017).

## 2.4 LiDAR Scanning Platforms

Laser scanning platforms are of three types: Terrestrial Laser Scanning (TLS), Mobile Laser Scanning (MLS), and Airborne Laser Scanning (ALS) (Crosby, 2016), as shown in Figures 2-3 and 2-4.



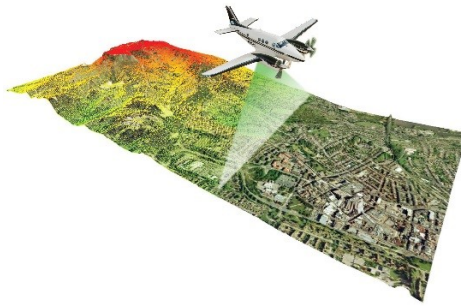
(a) TLS using tripod  
(FARO Technologies Inc., 2012)



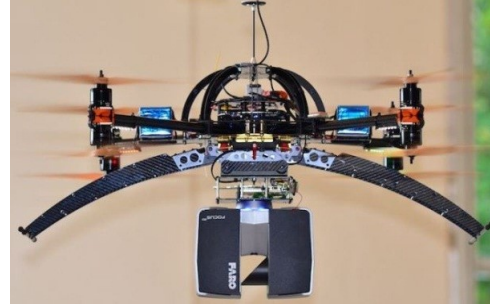
(b) MLS using car (Habel, 2017)

Figure 2-3. Terrestrial laser scanning and mobile laser scanning.





(a) Using airplane (Bennett, 2015)



(b) Using UAV (Synergy Positioning Systems Ltd., 2018)

Figure 2-4. Airborne laser scanning.

TLS uses tripods but provides limited access to bridge components. TLS typically operates at short ranges (few meters to hundreds of meters) and has cm-scale footprints (diameter of laser pulse). The spacing between consecutive pulses varies in the order of millimeters to centimeters. The instrument is stationary and is generally fixed on a survey tripod about 1.5 m above ground.

In MLS, a LiDAR is mounted on a moving object (e.g., van) (Lehtomäki et al., 2016). It is used mostly in pavement inspection to find cracks or road markings (Guan et al., 2015). The visual sensor can be mounted on a flying vehicle (e.g., airplane, helicopter, or UAV) in the ALS method, as shown in Figure 2-4. Since UAV can reach parts of large structures that are difficult to access, it has been used for monitoring bridges (Lattanzi and Miller, 2015) and power-lines (Kroll et al., 2009), as well as for the 3D reconstruction of boiler power-plant (Burri et al., 2012) and gas pipelines surveillance (Boon and Lovelace, 2014).

All three types undergo small scanner vibrations that can be caused by the LiDAR instability, the moving supporting vehicle (e.g., car or UAV), or the wind effect on the UAV (Reshetyuk, 2006). However, ALS is the most vibration-sensitive system (Xu et al., 2015). TLS provides high stability for the scanner and less vibration, but it is not time and cost-efficient because of the need to scan from multiple points.

In contrast to TLS, in ALS, the easy access to most parts of the structure helps the UAV to collect a denser set of points from a closer distance with a nearly perpendicular view of the damaged surfaces. This unmanned method also eliminates safety risks to inspectors (Metni and Hamel, 2007). Although the UAV flight path can be controlled remotely (Xu and Turkan, 2019), having an automated path planner can provide an optimal flight path.

## 2.5 Defect Detection Using Point Cloud and ML

Recently, laser scanning has become one of the main sources of data collection in different phases of a project (e.g., construction, operation, and maintenance) including surface defect detection. The number of research projects in this field has increased over the last decade (Khallaf and Khallaf, 2021). In this section, after a summary of the related ML-based methods, their applications in the inspection are reviewed. Then, the recently developed point cloud-based surface defect detection methods, including ML and non-ML methods, are presented.

## 2.5.1 Machine Learning and Deep Learning

### 2.5.1.1 Machine Learning

Artificial Intelligence (AI) is a technology with the ability of human intelligence simulation. In other words, it gives computer systems the “learning” and “human-like decision making” ability (Michalski et al., 2013). ML is a subfield of AI, enabling machines to learn from the available data from processing data without explicit programming. ML algorithms are used to train a model that can explain the relationship between several dataset features. Those algorithms can be categorized into three main groups: supervised, unsupervised, and reinforced learning (Khallaf and Khallaf, 2021).

### 2.5.1.2 Deep Learning

DL is a subset of ML that uses the experience of the dataset to be learned. The DL architecture is constructed by several layers including input, multiple hidden layers, and output, which are responsible for receiving data, extracting patterns, and producing results, respectively (Ongsulee, 2018). Deep Neural Network (DNN) is a well-known ML network inspired by the biological neurons of human brains with multiple hidden layers. Convolutional Neural Network (CNN) is a class of Neural Network (NN) specialized in using a convolving process to learn the input characteristics.

Although the breakthrough moment of CNN was in 2012, one of the main foundations of the works was invented by LeCun et al. (1998), which was an algorithm to recognize online handwriting. Each CNN hidden layer contains convolution feature extraction, nonlinear activation, and down-sampling. In CNN, each layer unit receives input from a set of units located in a small neighborhood in the previous layer. The local features (e.g., oriented edges, corners, and end-points) can be extracted by local receptive fields, which are the regions covered by a filter. A filter including weights is sliding or convolving around the input to generate a feature map. In a linear classifier, the weights are multiplied by the original pixel value for convolving (Lin and Shen, 2018). To reduce the resolution of the feature map and the sensitivity of the shifts, each convolutional layer is followed by a subsampling (pooling) layer to aggregate information within a region (Sun et al., 2017).

Linear activation functions are very simple to be solved, and their ability is limited in recognizing complex mapping. Therefore, nonlinear activation functions (e.g., Rectified Linear Unit (ReLU), Sigmoid, Tanha) can be used to solve the complex nonlinear problem (Sharma et al., 2017). Nonlinear activation functions are essential to remove the redundant data while preserving the features and to solve complex problems of complicated networks. ReLU, which is widely-used due to its easy computation and fast convergence speed, can be calculated using Equation 2-1. It returns zero if it receives any negative value; otherwise, it returns the input value (Lin and Shen, 2018).

$$f(x) = \max(0, x) \quad \text{Equation 2-1}$$

Figure 2-5 shows a piecewise function created by ReLU. Having more linear regions improves the network performance (Montufar et al., 2014). The number of linear regions ( $\Omega$ ) is bounded by

$$\Omega \left( \binom{n}{i}^{(L-1)d} \cdot n^d \right)$$

where  $L$ ,  $d$ , and  $n$  represent the number of layers, input dimension, and the number of nodes, respectively (Montufar et al., 2014).

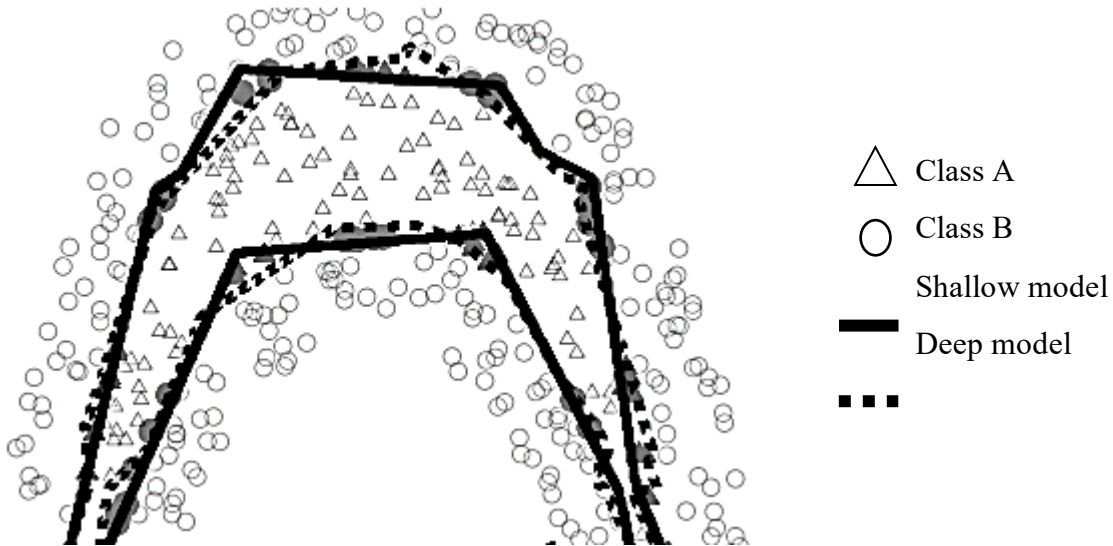


Figure 2-5. Binary classification using ReLU (Montufar et al., 2014)

Focusing on the most relevant features during training leads to higher accuracy in training and saves time (Brownlee, 2014). Although a neural network can be trained to distinguish between the levels of impact of different features in classification and segmentation, it may be confused, especially when the training dataset is not big. Therefore, the irrelevant features are removed from the dataset before training, or some methods (i.e., filtering and wrapping) can be used for feature selection (Chandrashekar and Sahin, 2014).

### 2.5.2 Point Cloud Datasets

Data plays an important role in any ML system. Therefore, collecting adequate datasets is necessary to achieve an accurate model. The dataset should be properly generated based on the system, which can be used for classification and semantic segmentation. Datasets should represent the appropriate parameters and include different cases and scenarios based on the requirement. For example, in the case of detecting defects, the dataset should include the defects in different sizes, directions, and types.

The datasets are categorized into three groups based on the nature of the data: 2D or plain Red Green Blue (RGB), 2.5D or RGB-Depth, and pure volumetric or 3D datasets (Garcia-Garcia et al., 2017). Since the focus of most of the previous research was on images, large-scale 2D datasets are available for semantic segmentation, such as PASCAL Visual Object Classes (VOC) (Everingham et al., 2015), PASCAL Context (Mottaghi et al., 2014), Semantic Boundaries Dataset (SBD) (Hariharan et al., 2011), Youtube-Objects (Prest et al., 2012), and SDNET2018 (Maguire et al., 2018). These datasets include gray-scale or RGB images. 2.5D datasets contain both RGB information and depth maps such as SUN3D (Xiao et al., 2013) and NYUDv2 (Silberman et al., 2012).

3D datasets, including Computer-Aided Design (CAD) meshes or point clouds, are limited, and generating them is costly and difficult. Since point clouds are disordered data and only a few DL methods can process them, they are not popular. A benchmark for 3D Mesh Segmentation (Chen

et al., 2009), Sydney Urban Object Dataset (Quadros et al., 2012), Large-Scale Point Cloud Classification Benchmark (Semantic3D) (Hackel et al., 2016), ShapeNet Part (Yi et al., 2016), and Stanford 3D Indoor Scene Dataset (S3DIS) (Armeni et al., 2017) are examples of available datasets for semantic segmentation (Garcia-Garcia et al., 2017). However, they have limited classes. The information of each dataset (i.e., the number of samples, the number of classes, their application) is listed in Table 2-2.

Table 2-2. Examples of 3D datasets for semantic segmentation (Garcia-Garcia et al., 2017).

3D datasets	Number of mesh/points	Number of Classes	Application	Sample of Classes
<i>A benchmark for 3D Mesh Segmentation</i>	380 meshes	19	General Objects	human, cup, glasses, airplane, ant, chair, octopus, table, teddy, hand, plier, fish, bird, armadillo, bust, mech, bearing, vase, four-leg
<i>Sydney Urban Object Dataset</i>	41 MPts	26	Urban Objects	vehicles, pedestrians, signs, trees
<i>Semantic3D</i>	4 BPs	8	Urban Objects	churches, streets, railroad tracks, squares, villages, soccer fields, castles
<i>ShapeNet Part</i>	32 MPts	16/50	Common Objects/parts	airplane, earphone, car, motorbike, bag, mug, laptop, table, guitar, knife, rocket, lamp, chair, pistol, car, skateboard
<i>S3DIS</i>	70.5 MPts	13	Indoor Objects	ceiling, floor, wall, column, beam, window, door, table, hair, bookcase, sofa, board, clutter

### 2.5.3 Point Cloud-based Semantic Segmentation Using DL

DL can be used for semantic segmentation of point clouds. Semantic segmentation aims to separate point clouds into several subsets and label each point according to the corresponding semantic meanings. Point clouds can be indirectly or directly used for semantic segmentation (Guo et al., 2020). Using two most common indirect methods, called Multiview-based (Boulch et al., 2018; Yavartanoo et al., 2018; Zhao et al., 2018) and voxel-based methods (Maturana and Scherer, 2015; Wu et al., 2015; Wang and Lu, 2016), point clouds are transformed into intermediate representations, which are multi-view images and voxel grids, respectively. Transforming point clouds into these representations results in voluminous data with unclear invariances in some cases (Grilli et al., 2017). In multi-view-based methods, after learning from 2D projections, the intermediate segmentation results are projected back to the raw point cloud. The generated meshes are complex with combinatorial irregularities (Qi et al., 2017a). Unlike the dataset of a voxel-based approach, which contains an ordered grid of point clouds (Pierdicca et al., 2020), a point cloud-based one does not require specific pre-processing, and the unordered dataset is trained directly. Moreover, learning from point clouds is easier than meshes due to their simplicity and unified structure. Due to the higher computational efficiency, point cloud-based methods have recently been popular in the construction industry (Yin et al., 2021).

#### 2.5.3.1 PointNet

PointNet (Qi et al., 2017a) is the pioneer point cloud-based DL architecture designed for point cloud classification, part segmentation, and semantic segmentation directly. This network was proposed to overcome the problems related to voxelization and rendering point clouds. Its input is

a set of points, which has three main characteristics. First, these points are unordered. Interaction among neighboring points is the second important characteristic of these datasets. The points are not isolated, and the local structure of the combination of neighboring points affects the semantic information of the point sets. The third characteristic of point clouds is being invariant under transformation. These three characteristics are considered in designing the architecture of PointNet. As shown in Figure 2-6, the network learns each point features using shared Multilayer Perceptrons (MLPs), which is a supplement of a feed-forward network, and global features using symmetrical pooling functions.

As shown in Figure 2-6, PointNet has two sets of MLP. The first MLP accepts blocks of points as input, and each layer extracts detailed features of points by passing through a “unit PointNet”, which is similar to one-by-one convolving on the blocks in CNNs. Every hidden layer includes batch normalization and the ReLU activation function. The main goal of this set of MLP is to extract local features per point from a 9-dimensional input including coordinates, colors, and normalized values of each point. The output of the first MLP is a vector of all input points, where every point has a weight. This vector represents the extracted local features of points. A max-pooling layer is applied to the feature vector to down-sample the features, followed by the second set of MLP to extract the global features of each point. The segmentation network has a set of MLP fed by the concatenation of the extracted local and global features. Thus, the score of each point can be predicted based on both global semantics and local geometry. Each convolutional layer in this set of MLP is followed by a dropout layer, except the last one. The output of this network is a vector of predicted probabilities belonging to each class for every point. Qi et al. (2017a) used S3DIS (Armeni et al., 2017) to validate PointNet semantic segmentation.

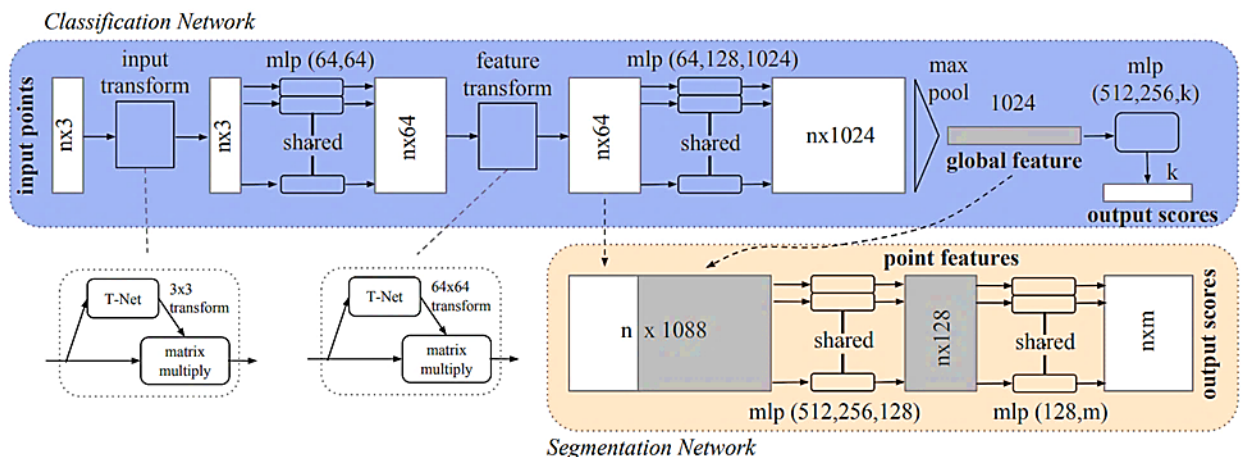


Figure 2-6. PointNet architecture (Qi et al., 2017a).

### 2.5.3.2 PointNet++

PointNet has two main shortcomings: (1) lack of local context learning and (2) translation invariance limitation (Yin et al., 2021). To overcome these limitations, PointNet++ proposed by Qi et al. (2017b) has hierarchical feature learning, which uses multiple Vanilla PointNet learners with different scales. As shown in Figure 2-7, first, a set of center points is sampled using the furthest-point sampling algorithm. Another main step is determining the neighboring points around each sampled center point using query ball grouping. Then, Vanilla PointNet is applied for local feature learning from each center and neighborhood. Moreover, since the density of point clouds

is not unified in all parts of a real dataset, PointNet++ uses multi-scale and multi-resolution grouping (Figure 2-8).

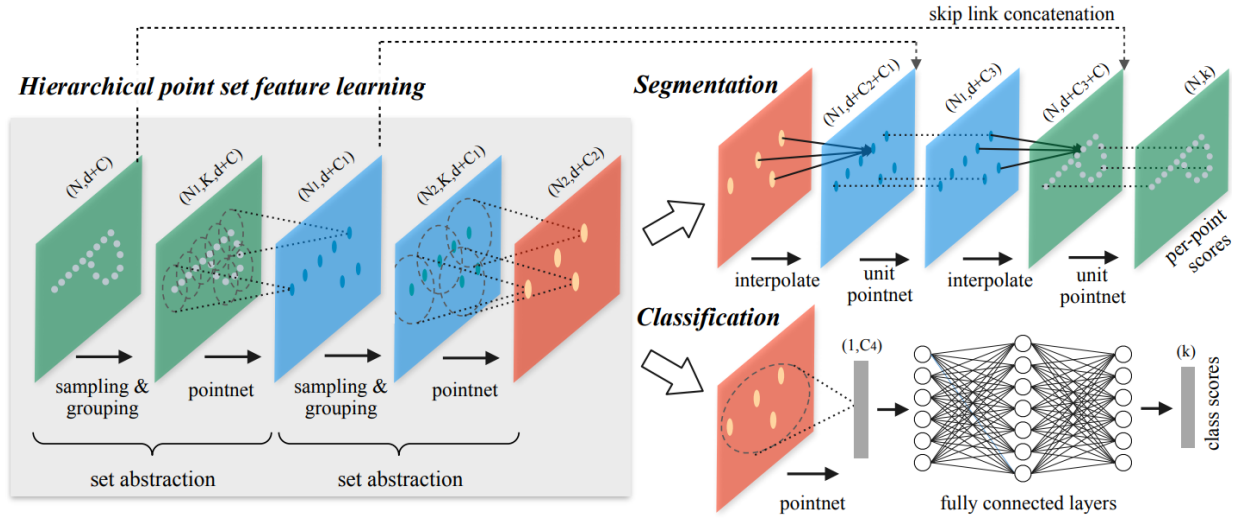


Figure 2-7. PointNet++ architecture for single scale point grouping (Qi et al., 2017b).

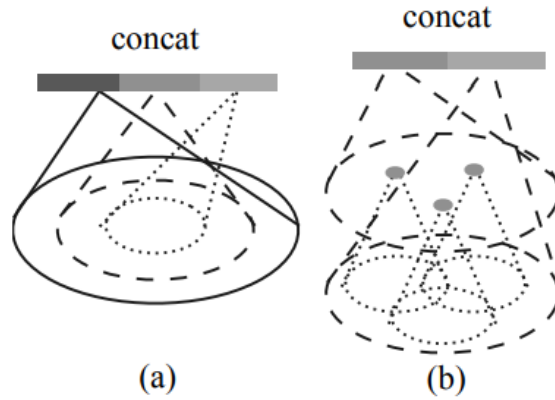


Figure 2-8. (a) Multi-scale grouping (MSG); (b) Multiresolution grouping (MRG) (Qi et al., 2017b)

### 2.5.3.3 Point Cloud-based DL Application in Structural Inspection

Some studies have been focused on the implementation of point cloud-based DL, such as PointNet, PointNet++, Dynamic Graph CNN (DGCNN), and PointCNN, for inspection of sewer defects (Haurum et al., 2021) and underwater pipes (Martin-Abadal et al., 2021).

Nasrollahi et al. (2019) adapted PointNet for semantic segmentation of the concrete bridge surface defects, which resulted in 74.9% accuracy and 46.6% Intersection over Union (IoU) (Figure 2-9). Using adapted PointNet++ led to 3.9% and 1.6% increases in defect detection accuracy and IoU, respectively (Nasrollahi, 2019).

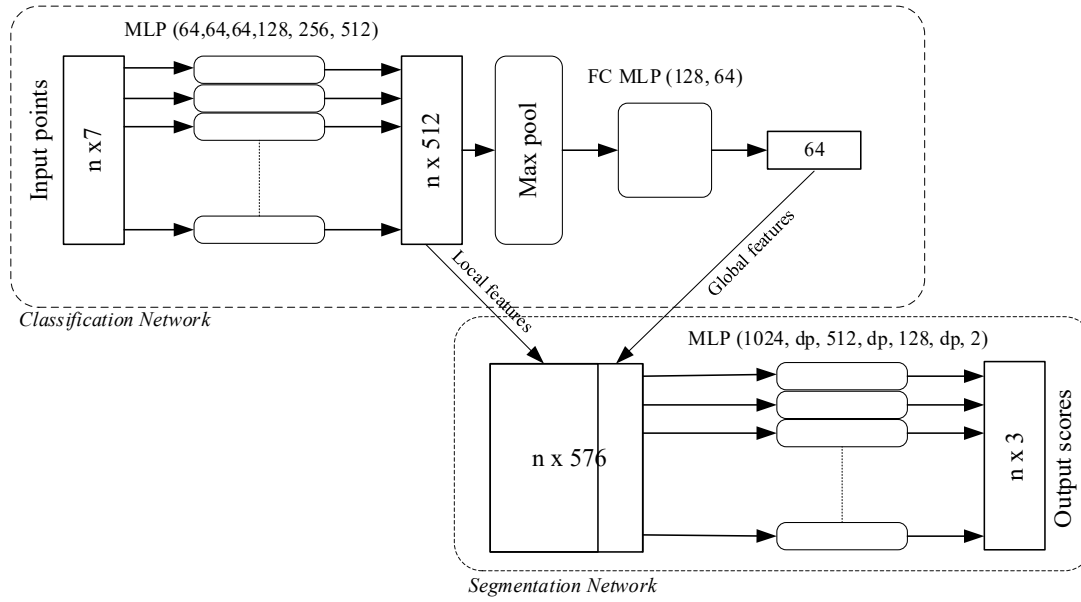


Figure 2-9. Adapted PointNet for defect detection (Nasrollahi et al., 2019)

Kim et al. (2020) used PointNet to detect decks and piers of reinforced concrete bridges from point clouds. Due to the small size of the training dataset, only two types of components were detected. Kim and Kim (2020) detected five types of components (e.g., abutment, slab, pier, girder, surface) using three different DL networks: PointNet, PointCNN, and DGCNN. The results showed that the Overall Accuracy (OA) of DGCNN was 0.66% and 1.94% more than PointNet and PointCNN, respectively. To implement PointNet, the training dataset should be divided into partitions, which may cause losing global features of large-scale scenes such as bridge structures (Xia et al., 2022). Hence, Xia et al. (2022) proposed a combination of ML and local descriptors focused on local regions. Based on the results, the performance of their proposed method was 12.39% and 48.8% better than PointNet in terms of OA and mean Intersection over Union (mIoU), respectively.

The proposed combined local descriptor and ML approach only concerns local regions and thus has the evident advantage when high precision semantic models are demanded. Haurum et al. (2021) proposed using PointNet and DGCNN for the sewer defect detection using point clouds. DGCNN performed much better than PointNet in terms of both OA and mIoU. Another application of PointNet is in detecting underwater pipes and valves (Martin-Abadal et al., 2021). Moreover, Yin et al. (2021) detected industrial objects (e.g., pipe, valve, beams, and tanks) using PointNet, PointNet++, and ResPointNet++ resulting in 94.02% OA and 87.34% mIoU. Koo et al. (2021) evaluated the application of PointNet and Multi-View Convolutional Neural Network (MVCNN) to classify ten types of infrastructure Building Information Model (BIM) elements including columns, culverts, retaining walls, sumps, and wing walls. To implement PointNet and MVCNN, each element was converted into point clouds and 12 multi-view images, respectively. Compared to PointNet, using MVCNN algorithm leads to better results. However, training those models is not practical due to their long computation time (Koo et al., 2021). Bahreini and Hammad (2021) adapted DGCNN to detect two types of concrete surface defects, cracks and spalls, which resulted in 95.94% and 71.06% OA and mIoU, respectively.

Table 2-3 illustrates the studies in inspection, which have been recently developed based on the well-known point cloud-based semantic segmentation models. Based on the literature review, only two studies focused on structural concrete surface defect detection (e.g., cracks and spalls). Moreover, the results can be directly compared only if the same dataset is used.



Table 2-3. Usage of point cloud-based methods in inspection (construction industry).

Reference	Objective	Classes	Method	Results (%)	Dataset
<i>Nasrollahi et al. (2019)</i>	- Concrete surface defect detection	2 classes (defect, non-defect)	PointNet	OA: 85.7	21.5MPts
<i>Pierdicca et al. (2020)</i>	- Semantic segmentation of historical architectural elements	10 classes (arc, column, decoration, floor, door, wall, window, stairs, vault, roof)	PointNet	OA: 21.58 mIoU: 10.93	1MPts
			PointNet++	OA: 24.48 mIoU: 17.96	
			DGCNN	OA: 54.67 mIoU: 35.78	
			PCNN	OA: 39.50 mIoU: 33.11	
<i>Kim and Kim (2020)</i>	- Comparison of bridge component classification DL methods	6 classes (abutment, slab, pier, girder, surface, background)	PointNet	OA: 93.83 mIoU: 84.29	N/A
			PointCNN	OA: 92.55 mIoU: 76.78	
			DGCNN	OA: 94.49 mIoU: 86.85	
<i>Kim et al. (2020)</i>	- Bridge components segmentation	3 classes (deck, pier, background)	PointNet	OA: 94 mIoU: 84	N/A
<i>Haurum et al. (2021)</i>	- Built a publicly available sewer point cloud dataset - Apply two DL methods for defect classification of sewer	4 classes (normal, displacement, brick, rubber ring)	DGCNN	OA: 47.93 mIoU: 46.10	17,027 Pts
			PointNet	OA: 18.38 mIoU: 18.52	
<i>Yin et al. (2021)</i>	- Built a publicly available industrial indoor LiDAR dataset for 3D semantic learning. - Proposed two neural modules to learn local structures and enable deeper networks	6 classes (I- beam, pipe, pump, rectangular beam, tank)	PointNet	OA: 53.02 mIoU: 21.14	5MPts
			PointNet++	OA: 70.58 mIoU: 45.52	
			ResPointNet++	OA: 94.02 mIoU: 87.34	

Table 2-3. Usage of point cloud-based methods in inspection (construction industry). (Cont.)

<i>Ref</i>	<i>Objective</i>	<i>Classes</i>	<i>Method</i>	<i>Results (%)</i>	<i>Dataset</i>
<i>Martin-Abadal et al. (2021)</i>	- Semantic segmentation of underwater pipe and valves	3 classes (pipe, valve, background)	PointNet	Mean F1-score: 89.3	262Pts
<i>Koo et al. (2021)</i>	- Classify infrastructure BIM elements	10 classes (column, 3 types of culverts, 5 types of walls, sump)	PointNet	OA: 83 F1-score: 87	1,496 elements
			MVCNN	OA: 98 F1-score: 98	
<i>Bahreini and Hammad (2021)</i>	- Detect concrete surface defect detection	3 classes (crack, spall, no defect)	DGCNN	OA: 95.94 mIoU: 71.06%	49MPts
<i>Xia et al. (2022)</i>	- Bridge components segmentation	2 classes (slab, pier)	PointNet	OA: 84.35 mIoU: 45.92	447MPts
			Proposed	OA: 96.74 mIoU: 94.72	

#### 2.5.4 Concrete Surface Defect Detection Using Point Cloud

Damage detection using point clouds can be based on geometrical features or/and color information (Mohammadi et al., 2019). Hou et al. (2017) proposed the combination of clustering algorithms (e.g., k-means, fuzzy c-means, subtract, and density-based spatial) and color (RGB) and intensity information of the point cloud dataset to detect surface defects such as metal corrosion and section losses. The results showed that k-means and fuzzy c-means clustering had better performance in terms of accuracy. Moreover, due to the effect of lighting conditions on color information, this information is less reliable in comparison to intensity information (Hou et al., 2017).

Liu et al. (2011) presented an algorithm to detect defects based on distance and gradient criteria of concrete bridge surfaces. This method was applied on a bridge cap to detect and quantify mass losses. Laefer et al. (2014) contributed a mathematical basis for using TLS to detect cracks in unit-block masonry (i.e., stone, brick, or concrete masonry units). According to the mathematical and experimental results, the general trend of this method was overestimated crack width. The most sensitive factor was the orthogonal distance. To have a reliable result in detecting the width of cracks less than 5 mm, the orthogonal scanner distance to the inspected surface should not be more than 10 m. Therefore, more sampling steps, as a function of scan distance, lead to higher accuracy. It is recommended to locate the TLS at a distance below 7.5 m from the target with a maximum 30° FoV. In another study, a new automated technique that can simultaneously localize and quantify spalling defects on concrete surfaces using TLS was proposed by Kim et al. (2015b). The proposed technique was applicable for defects larger than 3 mm in depth and width. As the guidance for optimal scan parameter selection, the most accurate localization and quantification can be detected in distances less than 12 m and incidence angle less than 15°, while the angular resolution does not exceed 0.018° (Kim et al., 2015b). Moreover, the minimum subdivision size of the scanned surface ( $\Delta$ ) can be calculated using Equation 2-2.

$$\Delta_{min} = \frac{L \times \alpha_R}{\cos(\alpha_1)} \quad \text{Equation 2-2}$$

where  $L$ : distance between the scanned concrete surface and the scanner ( $m$ );  $\alpha_R$ : angular resolution ( $rad$ ); and  $\alpha_1$ : incidence angle ( $degree$ ).

Figure 2-10 illustrates: (a) an overview of scan points lying on a concrete surface; (b) the definition of the angle deviation from the reference direction in the x-z plane view; and (c) the definition of the distance deviation from the globally fitted plane in the x-z plane view.

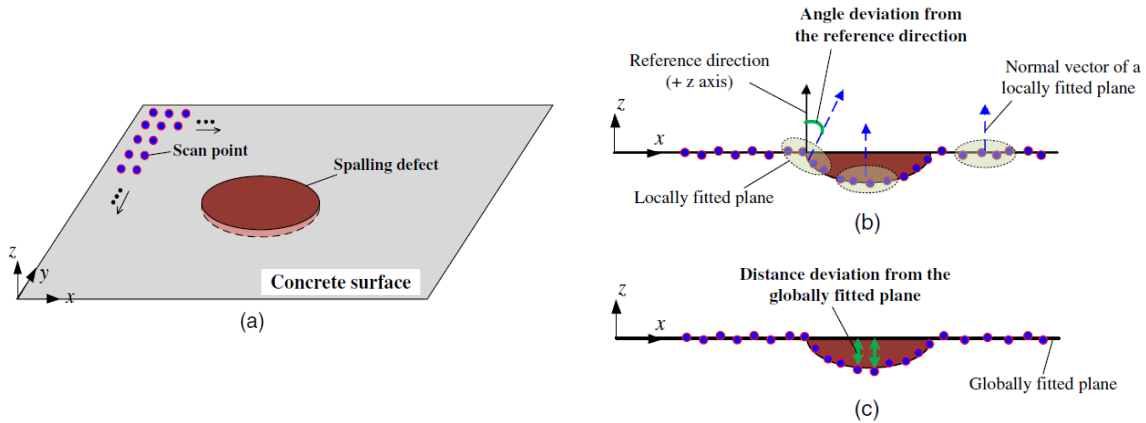


Figure 2-10. (a) Overview of scan points, (b) angle deviation, and (c) distance deviation (Kim et al., 2015).

Guldur et al. (2015) developed a strategy to define an appropriate threshold considering RGB color values of point clouds and/or intensity values to detect defects. Two methods were used for damage detection: graph-based and surface normal-based methods. After applying those methods, clustering was used for grouping the defect points into individual defects. The above-mentioned methods are sensitive to noise, uneven density, and complicated structures (Te et al., 2018). As shown in Figure 2-11, Truong-Hong et al. (2016) proposed a framework for bridge inspection with a LiDAR including five main phases: (1) preprocessing, (2) database system, (3) inspection, (4) management system, and (5) bridge assessment. To take advantage of both RGB color and  $x y z$  coordinates of points, first, the images and 3D point cloud were collected. After removing the irrelevant points manually and registration (Walsh et al., 2013), RGB color for each point was generated by image matching (Truong-Hong et al., 2016). Segmentation was based on a developed methodology by Walsh et al. (2013) to identify objects in the scene. Sharp features were located where the normal vector of point clouds changed significantly. Before segmentation, those features were detected by selecting a local neighborhood of points around each point using the  $k$ -nearest neighbors method (Rabbani et al., 2006). The size of neighbors should be small enough to estimate local properties and large enough to neglect the noises. In case of a high level of noise, a large size of neighborhoods is required. Then, the outlier points should be identified and removed to avoid the errors caused by scene movement or edge split. Next, curvature or surface variation was estimated based on the local neighborhood to be used in the following step, extraneous point detection (Pauly et al., 2002). Another cause of errors was the points with small-scale details, called “extraneous point”. Based on standard deviation and the average value of the curvature, these points can be removed, and then the Gauss map clustering was used to detect sharp features (Weber et al., 2010). Figure 2-12 shows a sample of crack detection at a bridge abutment using both RGB color and  $x y z$  coordinates of points to detect cracks and identify their size by calculating the distance between their edge points (Truong-Hong et al., 2016). As a result, the smallest detectable crack had 4.12 mm width, and the accuracy and precision were not calculated.

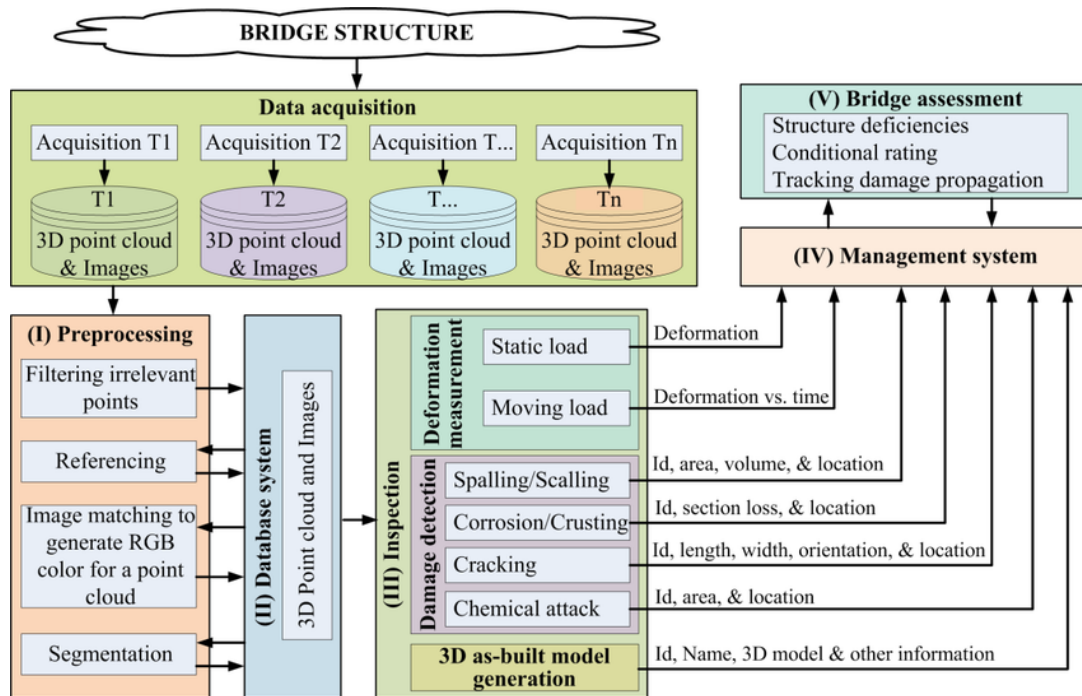


Figure 2-11. Framework for bridge inspection using TLS (Truong-Hong et al., 2016).

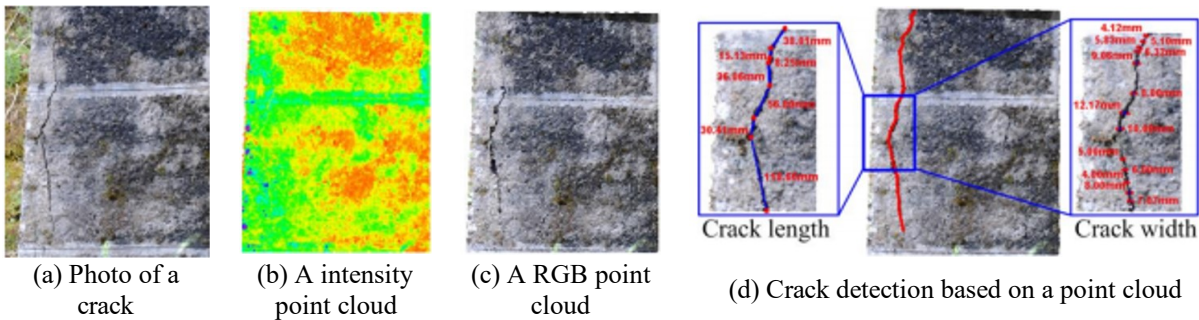


Figure 2-12. Crack detection at an abutment of Lungsdorf Bridge (Truong-Hong et al., 2016).

Moreover, Valença et al. (2017) integrated image processing and laser scanning considering the RGB value for each point to measure the crack characteristics (e.g., orientation, width, length). Figure 2-13 shows a sample of crack width and length characterization and measurements (Valença et al., 2017).

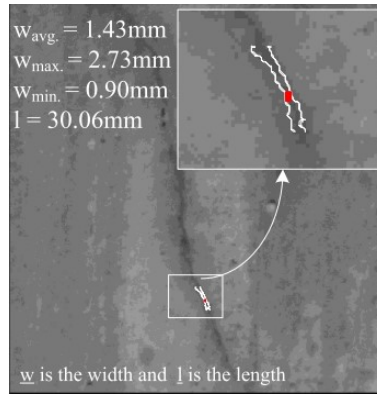


Figure 2-13. Sample of crack measurement (Valenca et al., 2017).

Severe spalling can lead to an immediate structure collapse or failure (Luckai et al., 2014). Therefore, several studies have been focused on detecting and quantifying this type of defect based on point clouds (Teza et al., 2009; Liu et al., 2010; Olsen et al., 2010; Mizoguchi et al., 2013; Guldur and Hajjar, 2017). Olsen et al. (2010) proposed a slicing analysis to quantify the spalling volume of large-scale structural elements based on the cross-sectional slices of the point clouds. There are other approaches based on damage-sensitive features such as curvature (Teza et al., 2009), distance and gradient (Liu et al., 2010), and surface normal (Guldur and Hajjar, 2017). Teza et al. (2009) proposed a Gaussian curvature-based surface detection method resulting in 52% and 48% accuracy in detecting damaged and undamaged areas, respectively. Guldur and Hajjar (2017) showed that data fusion between point clouds and vision data could lead to a promising solution to quantify the area and volume of the spalls.

Having an unordered nature, point clouds are mostly transformed into meshes (Kalfarisi et al., 2020) or voxels (Chen and Li, 2016) used in ML, particularly in supervised learning methods. Unlike images, ML-based methods on point cloud datasets have not made much progress for this purpose. As shown in Table 2-4, some studies proposed a two-step approach including applying the ML-based method on images for defect detection and then using point clouds for quantification of the detected defects. For example, McLaughlin et al. (2020) proposed a methodology to quantify spalls in concrete bridges in two steps: (1) the defect detection in the images using CNN, and (2) extracting the defects and quantifying their area based on the collected point clouds fused with labeled images using the Euclidean distance segmentation and RANdom SAMple Consensus (RANSAC) algorithms. Some methodologies were developed to reconstruct a bridge model using point clouds and then detect and/or quantify the bridge surface defects using image processing (Kim et al., 2018; Chow et al., 2021). Another proposed approach is using a non-DL-based method to detect the defects and clustering to group the points and extract individual defects (Guldur et al., 2015). Turkan et al. (2018) proposed an adaptive wavelet neural network-based approach using point clouds to detect concrete surface defects. The developed network was fed by X and Y components and estimated the Z coordinate for defect detection and localization. The effect of data compression was investigated and improved from 21.2% to 63.9% in crack detection. The errors of four specimens varied between 1.4 mm to 2.8 mm depending on the size of the crack and the resolution of the point clouds. It was concluded that this method was not practical for detecting hairline defects (Turkan et al., 2018).

The available information regarding the samples and results of the most relevant papers are summarized in Table 2-4. Not only the sizes and types of the defects but also the data acquisition circumstances vary from one paper to another. Therefore, the results of these papers are not comparable. For instance, detecting a large spall from a short distance leads to better outcomes compared to a small crack from a long distance. Moreover, one of the major obstacles to comparing the results of those studies is the lack of unified evaluation metrics (e.g., accuracy, recall, error). The same evaluation factors (e.g., error) are not comparable unless their definitions are the same. For example, although both Guldur and Hajjar (2017) and McLaughlin et al.(2020) used errors to report the study evaluation, those factors were calculated based on different formulas and reflected different concepts. However, most ML methods are evaluated based on unified performance metrics such as accuracy, IoU, recall, and precision.

Table 2-4. Summary of the most related point cloud-based concrete surface defect detection studies.

Reference	Type of defect					Features		Methodology		Results	Size of defects
	Defect	Crack	Spall	Corrosion	Mass loss	Geometrical	Color /Intensity	ML	Non-ML		
Teza et al. (2009)	✓	-	-	-	-	✓	-	-	PC	<ul style="list-style-type: none"> <li>Damaged accuracy (%): Circle (19 - 96) Rectangular (50-100) Square (50-100);</li> </ul>	<ul style="list-style-type: none"> <li>Sample of Defects: Circle (<math>r = 20 \text{ cm}</math>); Rectangular (<math>10 \times 20 \text{ cm}^2</math>) Square (<math>20 \times 20 \text{ cm}^2</math>)</li> </ul>
Olsen et al. (2010)	-	-	✓	-	-	✓	✓	-	PC	N/A	<ul style="list-style-type: none"> <li>Specimen size (<math>\text{cm}^3</math>): Column: <math>76 \times 92 \times 442</math> Beam: <math>56 \times 92 \times 777</math></li> <li>Spall size (<math>\text{cm}^3</math>): 8,700 - 24,500</li> </ul>
Liu et al. (2011)	-	-	-	-	✓	✓	-	-	PC	N/A	<ul style="list-style-type: none"> <li>Defect volumes (<math>\text{m}^3</math>): <math>2.71 \times 10^{-3} - 9.14 \times 10^{-3}</math></li> <li>Defect Areas (<math>\text{m}^2</math>): <math>4.27 \times 10^{-2} - 7.91 \times 10^{-2}</math></li> </ul>
Kim et al.(2015b)	-	-	✓	-	-	✓	-	-	PC	<ul style="list-style-type: none"> <li>Actual concrete panel (%): Accuracy: 68.3-91.3</li> <li>Lab test (%): Recall: 49.8-92.3; Pre: 83.4-97.1</li> </ul>	<ul style="list-style-type: none"> <li>Spall size (<math>\text{mm}^3</math>) Min: <math>10 \times 10 \times 4</math> Max: <math>100 \times 100 \times 7</math></li> </ul>
Truong-Hong et al. (2016)	-	✓	-	✓	-	✓	✓	-	PC	N/A	<ul style="list-style-type: none"> <li>Area (<math>\text{cm}^2</math>): 0.93– 2.06</li> <li>Dimensions (<math>\text{mm}</math>): W: 4– 12; L: 18 – 8</li> </ul>
Guldur and Hajjar (2016)	✓	-	-	-	-	✓	✓	PC	PC	<ul style="list-style-type: none"> <li>Classification (%): Accuracy: 93.5 Pre: 94.5</li> </ul>	N/A



Table 2-4. Summary of the most related point cloud-based concrete surface defect detection studies. (Cont.)

Reference	Type of defect					Features		Methodology		Results	Size of defects
	Defect	Crack	Spall	Corrosion	Mass loss	Geometrical	Color /Intensity	ML	Non-ML		
<i>Hou et al.(2017)</i>	✓	-	-	✓	-	-	✓	PC	-	<ul style="list-style-type: none"> <li>• Min clustering error (%): water staining: 10.1; Spall: 4.2</li> </ul>	N/A
<i>Guldur and Hajjar (2017)</i>	-	✓	✓	-	-	✓	✓	PC	PC	<ul style="list-style-type: none"> <li>• Error (%): Crack: 0.15 - 8.2; Spall: 0.43-1.62</li> </ul>	<ul style="list-style-type: none"> <li>• Dimensions (<i>in</i>): Crack: L:1.88 – 11.95; W:1 Spall: L: 11.7; W: 3.7</li> </ul>
<i>Valença et al. (2017)</i>	-	✓	-	-	-	✓	✓	-	Img PC	N/A	<ul style="list-style-type: none"> <li>• Crack size in images(<i>mm</i>): L: 30.1; W:1-4</li> </ul>
<i>Turkan et al. (2018)</i>	-	✓	-	-	-	✓	-	PC	-	<ul style="list-style-type: none"> <li>• Error (<i>mm</i>): 1.4 – 2.8</li> </ul>	<ul style="list-style-type: none"> <li>• Dimensions (<i>mm</i>) L: 9.8 - 35.1; W: 8.9 – 15.5</li> </ul>
<i>McLaughlin et al.(2020)</i>	-	-	✓	-	-	✓	✓	Img	PC	<ul style="list-style-type: none"> <li>• Error (%): 25.9</li> </ul>	<ul style="list-style-type: none"> <li>• Area (<i>cm</i><sup>2</sup>): 394 – 3,277</li> </ul>
<i>Nasrollahi et al. (2019)</i>	✓	-	-	-	-	✓	✓	PC	-	<ul style="list-style-type: none"> <li>• Testing defect accuracy (%): 14.9 – 87.5 (Avg: 69.5)</li> </ul>	<ul style="list-style-type: none"> <li>• Testing sample size: Max depth (<i>cm</i>): 1.5 – 8.1</li> </ul>
<i>Bahreini and Hammad (2021)</i>	-	✓	✓	-	-	✓	✓	PC	-	<ul style="list-style-type: none"> <li>• Testing recall (%): Spall: 89.77; crack: 55.2</li> <li>• Testing precision (%): Spall: 79.3; crack: 55.2</li> <li>• Testing IoU(%): spall: 72.72; crack: 44.68</li> </ul>	<ul style="list-style-type: none"> <li>• Testing sample size: Max depth (<i>cm</i>): 1.5 – 8.1</li> </ul>
<i>This research</i>	-	✓	✓	-	-	✓	✓	PC	-		

ML: Machine Learning; PC: Point Clouds; Img: Image; L: Length; W: Width;

### 2.5.5 Comparison between Image and Point Cloud-based Defect Semantic Segmentation

Some of the recent image-based crack detection methods using ML are tabulated in Table 2-5. Comparing the results of these papers is not possible unless the same sensor specifications (e.g., resolution) have been considered for data acquisition. The effect of the different environmental conditions (e.g., lighting) cannot be ignored even if the same sensor settings have been used. It can be concluded that the higher accuracy cannot be a reliable indication of the better methodology unless the same datasets are used. Therefore, comparing the results in Table 2-4 and Table 2-5 is not possible unless the comparison is based on the same input dataset.

Table 2-5. Image-based semantic segmentation of surface defect using DL.

Reference	Type of defect			Domain		Methodology	Best Results (%)	Dataset size
	Defect	Crack	Spall	Bridge	Concrete			
<i>Lee et al. (2019)</i>		✓		✓		CNN	Precision: 87 Recall: 74 OA: 98	242 cracks
<i>Lopez Droguett et al. (2020)</i>		✓		✓		DenseNet 13	Avg. mIoU: 94 Avg. recall: 98	256,115 cracks 153,316 no-crack
<i>Hoskere et al. (2020)</i>		✓	✓		✓	Encoder-Decoder	IoU: crack 68 spall: 81	341 cracks 324 spalls
<i>Fu et al. (2021)(2021)</i>		✓		✓		ResNet (DeepLabv3+)	mIoU: 65	5,000 cracks
<i>Mohammed Abdelkader et al. (2021)</i>			✓	✓		NN	OA: 90 F1: 92	60 spalls
<i>Wang et al. (2022)</i>		✓			✓	ResNet, DenseNet	IoU: 63 Recall: 68	2,446 images

## 2.6 UAV Path Planning

UAV flight path planning can be an approach to automate and optimize the data collection process. A UAV can be equipped with camera (Janoušek and Faigl, 2013; Bircher et al., 2015, 2016b; Phung et al., 2017b; Rafanavicius et al., 2017; Freimuth and König, 2018), sonar camera (Englot and Hover, 2010; Hover et al., 2012), or LiDAR (Yoder and Scherer, 2016; Nasrollahi et al., 2018) to collect data. In path planning algorithms, which are based on different number of Degrees of Freedom (DoF) (i.e., x, y, z, yaw), several factors such as speed (Bircher et al., 2015), wind, and battery capacity (Guerrero and Bestaoui, 2013) can be considered.

In path planning, there are two exploration methods based on the existence of the prior model of the object to be inspected: model-based and non-model-based methods (Almadhoun et al., 2016). In non-model-based methods, no initial knowledge about the object is available. This method can be applied in reconstructing a 3D object, structure (Yoder and Scherer, 2016), or environment

(Quin et al., 2013). In model-based exploration, the model must be available in advance, either as a BIM or 3D CAD model. The main objective of model-based VPI selection is to provide full visibility of the expected area. Figure 2-14 represents the general process of UAV path planning for a model-based path planning method, which includes three main phases: (1) Providing the model and defining the constraints (e.g., FoV, DoFs); (2) VPI selection considering some criteria (e.g., overlapping views, criticality) (Scott, 2009). The Model-based methods are categorized into set theory methods, graph theory methods, and computational theory methods (Scott et al., 2003). Set theory (Scott, 2009) and computational methods (Englot and Hover, 2010; Hover et al., 2012; Bircher et al., 2015) are used more often compared to the graph theory (Almadhoun et al., 2016). Art Gallery Problem (AGP) solvers, which are computational methods, can be used to find the minimum set of VPIs that cover the inspected area's surface. (3) Path planning, which is finding the shortest path passing through the selected VPIs by solving TSP. Moreover, some algorithms such as Rapidly-exploring Random Tree (RRT), RRT\*, and A\* can be used to find a short obstacle-free path between two VPIs.

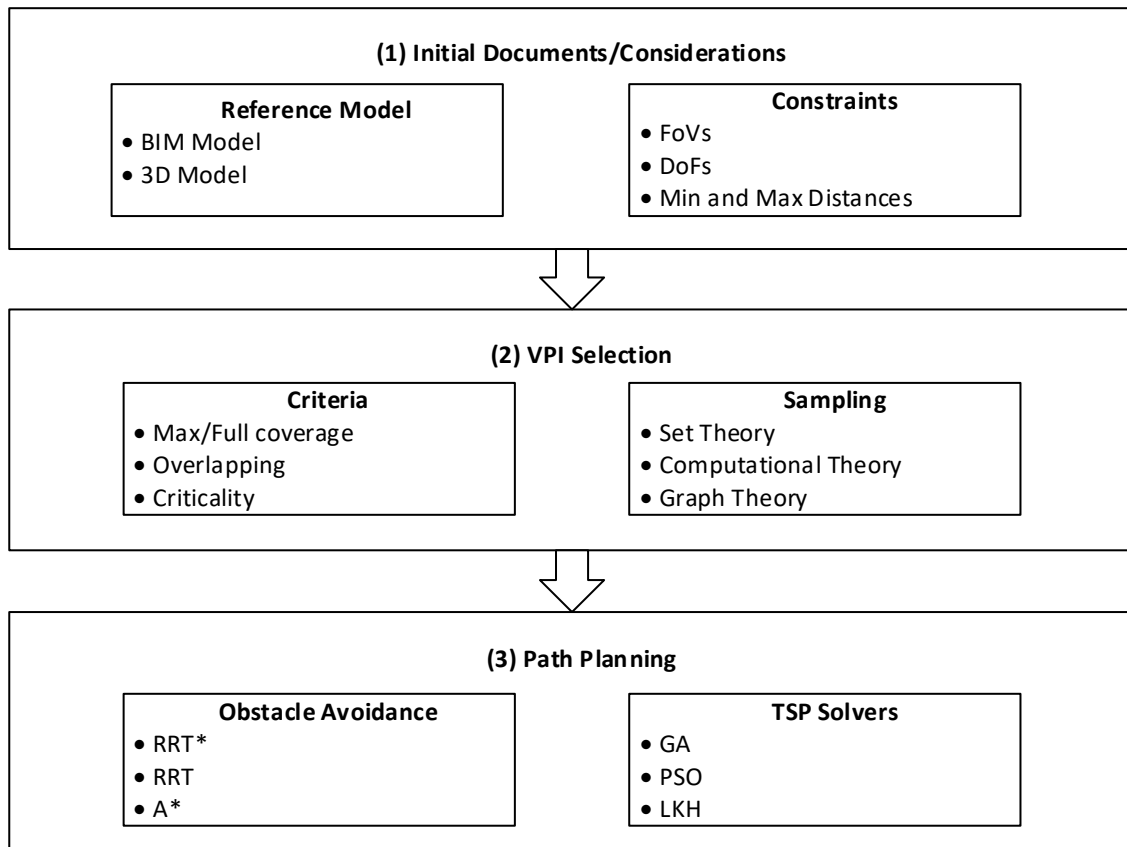


Figure 2-14. General process of UAV model-based path planning method.

In the case of surveying or inspection using UAV, path planning has three main objectives:

- (1) *Collision avoidance* is the initial factor considered in all path planning algorithms. Collisions may be dynamic or static. For UAV inspection path planning, the static obstacles are considered including the body of the inspected structure and surrounding objects.
- (2) *Minimum time-of-flight* is an objective of path planning. The main part of this criteria is the time spent moving from one point to another (changing the position of the UAV), and



along the smallest expected total distance. The main goal of this algorithm is minimizing the path length, which is calculated based on Equation 2-3.

$$f(n) = g(n) + h(n) \quad \text{Equation 2-3}$$

where  $g(n)$  is the actual distance between the node  $n$  and start point, and  $h(n)$  is the estimate of the distance between  $n$  and the end point. If two nodes have the same  $f(n)$ , the priority is given to the node with the smallest  $h(n)$ . The key elements of the A\* algorithm are OPEN list, CLOSED list, and  $f(n)$ . The OPEN list is established to store the candidate nodes, which will be detected, while the CLOSED list is established to store the selected nodes of the final path. In accordance with the evaluation value, the nodes in the neighborhood are arranged from lowest to highest  $f(n)$  into the OPEN list. The node with the minimum evaluation value is transferred to the CLOSED list. Then the transferred node is regarded as the current node, and a new list is established based on the calculated  $f(n)$ . The above process is repeated until the path reaches the target node (Fu et al., 2018). Figure 2-16 shows the schematic diagram of the A\* algorithm path search.

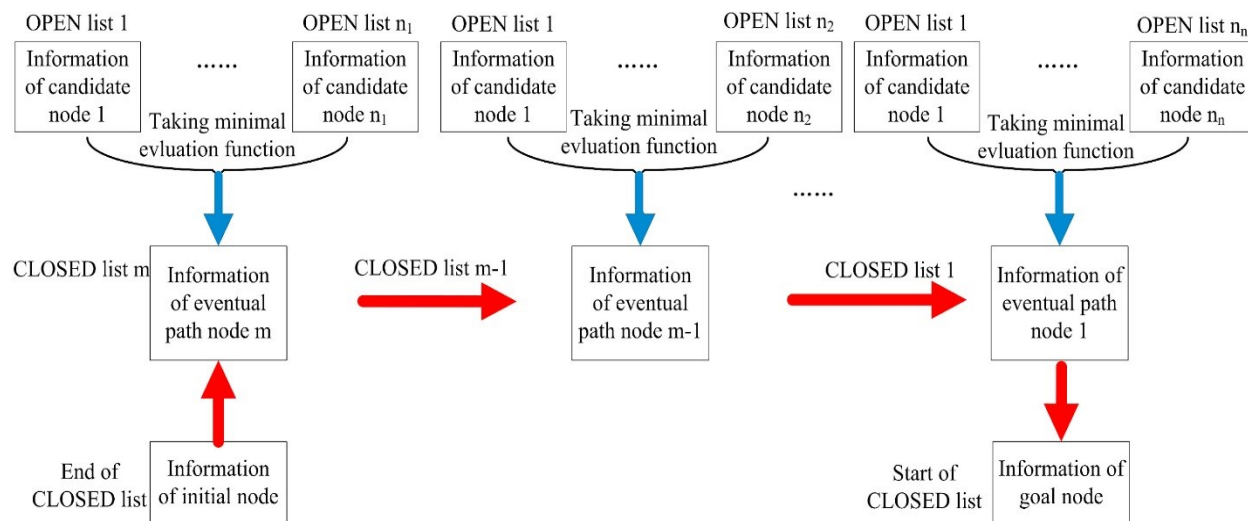


Figure 2-16. The schematic diagram of the A\* algorithm path search (Fu et al., 2018).

### 2.6.1.3 Rapidly-exploring Random Trees

RRT is a popular single-query sampling-based planner. It is based on growing a tree of configurations to cover the state of space. The process starts from the initial point, continues over adding random leaves to the tree, and finishes when the predefined end point is reached (LaValle and Kuffner Jr, 2001). According to the pseudocode of RRT (Figure 2-17), first,  $x_{rand}$  is picked and connected to  $x_{init}$ . Then, the next  $x_{rand}$  is selected. After finding the nearest neighbor point of the tree ( $T$ ) to  $x_{rand}$ , called  $x_{nearest}$ , a vertex between  $x_{rand}$  and  $x_{nearest}$  is considered and a new point ( $x_{new}$ ) is selected with the distance of  $\epsilon$  from its  $x_{near}$  on the vertex. If some obstacles appear over connecting  $x_{nearest}$  and  $x_{rand}$ , that point is unacceptable. Otherwise, an edge (E) and a vertex (V) would be added to the tree, making a new leaf. A sample of RRT operation is represented in Figure 2-18.

### RRT

```
1  $V \leftarrow \{x_{init}\}; E \leftarrow \emptyset;$   
2 for  $i = 1$  to  $n$  do  
3    $x_{rand} \leftarrow \text{SampleFree}();$   
4    $x_{nearest} \leftarrow \text{Nearest\_Neighbor}(x_{rand}, T = (V, E));$   
5    $x_{new} \leftarrow \text{Steer}(x_{rand}, x_{nearest});$   
6   if  $\text{ObstacleFree}(x_{nearest}, x_{new});$   
7      $V \leftarrow V \cup \{x_{new}\}$   
8      $E \leftarrow E \cup \{x_{nearest}, x_{new}\}$   
9 Return  $T = (V, E)$ 
```

Figure 2-17. Pseudocode of RRT algorithm.

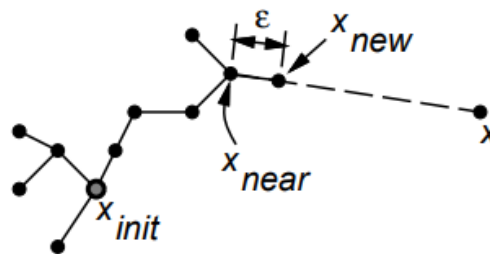


Figure 2-18. Extended operation (Kuffner and LaValle, 2000).

Several improvements have been made to the RRT algorithm. Dual RRT Planner can be used when both the start  $x_{init}$  and the end point  $x_{end}$  of the path are determined. In this case, two paths starting from  $x_{init}$  and  $x_{end}$  are found simultaneously and expanded until both trees reach each other (Figure 2-19).

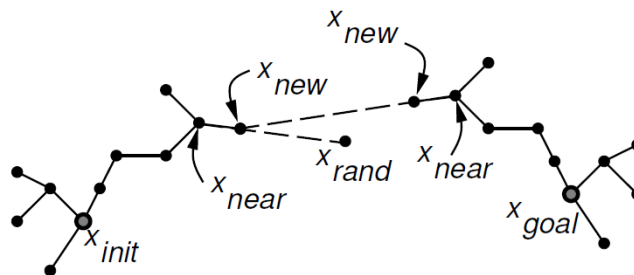


Figure 2-19. Dual RRT (LaValle and Kuffner Jr, 2001).

The main advantage of using an RRT is that it is a feasible solution for searching high-dimensional spaces (Kulling, 2009), and non-holonomic constraints could be taken into account. A non-holonomic system can determine a differential relationship between state and inputs. It means that the history of the previous states is needed to determine the current state (Dong, 2015). This method can cover high DoFs, and steering is not required (LaValle and Kuffner Jr, 2001). Moreover, one of the desirable properties of this algorithm is its simplicity because it uses few parameters during the process.

### 2.6.1.4 Rapidly-exploring Random Trees Star (RRT\*)

RRT\* is very similar to RRT; however it can generate smoother path compared to RRT using two key modifications: selecting least cost parent and rewiring tree, which are shown in Figure 2-20 and Figure 2-21. The following pseudocode describes the RRT\* algorithm (Nasir et al., 2013).

Same as RRT, RRT\* starts by sampling a random point ( $x_{rand}$ ) in C space, followed by finding the nearest neighbor ( $x_{near}$ ) to the  $x_{rand}$  and placing a new node ( $x_{new}$ ) at a radius distance of  $\Delta q$  from  $x_{near}$  in the direction of  $x_{rand}$  if the connection between two nodes is collision-free. Then, in RRT\*, neighboring points of  $x_{new}$  are checked within the area of a ball of  $\beta(\log(n)/n)$  radius, where  $n$  and  $\beta$  represent the dimension of the given space and the constant based on environment, respectively. Afterward, a parent node, which is a neighboring node that results in a shorter path to the start node, is found. Another modification is rewiring the tree. After successfully adding a new point and selecting the parent node, neighbors are checked once again. If considering  $x_{new}$  as a parent for a neighbor node decreases the cost, the path from that node to the start node is rebuilt as shown in Figure 2-20(d) (Noreen et al., 2016).

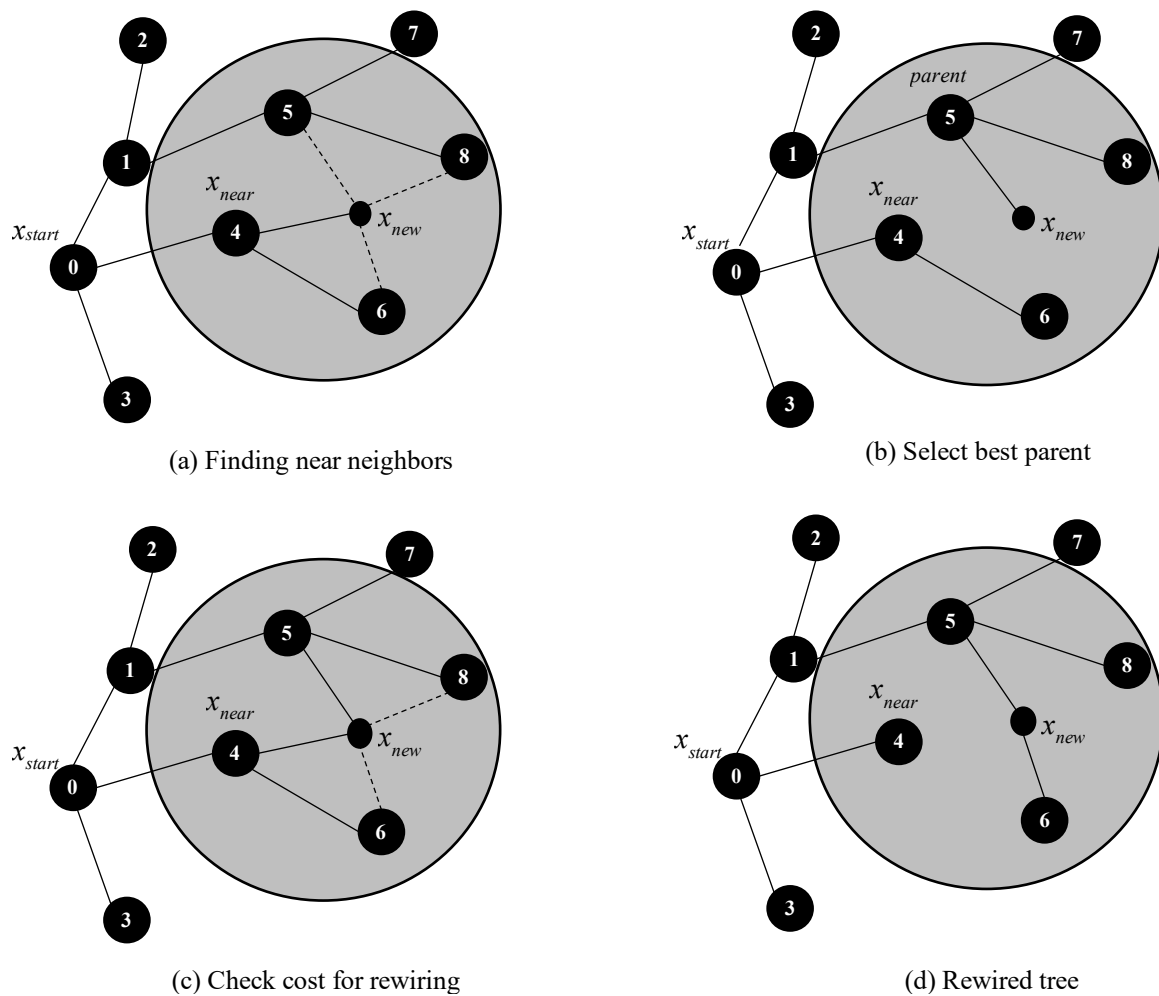


Figure 2-20. Near neighbor search and rewiring operations in RRT\* (Noreen et al., 2016).

```

1   $T \leftarrow \text{InitializeTree}();$ 
2  for  $k = 1$  to  $K$  do
3       $x_{rand} \leftarrow \text{RANDOM\_STATE} ();$ 
4       $x_{nearest} \leftarrow \text{NEAREST\_NEIGHBOR} (x_{rand}, T);$ 
5       $(z_{new}, u_{new}, T_{new}) \leftarrow \text{STEER} (x_{rand}, x_{nearest});$ 
6      if  $\text{ObstacleFree} (z_{near})$  then
7           $X_{near} \leftarrow \text{NEAR}(x_{new}, u, |V|);$ 
8           $x_{min} \leftarrow \text{CHOOSE\_PARENT} (X_{near}, x_{nearest}, x_{new}, z_{new});$ 
9           $T \leftarrow \text{INSERT\_NODE} (x_{new}, x_{min}, T);$ 
10          $T \leftarrow \text{REWIRE} (X_{near}, x_{min}, x_{new}, T);$ 
11 Return  $T$ 

```

Figure 2-21. Pseudocode of RRT\* algorithm (Nasir et al., 2013).

### 2.6.1.5 Comparison between Path Planning Algorithms

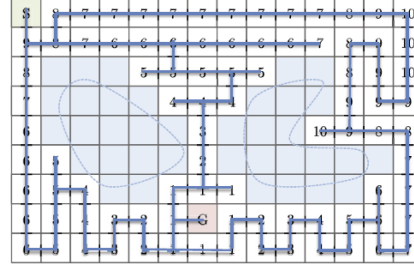
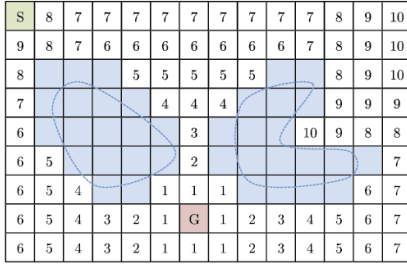
Three algorithms, Bug, A\*, and RRT, were compared from different aspects of views such as optimality, completeness, query type, DoFs, and so on by Langari (2015). Bug algorithm is a simple planner which can take place in 2D and is practical for the object with two DoFs. A\* is applied in 3D space (Marzouk and Ali, 2013). Nav-mesh can be used in the Unity 3D game engine to apply A\*. Unlike these two algorithms (Bug and A\*), RRT applies to the cases with two DoFs and more. However, this sampling-based algorithm cannot provide an optimal result (AlBahnassi and Hammad, 2011)

Completeness is the state of finding the path, and Bug planner is able to complete the path in a reasonable time (AlBahnassi and Hammad, 2011). For the A\* algorithm, depending on the size of graph grids, it may be complete or not.

### 2.6.2 Coverage Path Planning Algorithms

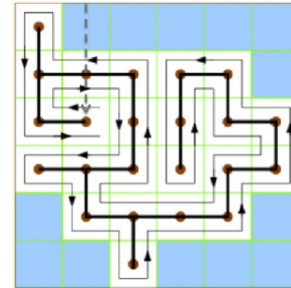
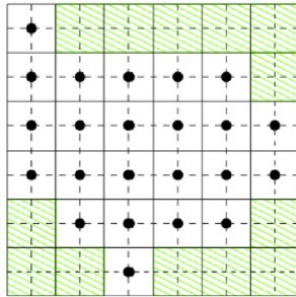
The coverage algorithms can be heuristic or complete based on the probability of coverage completeness in the free space (Galceran and Carreras, 2013). In most coverage path planning algorithms, the target space is decomposed into small pieces called cells, which can have square, triangle (Oh et al., 2004), or other shapes. Although using square cells leads to a lower resolution compared to triangular ones, it is suitable for mobile robotics path planning because most robots cannot adjust their movements to very small cells (Galceran and Carreras, 2013). Consequently, high-resolution grids are not necessary. Due to the simplicity of creating a grid map and marking the covered area, these algorithms are widely used (Thrun, 1998). However, to achieve a good result and accurate localization, a high capacity of the memory is required for storing high-resolution cells (Castellanos et al., 1997). Figure 2-22 and Figure 2-23 represent two grid-based coverage path planning methods using wavefront and spanning trees algorithms, respectively.





(a) Wavefront distance transform for the selection of the start position (S), goal position (G) (b) Coverage path generated using the wavefront distance transform with the selection of the start position (S)

Figure 2-22. Coverage path planning using wavefront algorithm (Galceran and Carreras, 2013).



(a) Approximate cell decomposition in mega cells and robot-sized cells (b) Coverage path generated with the Spanning trees algorithm

Figure 2-23. Coverage path planning using spanning trees algorithm (Galceran and Carreras, 2013).

### 2.6.3 Traveling Salesman Problem

TSP is a Nondeterministic Polynomial (NP)-hard problem in combinatorial optimization aiming to find the shortest path if a traveling salesman wants to visit each city exactly once and return to the first city (Applegate et al., 2011). Complete historical development of this and related problems can be found in Hoffman et al. (2013), Applegate et al. (2011), and Cook (2014). Equations 2-4 introduces the  $x_{ij}$  variable to formulize TSP on  $m$  points.

$$x_{ij} = \begin{cases} 1 & \text{if the edge } i \rightarrow j \text{ is in the tour} \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation 2-4}$$

where  $x_{ij}$  represents the path between points (cities)  $i$  and  $j$ . Moreover, only one edge can be made at each point (city). The problem and the additional constraints can be formulated as follow.

$$\begin{aligned} \min & \sum_{j=1}^m \sum_{i=1}^m l_{ij} x_{ij} && \text{Equation 2-5} \\ \text{s.t.} & \sum_{j=1}^m x_{ij} = 1 \text{ for } i = 1, \dots, m \\ & \sum_{i=1}^m x_{ij} = 1 \text{ for } j = 1, \dots, m \end{aligned}$$

where  $l_{ij}$  is the number of points (cities) and the length of the path between  $I$  and  $j$  (Hoffman et al., 2013). The constraints demonstrate that every point of the path must have only one path going towards it and another one going away from it.

Several optimization methods are proposed for this problem, such as GA (Grefenstette et al., 1985; Hussain et al., 2017), Particle Swarm Optimization (PSO) (Phung et al., 2017b), and Ant Colony Optimization (ACO) (Yang et al., 2018).

### 2.6.3.1 Genetic Algorithms

GAs are evolutionary algorithms used to solve optimization problems inspired by population biologic genetics. Although using this approach does not guarantee the optimal solution, it leads to a good near-optimal result in reasonable time (Hussain et al., 2017). A GA is used for solving NP-hard problems such as TSP. The fitness function in this method minimizes the path's length, which is computed based on the following formula.

$$\min \sum_{i=1}^{n-1} l_i \quad \text{Equation 2-6}$$

where  $n$  is the total number of points, which the path should pass through them.  $l_i$  is the path between two points, which is selected from a set of  $n(n-1)$  paths between all points. The main GA operators are selection, crossover, and mutation.

### 2.6.3.2 Particle Swarm Optimization Algorithm

PSO is an intelligent optimization algorithm that belongs to a class of optimization algorithms called metaheuristics. PSO is based on the paradigm of swarm intelligence, and it is inspired by the social behavior of some animals such as birds and fishes. PSO is a simple optimization algorithm, and it is successfully applied to many applications in various fields of science and engineering like ML, image processing, data mining, and many other fields. Despite its simplicity, which is one of the key points of the PSO, it is a very powerful algorithm. Initially, PSO was introduced by Eberhart and Kennedy (1995). Over the last two decades, PSO has become one of the most useful and popular algorithms to solve optimization problems. Based on the behavior of bird flocking, a group of birds searching for a piece of food is considered. They only know the distance to the food, not its location. So, the most effective strategy is to follow the bird which is nearest to the food.

PSO contains "particles" while each is a single solution with a fitness value. Each particle, which has a memory about the best position and the neighborhood's positions, moves with the speed of  $V$  between its positions. The framework of a PSO is demonstrated in Figure 2-24.

```

procedure PSO
  Initialize a population of particles
  do
    for each particle  $p$  with position  $x_p$  do
      if ( $x_p$  is better than  $pbest_p$ ) then
         $pbest_p \leftarrow x_p$ 
      end_if
    end_for
    Define  $gbest_p$  as the best position found so far by any of  $p$ 's neighbors
    for each particle  $p$  do
       $v_p \leftarrow \text{Compute\_velocity}(x_p, pbest_p, gbest_p)$ 
       $x_p \leftarrow \text{update\_position}(x_p, v_p)$ 
    end_for
  while (a stop criterion is not satisfied)

```

Figure 2-24. Framework of a PSO algorithm (Goldberg et al., 2006).

### 2.6.3.3 Lin-Kernighan Heuristic Algorithm

The Lin-Kernighan Heuristic (LKH) algorithm is a local optimization algorithm, which works based on exchanging one path to another. Given a feasible path, in each iteration the specific number of sub tours ( $\lambda$ ) are exchanged to decrease the length of the current path. This process repeats until no exchange causes an improvement. As  $\lambda$  increases, in a longer computation time, better results can be achieved. Moreover, due to its time complexity of  $O(n^\lambda)$ , increasing the number of points rapidly increases the number of operations. Therefore, this method is mostly practical for paths with small number of points (Helsgaun, 2000).

### 2.6.3.4 Comparison between GA and PSO

Most of the evolutionary techniques contain four main steps: random generation of an initial population, fitness value calculation for each subject, and reproduction of the population (Kachitvichyanukul, 2012). Both GA and PSO algorithms update the population and search for the optimum result using random techniques. They also do not guarantee complete success. Unlike GA, PSO does not have crossover and mutation, and the particles are updated by themselves with internal velocity (Borna and Khezri, 2015). The information-sharing mechanism is different in PSO and GA. In GA, chromosomes' population moves as a group, while sharing information towards an optimal area. However, PSO has a one-way information sharing mechanism from only  $gbest$  to others. It is still argued that which method has the higher convergence speed, GA (Gharib et al., 2015) or PSO (Goldberg et al., 2006).

## 2.6.4 UAV Path Planning for Inspection

As shown Table 2-6, several efforts have been made to use automated structural inspection systems using a UAV equipped with a vision-based sensor. Bircher et al. (2015) proposed a path planning algorithm using LKH and RRT\*, which is based on a mesh representation of the environment. First, one VPI is sampled for each triangle in the mesh considering the limitations (i.e., FoV,  $d_{min}$ ,  $d_{max}$ ). Then, the cost of moving from one VPI to another (path length), is computed, and RRT\* is used in case of the presence of an obstacle between two VPIs. Based on the calculated cost matrix, the initial path is found using LKH. The short inspection path is computed using an alternating two-step optimization algorithm in their method. The new set of VPIs is selected in each iteration to minimize the path length and the rotation duration using convex problem formulation subjected

to visibility. Figure 2-25 shows how the boundaries of the VPI sampling space (gray planes) can be found based on the incidence angle, minimum angle (green plane), and maximum distance between the mesh and the equipment (red plane). The calculated path for turbine inspection is shown in Figure 2-26, which is a zigzag path.

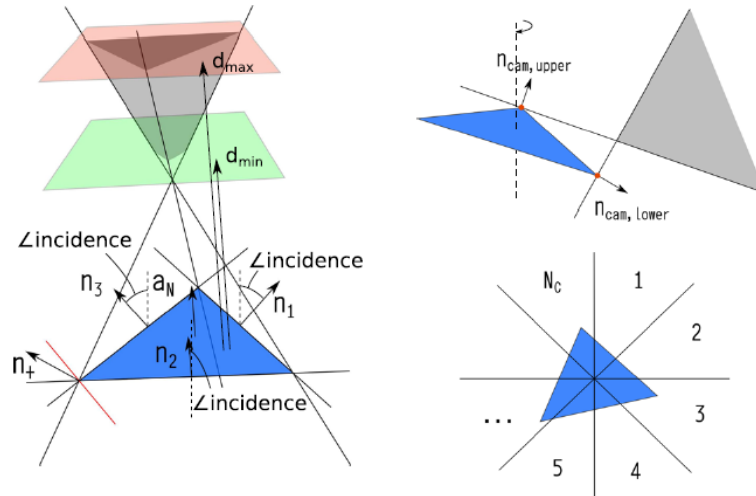


Figure 2-25. Finding boundaries of the sampling space (Bircher et al., 2015).

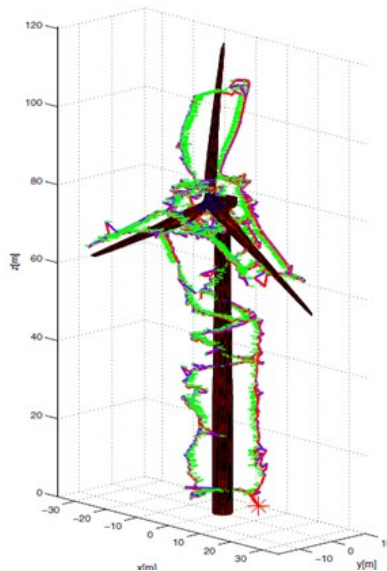


Figure 2-26. Path planning using LKH and RRT\* (Bircher et al., 2015).

Phung et al. (2017a) also proposed a TSP-based path planning method for structural inspection using a camera-equipped UAV and VPIs were sampled considering FoV, focal length, sensor size, and overlapping percentage (Figure 2-27). Unlike Bircher’s research (2016), A\* and PSO were used instead of RRT\* and LKH, respectively.

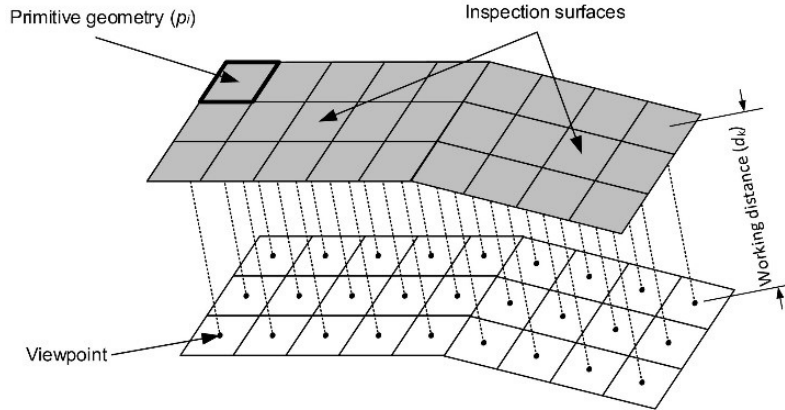


Figure 2-27. Generating VPIs (Phung et al., 2017a).

On the other hand, Freimuth and König (2018) proposed a path planning method based on the A\* algorithm. Dronecode Foundation software (Dronecode Project Inc., 2017) was used in this research. Despite of its ability to consider flight dynamics (e.g., wind and UAV speed) during flight simulation, constant speeds were considered in case studies. The main objective of their research was combining BIM and Dronecode to increase the degree of the automation of path planning in building inspection.

Alves et al. (2020) focused on finding a short path between predefined locations in a building in real-time. Compared to A\* and RRT, their proposed Fast Path Planning Method (FPPM) was more practical for real-time path planning. Although FPPM cannot find the optimal path, the computation time is much less than A\* and RRT. Cao et al. (2020) proposed a path planning method in complex 3D environments like bridges and airplanes. The sampled viewpoints were set at distance  $D$  away from the surface point in the surface normal direction, as shown in Figure 2-28, where the dotted curve and the red dot represent the inspected surface and the sub-sample points, respectively. Also, the grey and red vectors indicate the normal directions and the sample viewpoints direction at distance  $D$ . Then, a global TSP was solved using Google OR-Tools (Perron and Furnon, 2020). The distance matrix was calculated based on the Euclidean distance between two points in this method. In case of any obstacle between two points, the path length was set to infinity. Then, the detailed coverage was calculated, and a collision-free path in each individual subspace was found solving local TSPs. Figure 2-29 shows an example of a test result where the orange lines and yellow rectangles indicate the path and the subspaces, respectively. The optimization method used in OR-Tools was not mentioned. However, this method may not be time-efficient for big-scale structures because solving global and local TSPs several times may need long computation time. Moreover, overlaps were not considered in selecting viewpoints.

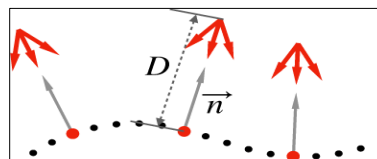


Figure 2-28. Sampling viewpoints (Cao et al., 2020).



Table 2-6. Summary of the literature review of UAV path planning for inspection.

Reference	Path planning method		Sensor			Reference model				Conditions				Consideration for VPI generation				VPI generation	Path generation	Application		
	Modeled-based	Non-modeled based	Camera	Sonar camera	LiDAR	3D CAD model	BIM	2D plan	3D point cloud	Images	Vibration	Speed	Wind	Battery capacity	DoF	Full coverage	Overlapping				Criticality	Optimal path
<i>Englot and Hover (2012)</i>	✓	-	-	✓	-	✓	-	-	-	-	-	-	-	-	4	✓	-	-	✓	AGP (computational) Triangular mesh	Improved RRT* Redundant roadmap	Ship hull inspection
<i>Guerrero and Bestaoui (2013)</i>	✓	-	-	-	-	✓	-	-	-	-	-	✓	✓	✓	4	✓	-	-	✓	Distance-based meshing method (triangular meshing)	TSP solver: Zermelo Algorithm	Structure inspection
<i>Janoušek and Faigl (2013)</i>	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	3	✓	-	-	✓	Simultaneously with the path optimization (the initial interest points are selected manually)	PRM	City-like environment inspection
<i>Dornhege et al. (2016)</i>	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	3	✓	-	-	✓	Set cover problem	TSP solver: TFD/LKH	Covering 3D environment
<i>Bircher et al.(2015)</i>	✓	-	✓	-	-	✓	-	-	-	-	-	✓	✓	-	4	✓	-	-	-	AGP (computational) Triangular mesh	LKH & RRT* & optimization of the generated path	Structure inspection
<i>Yoder and Scherer (2016)</i>	-	✓	✓	-	✓	-	-	-	-	-	-	-	-	-	4	✓	-	-	-	Next best view	Next best view	Bridge modeling
<i>Bircher et al.(2016b)</i>		✓	-	-	-	-	-	-	-	-	-	✓	✓	-	4	✓	-	-	✓	Next best view	Next best view	Bridge and indoor building modeling
<i>Phung et al. (2017a)</i>	✓	-	✓	-	-	✓	-	-	✓	-	-	-	-	-	3	✓	✓	-	✓	Sampled w.r.t FoV & overlapping (cube cells)	TSP solver: PSO and A*	Bridge and indoor building inspection

Table 2-6. Summary of the literature review of UAV path planning for inspection. (Cont.)

Reference	Path planning method		Sensor			Reference model				Conditions				Consideration for VPI generation				VPI generation	Path generation	Application		
	Modeled-based	Non-modeled based	Camera	Sonar camera	LiDAR	3D CAD model	BIM	2D plan	3D point cloud	Images	Vibration	Speed	Wind	Battery capacity	DoF	Full coverage	Overlapping				Criticality	Optimal path
Rafanavicius et al. (2017)	✓	-	✓	-	-	-	-	✓	-	-	-	-	-	-	3	-	-	-	-	Pre-defined manually	Hierholzer's algorithm (Mission planner software)	Power line inspection
Luo et al. (2017)	✓	-	-	✓	-	-	-	-	-	✓	-	-	-	-	3	✓	-	-	✓	Pre-defined manually	Bezier Curve-PSO Based Algorithm	PV farm inspection
Yang et al. (2018)	✓	-	-	✓	-	-	-	-	-	✓	-	✓	-	-	2	✓	✓	-	-	Sampled w.r.t the FoV & overlapping (cube cells)	ACO	Surveying
Freimuth and König (2018)	✓	-	✓	-	-	-	✓	-	-	-	-	-	✓	-	4	✓	-	-	✓	Pre-defined manually	A*	Building inspection
Almadhoun et al. (2019)	✓	-	✓	-	-	-	-	✓	-	-	-	-	-	-	3	✓	-	-	✓	Initial VPIs based on FoVs and range limitation (resampling after coverage evaluation)	LKH and RRT*	Inspection / 3D reconstruction
Alves et al. (2020)	✓	-	✓	-	-	-	-	✓	-	-	-	-	-	-	3	-	-	-	-	Pre-defined manually	FPPM and GA	Indoor inspection
Cao et al. (2020)	✓	-	✓	-	-	-	-	-	✓	-	-	-	-	-	4	✓	-	-	✓	Pre-defined manually	LKH	Inspection
Hamledari et al. (2021)	✓	-	✓	-	-	-	✓	-	-	-	-	-	-	-	3	✓	-	-	✓	Pre-defined manually	TSP solver: ACO and Dijkstra algorithm	Indoor inspection



## 2.7 Summary

Condition assessment of bridges is one of the main concerns of bridge managers. Therefore, choosing the most appropriate method in all steps of bridge inspection (i.e., data acquisition and data analysis) results in high efficiency. Detecting surface defects, such as cracks and spalls, is important for assessing bridge conditions. Therefore, several studies have been focused on developing efficient technologies to detect surface defects. After comparing some technologies in visual inspection, a LiDAR-equipped UAV is found one of the best equipment for scanning the bridge surfaces from different aspects such as safety, time-efficiency, and accuracy.

The first step of the bridge inspection is data acquisition. Several parameters related to LiDAR and UAV can significantly affect the quality of data collection. Flying the UAV at a close distance of the bridge surface with a perpendicular view allows collecting accurate data. Due to the existence of obstacles, the UAV may not be able to reach some parts of the surface or to scan it from a perpendicular view. Therefore, many methods were proposed to find a collision-free path with maximum coverage. TSP can be solved to optimize the path length, which includes two main steps: selecting the best VPIs and finding the path passing through them. According to our review, all previous methods selected the points based on the sensor characteristics, and without considered the level of criticality of the surface. Considering this factor in selecting VPIs helps increase the inspection quality of high-risk areas with perpendicular and overlapping views. Therefore, the condition assessment will be more reliable based on the collected point clouds.

Although point cloud-based DL methods have been widely used in many fields, only limited researchers have developed surface defect detection approaches based on those models. Therefore, one popular point cloud-based semantic segmentation method, PointNet++, is selected as the basic network to train a defect detection model.

Furthermore, a large dataset plays an important role in improving the performance of a DL network. Since there is a lack of dataset for concrete surface defect detection, it is necessary to provide a publicly available concrete surface defects dataset for point cloud-based semantic segmentation learning.

## CHAPTER 3 POINT CLOUD-BASED CONCRETE SURFACE DEFECT SEMANTIC SEGMENTATION

### 3.1 Introduction

As explained in Section 1.4, the defects should be detected after scanning the bridge and collecting 3D point clouds. This chapter aims to take advantage of both point clouds and DL-based semantic segmentation to detect concrete surface defects by applying a modified version of PointNet++ (Qi et al., 2017b) called Surface Normal Enhanced PointNet++ (SNEPointNet++). The structure of this chapter is as follows. First, all the aspects, which are considered in SNEPointNet++, are explained and the framework of the two adjusted networks are elaborated in Section 3.2.3. In Section 3.3, the adapted PointNet++ and SNEPointNet++ are validated using the point clouds, which are collected from four bridges in Montreal. The obtained experimental results and their discussion are presented in Section 3.4. Finally, the conclusions of the proposed method are outlined.

### 3.2 Proposed Method

#### 3.2.1 Aspects Considered in the Method

Considering that the PointNet++ semantic segmentation method was originally designed to detect indoor building elements, it cannot be applied in an off-the-shelf manner to the task of concrete surface defect detection. As such, four main aspects differentiate this study from the original one (Qi et al., 2017b), as explained below.

(a) *Creating a large high-quality point cloud dataset*

A sufficient point cloud dataset is a key issue in the point cloud-based semantic segmentation of surface defects. Unlike point clouds, many images of concrete cracks and spalls are available online, which can be used for training a DL model. Strict safety regulations, availability, and accessibility complications in scanning a bridge are the main reasons for the lack of point cloud datasets. Therefore, this study aims to provide a high-quality point cloud dataset to be used in surface defect detection by different groups of researchers. Furthermore, several data augmentation approaches such as shifting, flipping, and rotating can be applied to generate a bigger dataset based on the available one. In this work, the collected point cloud datasets were flipped, however, shifting and rotation were not applied because they can alter the characteristics of some defects (e.g., location and orientation of shear cracks).

(b) *Redefining the features of points to better capture the main features of defects*

In the original PointNet++, the objects in each class are very similar. For example, the heights of walls in an office are mostly the same and similar chairs are used in each office as well. However, in defect detection, each defect has a unique shape, which is very different from other defects. These variations make the learning process difficult. Nevertheless, all cracks and spalls have two main features; they are deeper and darker than their adjacent areas. Therefore, if trained properly, the network can learn to use these features (i.e., point location  $(x_i, y_i, z_i)$  and color (R,G,B)) to distinguish between different types of defects and improve its performance.

In addition, to capture the curvature change of the surface, this study proposes using normal vectors, as another input feature for training. The use of normal vectors for spall localization has

been suggested in other works, such as Kim et al. (2015b), which developed a methodology based on the changes in the depth and the normal vectors of defects. However, these works did not use ML. Let  $F(x, y, z) = 0$  represents the surface, which is calculated based on local surface fitting method. For a point  $P_o = (x_o, y_o, z_o)$ , the normal vector ( $\vec{N}$ ) is computed using Equation 3-1 (Silverman, 2002).

$$\vec{N} = \frac{\nabla F}{\|\nabla F\|} \quad \text{Equation 3-1}$$

where  $\nabla F$  is the gradient of  $F$  and  $\|\nabla F\|$  is the vector length. Ideally, as shown in Figure 3-1, the normal vectors of the no-defect area and the deepest point of a defect are perpendicular to the  $XZ$  plane, and the deviation between two adjacent normal vectors identifies the potential presence of a defect.

Furthermore, in the original PointNet++ (Qi et al., 2017b), the normalized coordinate values of  $(X'_i, Y'_i, Z'_i)$  are also used, which provide the network with the relative location of each point with respect to all other points in the segment. These parameters are computed based on Equations 3-2, 3-3, and 3-4 (Qi et al., 2017a).

$$X'_i = x_i / x_{max} \quad \text{Equation 3-2}$$

$$Y'_i = y_i / y_{max} \quad \text{Equation 3-3}$$

$$Z'_i = z_i / z_{max} \quad \text{Equation 3-4}$$

where  $x_{max}$ ,  $y_{max}$ , and  $z_{max}$  are the maximum values of  $x_i$ ,  $y_i$ , and  $z_i$  in each segment, respectively. However, since the changes in depth indicate the possibility of defect existence, only  $Y'_i$  is used in this research. Removing  $X'_i$  and  $Z'_i$  helps the model to understand the contribution of depth in detecting surface defects. As shown in Figure 3-1, a reference plane matching the damaged surface is used to calculate the depth of the points of the defect ( $y_i$ ) with respect to that surface. It can be seen that the value of  $Y'_i$  increases with the depth of the defect. Moreover, ideally the normal vectors of the deepest point of a defect ( $\vec{N}_1$ ) and the non-defect points ( $\vec{N}_2$ ) are perpendicular to the reference plane.

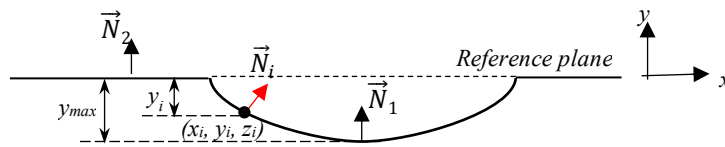


Figure 3-1. Cross-section of a sample defect with the normal vector and depth of a point.

Figure 3-2 shows a sample of blocks of points in a segment. Unlike the original PointNet++, in this research, 2D convolving for wrapping the dataset is performed on the  $XZ$  surface (vs. the  $XY$  surface) using the blocks with the given size of  $A \times A$ .

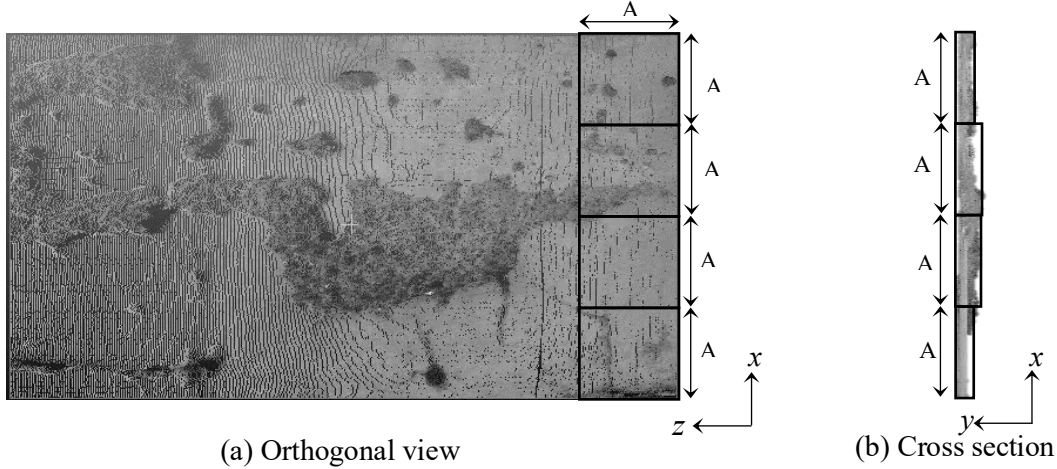


Figure 3-2. Sample of blocks of points in a segment.

(c) Addressing the issue of imbalanced data for priority classes

The imbalanced class distribution is one of the challenges in this research. Because not only the number of defects in each class (i.e., spalls and cracks) is different, but their sizes are also very different. Moreover, the largest part of the dataset belongs to the no-defect class, which has the least priority. Therefore, a weighted softmax cross-entropy loss function is applied to increase the contribution of the minority classes (spalls and cracks), which have priority. The weight ( $w_i$ ) and the cost weight of class  $i$  ( $W_i$ ) are calculated using Equations 3-5 and 3-6 (Cui et al., 2019).

$$w_i = N_i / \left( \sum_{i=1}^K N_i \right) \quad \text{Equation 3-5}$$

$$W_i = 1 / \log(1.05 + w_i) \quad \text{Equation 3-6}$$

where  $N_i$  and  $K$  represent the number of points in class  $i$  and the total number of classes, respectively.

(d) Applying sensitivity analysis to adjust the hyperparameters in order to better capture the features of small defects.

Sensitivity analysis is applied for observing how the changes of each hyperparameter impact the network's performance. The hyperparameters related to the network architecture and the dataset should be adjusted in case of using different datasets and features. In this research, the sizes of defects are much smaller than the indoor building elements (e.g., chairs, tables), which were the focus of the original PointNet++ (Qi et al., 2017b). The following sensitivity analysis is applied to adjust the hyperparameters in order to better capture the features of small defects. The range of each hyperparameter is selected to cover the most promising values.

(1) Hyperparameters related to the network architecture:

- **Number of sub-layers:** Sub-layers in PointNet++ are responsible for extracting the features from the sampled and grouped points using the mini-PointNet networks. The learning capacity of a neural network increases exponentially with its depth (i.e., number of sub-layers) and polynomially with its width (i.e., number of nodes) (Montufar et al.,

2014). As such, a sensitivity analysis on the number of sub-layers is performed to obtain the optimal network topology. In addition, two sets of features are used in the input layer (with and without the normal vectors) to study the impact on performance.

- **Sampling size:** Having multiple sampling sizes is one of the advantages of PointNet++, it is expected to reach better results using a variety of sampling sizes for different layers. Although all defects are relatively small, their sizes vary in a wide range. Therefore, the optimal sampling sizes should be able to accommodate large defects such as severe spalls as well as small cracks.

(2) Hyperparameters related to the dataset:

- **Size of the blocks and number of points in each block:** The density of the dataset depends on the number of points in each block and the block size. If the density of a block is less or more than the predefined value, it will be up-sampled or down-sampled, respectively. In order to minimize the changes due to down-sampling and up-sampling, the number of points in each block and the block size must be adjusted considering the distribution of segments based on their densities.

The original PointNet++ (Qi et al., 2017b) used 8,192 points in each  $1.5 m \times 1.5 m$  block for making a uniform dataset. Block sizes should be set based on the size of the segments and objects in each class (e.g., cracks, spalls). Therefore, it is expected to get better results using smaller blocks for defect semantic segmentation. Moreover, the number of points per block is selected based on the predefined block size so that the density of points fits within a reasonable density range. The density of points is calculated according to Equation 3-7.

$$Density = \frac{Number\ of\ points\ in\ the\ area}{Area} \quad \text{Equation 3-7}$$

Figure 3-3 represents the process of defining the appropriate values for block size ( $BL$ ) and number of points per block ( $NP$ ). The initial values for the number of points per block ( $NP_0$ ) and block size ( $BL_0$ ) are defined as follows:

$$NP_0 = NP_{min} = \text{original value} \quad \text{Equation 3-8}$$

$$BL_0 = BL_{max} \leq \min \{S_{min}, D_{max}\} \quad \text{Equation 3-9}$$

where  $NP_{min}$  is 8,192 points per block, and  $S_{min}$  and  $D_{max}$  are the sizes of the smallest segment and the biggest defect, respectively. In the first step, starting with  $NP_0$  and  $BL_0$ , the density is calculated. If it is in the acceptable density range, which is defined based on the density of most of the segments to limit the changes in the dataset due to down-sampling and up-sampling,  $NP_0$  and  $BL_0$  are considered as an acceptable combination ( $COM$ ). Then, the other acceptable  $NPs$  for  $BL_0$  are determined by increasing  $NP$  value by a certain value of  $\alpha * NP_0$  and the  $NP$  list is updated. If the density is not in the range,  $BL$  is decreased by  $\beta * BL_0$  and the acceptable  $NPs$  are selected out of the created  $NP$  list for new  $BL$ . For example, as will be explained in Section 3.3.3, in Table 3-4, first a list of acceptable  $NPs$ , including 22  $NPs$ , is created for  $BL_0$ , and then for  $BL_1$  the acceptable  $NPs$  are selected out of the available  $NP$  list. This process continues until the calculated density using the considered  $BL$  and  $NP_0$  is out of the predefined density range (O.R.).

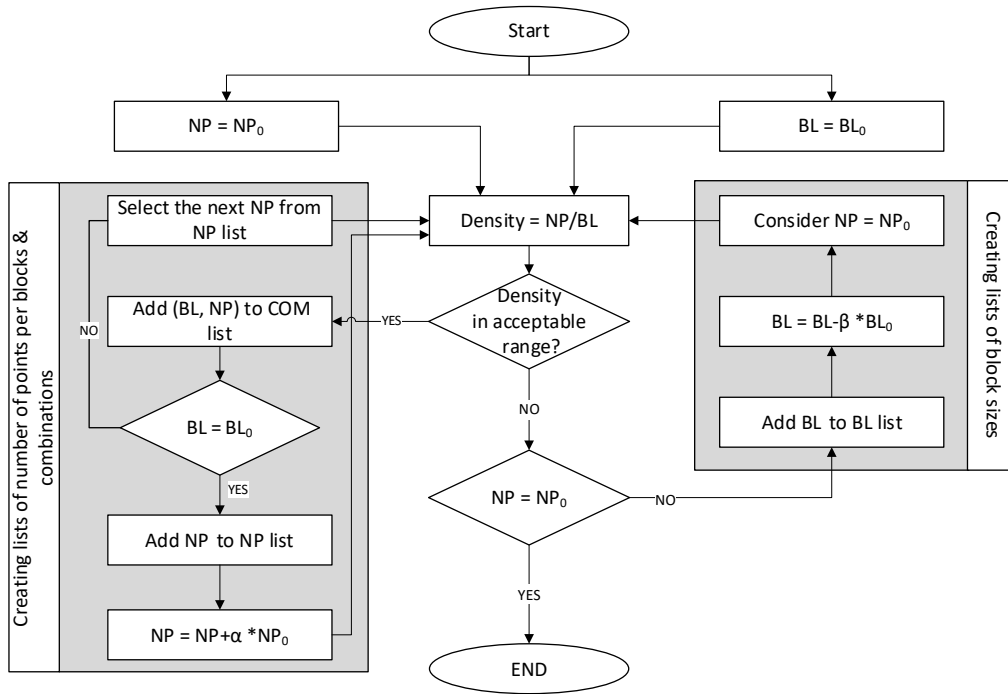


Figure 3-3. Defining the values for sensitivity analysis for number of points per block and block size.

- **Stride size:** Adding a stride in the data wrapping process may improve the results; however, it will increase the computation time. Therefore, sensitivity analysis helps to find the effect of stride size on the results and select the most suitable value. Figure 3-4 shows a sample of adding 25% stride.

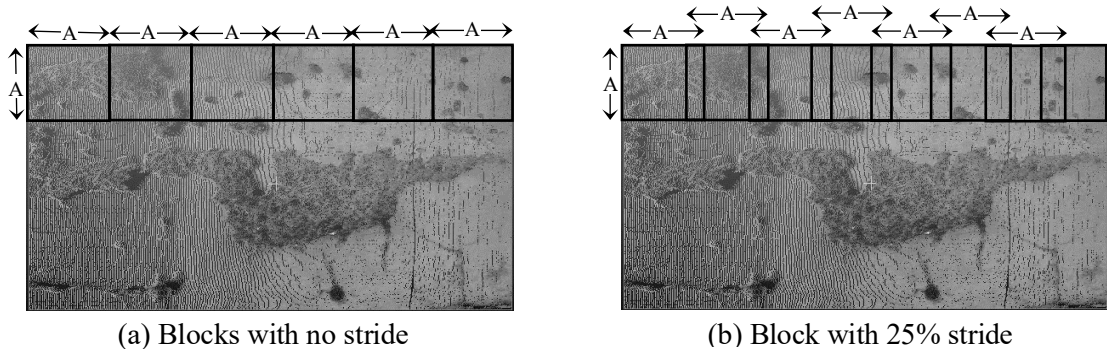


Figure 3-4. Sample of stride.

An efficient sensitivity analysis should be done in an organized multi-step process. As shown in Figure 3-5, the sensitivity analysis is applied in four steps. Step 1 aims to find the best architecture by varying the number of sub-layers and the sampling sizes, where the block size is set to its minimum value and the number of points per block is set to its maximum value because these values are expected to give the best results. For each number of layers, different combinations of sampling sizes are tried, and the performance metrics are calculated. Then, if it is the first case, a sub-layer is added to investigate the performance of a deeper network. Otherwise, the results are

compared with the previous models to decide if more layers are required or not and to select the best model.

After finding the best model in Step 1, the sensitivity analysis is applied to capture the best block size out of the values, which are specified based on Figure 3-3. In Step 2, the number of points per block and the stride size are fixed (i.e., maximum number of points and no stride). Based on the calculated performance metrics, the best value of the block size is selected. Having the best model and block size, Step 3 focuses on obtaining the best value for the number of points per block in case of no stride. Finally, Step 4 aims to investigate the effect of strides in different cases (i.e., 0%, 25%, and 50% strides).

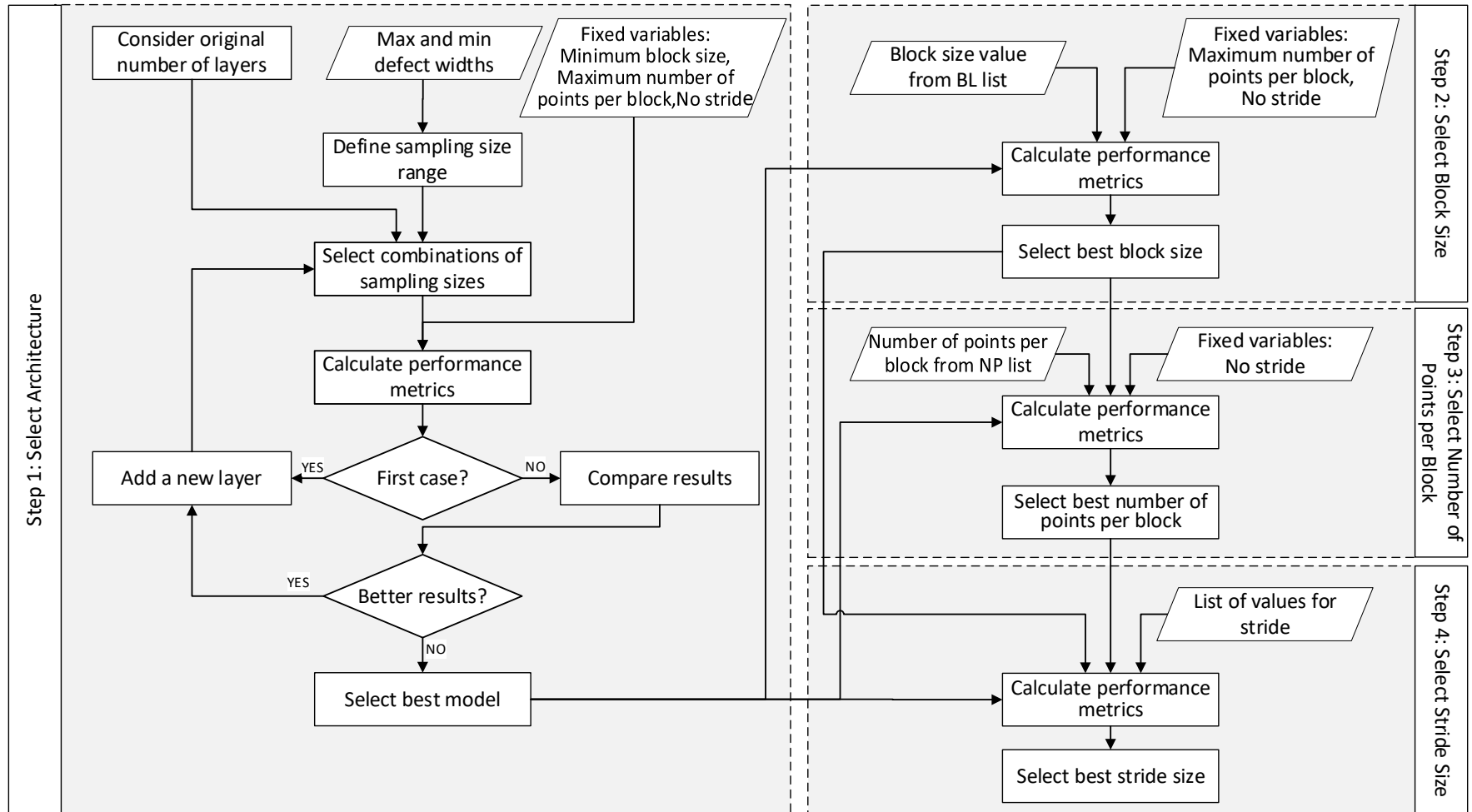


Figure 3-5. Sensitivity analysis framework.



### 3.2.2 Performance Metrics

Sensitivity analysis is applied and the best combination of hyperparameters is selected based on five performance metrics, which are computed using Equations 3-10 to 3-14.

$$\text{Precision} = TP / (TP + FP) \quad \text{Equation 3-10}$$

$$\text{Recall} = TP / (TP + FN) \quad \text{Equation 3-11}$$

$$F1 \text{ score} = 2 (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall}) \quad \text{Equation 3-12}$$

$$IoU = TP / (FP + TP + FN) \quad \text{Equation 3-13}$$

$$OA = \left( \sum_{i=1}^k TP_i \right) / (\text{Number of points}) \quad \text{Equation 3-14}$$

where  $i$  and  $k$  are the class number and the total number of classes, respectively. Figure 3-6 shows an example of calculating True Positive (TP), False Negative (FN), True Negative (TN), and False Positive (FP) for a class (e.g., crack).

Higher TP leads to detecting more points correctly, which leads to higher precision and recall. If the focus is mainly on detecting most defects, recall has the priority to evaluate the network performance. For example, a network with higher cracks precision is able to predict more cracks with a smaller number of other defects (i.e., spalls and no defects) miss-classified as cracks, while higher crack recall means missing less cracks and having less cracks miss-classified. On the other hand, IoU is a commonly used metric to measure the similarity between the ground-truth and predicted region (Rahman and Wang, 2016). Higher similarity requires minimizing FP and FN and maximizing TP. To ensure the efficiency of the proposed network, all these factors were considered in network performance evaluation. Most DL-based segmentation methods use simple loss functions (i.e., softmax) to optimize the OA (Rahman and Wang, 2016).

		True/Actual		
		Crack	Spall	No defect
Predicted	Crack	4	6	3
	Spall	1	2	0
	No defect	1	2	6

Annotations in the table:  
 - TP: Arrow points to the value 4 in the Crack/Crack cell.  
 - FN: Arrow points to the value 1 in the Spall/Crack cell.  
 - FP: Arrow points to the values 6 and 3 in the Crack/Spall and Crack/No defect cells.  
 - TN: Arrow points to the values 2 and 0 in the Spall/Spall and Spall/No defect cells.

Figure 3-6. Example of TP, TN, FP, and FN for the crack class.

A comparison between the accuracy of the model during training and evaluation in each epoch gives insight into the learning behavior, such as overfitting and underfitting (Brownlee, 2016). There are other factors, including learning rate, batch size, and number of epochs, which should be set properly for training the model.

### 3.2.3 Framework of Adjustments Made to PointNet++

This research is done in two stages. The first stage, adapted PointNet++, can be considered as a preliminary step and it focuses on answering the following questions: (1) Is flipping an appropriate approach for data augmentation? (2) Is focusing on depth of the points by removing  $X'$  and  $Z'$  and convolving on  $XZ$  surface beneficial? and (3) What are the best values for some hyperparameters such as stride sizes and number of points per block? To this end, the dataset is doubled by flipping horizontally, and the network is fed by a seven-dimensional array including  $x$ ,  $y$ ,  $z$ ,  $R$ ,  $G$ ,  $B$ , and  $Y'$ . Moreover, this method is useful in the case of limited computation resources (i.e., limited Random-Access Memory (RAM)).

After evaluating the performance of the adapted PointNet++ and ensuring its feasibility, all factors mentioned in Section 3.2.1 are applied in SNEPointNet++. Since flipping the dataset has not led to overfitting, the dataset is flipped twice (i.e., horizontally and vertically) in SNEPointNet++ to increase the size of the dataset. After preprocessing the dataset, the network is trained based on the points represented by a 10-dimensional vector with values of  $x$ ,  $y$ ,  $z$ ,  $R$ ,  $G$ ,  $B$ ,  $N_x$ ,  $N_y$ ,  $N_z$ , and  $Y'$ . Moreover, sensitivity analysis covers all the hyperparameters shown in Figure 3-5.

Figure 3-7 represents the overall framework of the proposed defect semantic segmentation, where the dashed boxes are the steps which are different between the two stages.

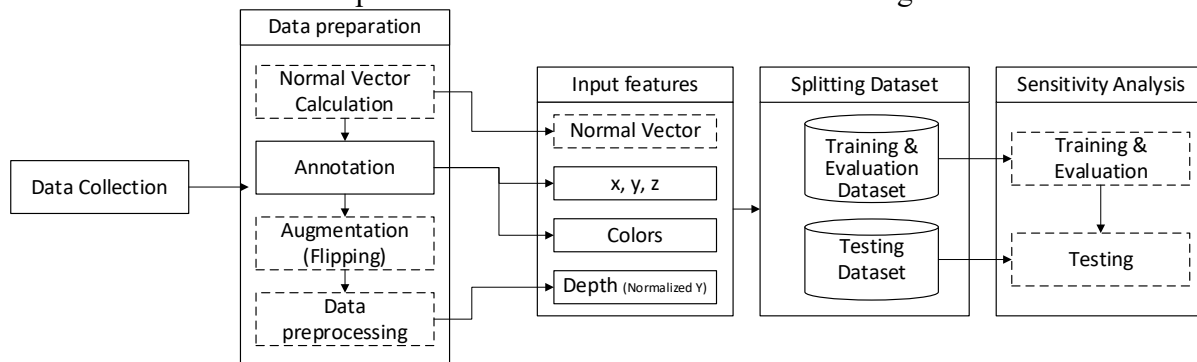


Figure 3-7. Overall framework of the proposed defect semantic segmentation

### 3.3 Implementation and Case Study

Figure 3-8 shows the implementation and the case study process of defect semantic segmentation in detail. As shown in this figure, the prior steps, which are mostly related to removing irrelevant points and registration of the collected point clouds, are common between adapted PointNet++ and SNEPointNet++. Although the process of some steps (i.e., annotation, data preprocessing) are the same for both networks, the inputs and outputs are different.

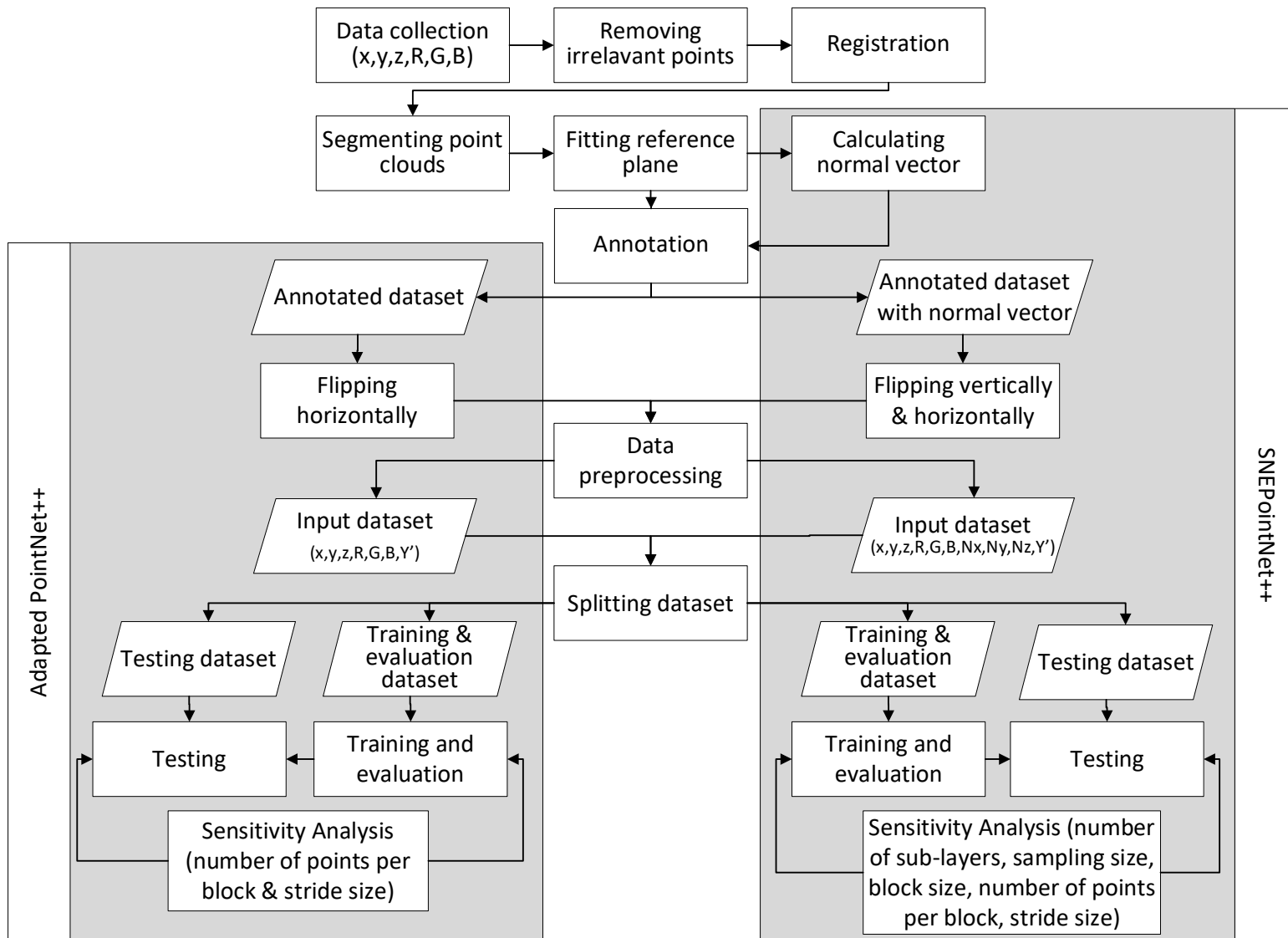


Figure 3-8. Implementation and case study steps of adapted PointNet++ and SNEPointNet++.

### 3.3.1 Data Collection

The inputs of the proposed method, which are the 3D point cloud datasets, were scanned from four reinforced concrete bridges in Montreal using a FARO Focus3D scanner (FARO Technologies Inc., 2012). The 3D Faro LiDAR is equipped by a camera, which automatically captures images during scanning and detects the color of each point. The specifications of the scanner are tabulated in Table 3-1.

Table 3-1. FARO Focus3D LiDAR specifications (FARO Technologies Inc., 2011).

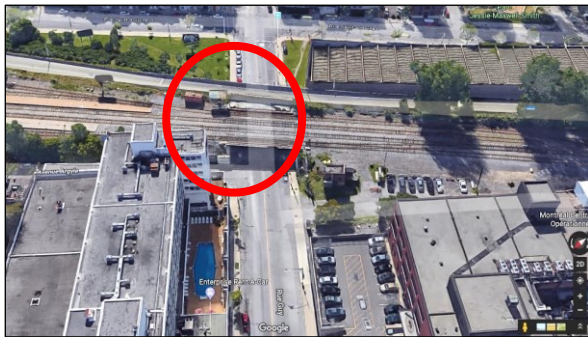
LiDAR	Points per Second	Field of View		Angular Resolution	Accuracy	Measurement Range
		Vertical	Horizontal			
FARO Focus 3D	976,000	305°	360°	0.009°	±2 mm	1.5 m – 120 m

The locations of the scanned bridges are shown in Figure 3-9. Some studies focused on formalizing the scan planning to minimize the number of scanning stations (viewpoints) with full coverage of inspected surfaces considering scanning parameters and the required level of details (Aryan et al., 2021; Huang et al., 2021). Since scan planning is out of the scope of this research, planning the scanning stations and acquisition parameter settings for each bridge are not optimized. To guarantee that the acquired datasets fully cover the scanning targets including several parts of the bridge substructures (i.e., abutments), they are scanned from several stations. Figure 3-10 shows samples of some scanning positions on the western side of Bridge 1. Moreover, increasing the number of stations may improve the datasets' quality and increase the level of details due to more overlapping datasets. Besides, the scanner location impacts the incidence angle. A larger incidence angle gives a coarser dataset. Therefore, scanning the same surface with different incidence angles may help to generalize the trained model for different intensities. To this end, scanning Bridge 1 based on two different scan settings provides two datasets with different incidence angles and different qualities and colors.

The acquisition parameters, such as FoVs, resolution, quality, and the number of scanned points, are presented in Table 3-2. Quality, which varies between 1x and 6x, is related to the length of time to capture points. For example, 1x and 4x mean 1  $\mu$ s and 8  $\mu$ s per scan point, respectively. Higher slider leads to longer scanning time and less errors. (FARO Technologies Inc., 2017). The resolution is a factor to set the number of points acquired per rotation. It determines the density of the scan point and can be between 1/32 and 1. For instance, Faro can scan 710 Million and 0.7 Million points over a full area scan with its highest (1/1) and lowest (1/32) resolutions, respectively (FARO Technologies Inc., 2011). In other words, the resolution factor directly affects the distances between scanned points. For example, the minimum distances between two points using (1/1) and (1/2) resolutions are 0.3 mm and 0.62 mm, respectively, at 2 m scanning distance. Therefore, the minimum detectable widths of the defects, which are located at the perpendicular view of the scanner, are 0.6 mm and 1.24 mm, respectively. In this case study, the minimum defect width (2 mm) is higher than these values. Although using the highest resolution with maximum quality takes longer scanning duration, it is required to minimize the errors. However, this is not always a feasible option due to the battery limitations (i.e., its capacity and performance in harsh weather), as well as traffic restrictions. To prevent scanning irrelevant objects, the horizontal and vertical

FoVs can be decreased. Depending on the scan parameters, the required time and the approximate number of points can be calculated by the scanner in advance.

CloudCompare (Girardeau-Montaut, 2020) is a 3D point cloud processing software. It has recently gained more popularity than some other software tools (e.g., Trimble (Trimble Inc., 2020) and ReCap (Autodesk Inc., 2021) due to its free accessibility and higher speed in simple tasks such as registration and removing irrelevant data. However, due to an inconsistency between the output format of the Faro scanner (\*.fls) and the input format acceptable by the CloudCompare software (\*.ply), a format conversion was required, which was performed using the ReCap software.



(a) Bridge 1: Guy Street



(b) Bridge 2: Lucian L'Allier Street



(c) Bridge 3: Avenue Atwater



(d) Bridge 4: Chemin Macdonald

*Figure 3-9. Data collection locations in Montreal.*

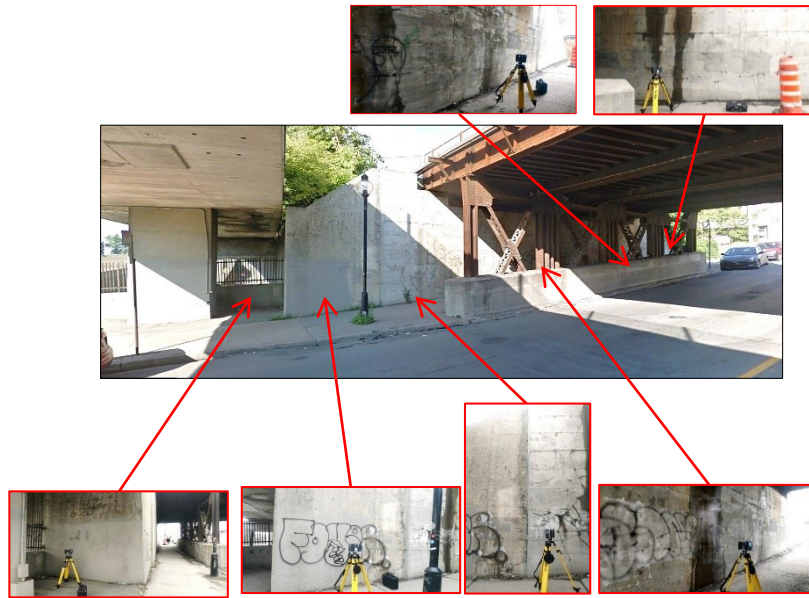


Figure 3-10. Samples of scanning positions on the western side of Bridge 1.

Table 3-2. Scanning information.

	Scan 1	Scan 2	Scan 3	Scan 4	Scan 5
<i>Bridge Number</i>	1	1	2	3	4
<i>Number of Stations</i>	8	4	6	4	2
<i>Horizontal FoV</i>	23° to 259°	23° to 259°	0° to 360°	0° to 360°	0° to 360°
<i>Vertical FoV</i>	-43° to 71°	-43° to 71°	-63° to 90°	-45° to 71°	-60° to 90°
<i>Resolution</i>	1/4	1/4	1/1	1/2	1/2
<i>Quality</i>	6x	6x	2x	4x	4x
<i>Number of Points (Mpts)</i>	25.5	25.5	710.7	134.5	177.7

### 3.3.2 Data Preparation

The first step in the data preparation is registering the scanned point clouds and removing the irrelevant points (i.e., buildings, trees). The following sections explain the other steps of preparing a raw point cloud dataset for training.

### 3.3.2.1 Segmenting Point Clouds

Figure 3-11(b) shows three parts, which were segmented out of the scanned bridge surface after registration and removing irrelevant points. Although the sizes of the segments were not fixed, three simple rules were set based on experience: (1) Since the blocks have a box shape, the scanned surfaces are divided into rectangular segments; and (2) Each segment should be big enough to contain different sizes of defects and small enough to avoid covering many undamaged areas. Therefore, the minimum size of the segments was considered around  $0.6\text{ m} \times 0.6\text{ m}$  and the average size was about  $2\text{ m} \times 2\text{ m}$ . A high-quality dataset affects the accuracy of the annotation, while also segments with different densities are required to have an invariant network to the density of the dataset. This dataset is available at <https://github.com/neshatbln/SNEPointNet2>.

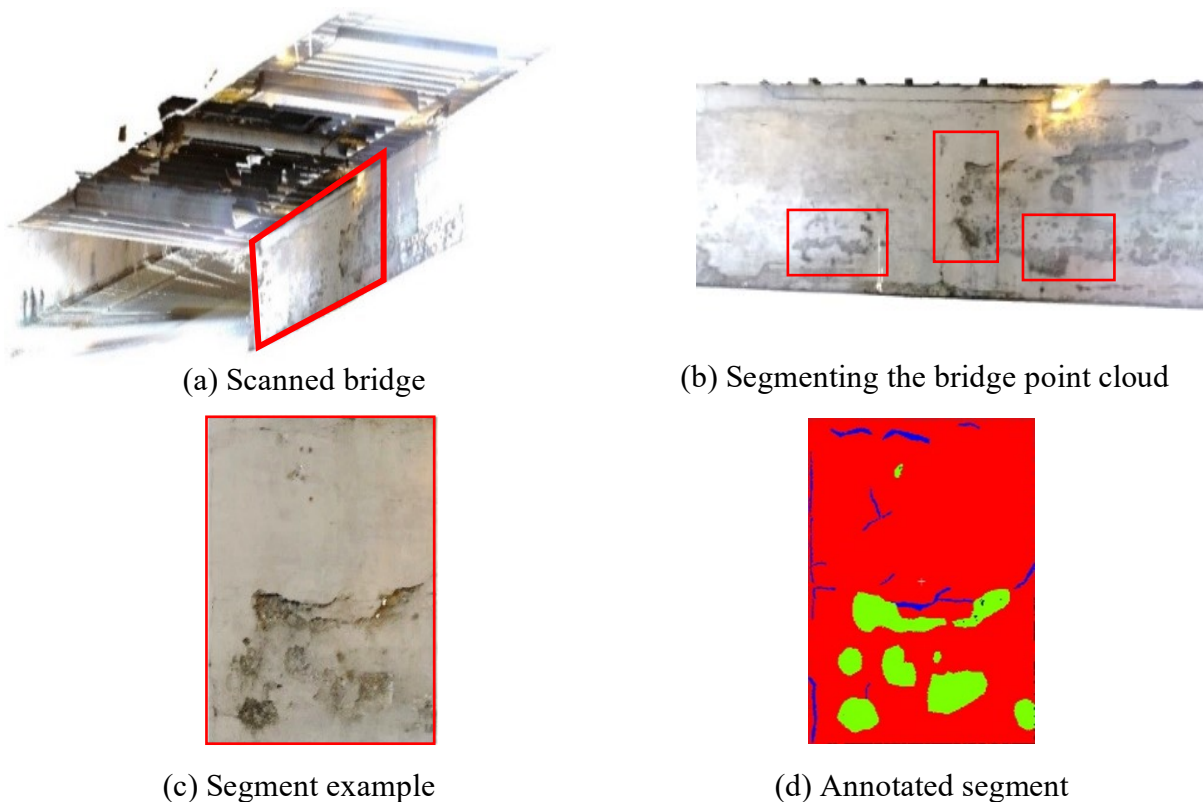


Figure 3-11. Example of segmentation and annotation processes.

### 3.3.2.2 Fitting Reference Plane

Using CloudCompare, the  $Z$ -axis of the canonical coordinate system was set in the vertical direction. As shown in Figure 3-1, the  $X$ -axis is along the concrete bridge surface, and the  $Y$ -axis is perpendicular to the concrete surface (outside direction). As a result of this setting, the points representing defects will have negative  $Y$  values.



### 3.3.2.3 Calculating Normal Vectors

Normal vectors, which were only considered in SNEPointNet++, were calculated in CloudCompare software and three attributes ( $N_x$ ,  $N_y$ , and  $N_z$ ) were added to each point. Figure 3-12 shows a fitted segment on the  $XZ$  plane with some samples of points and their normal vectors.

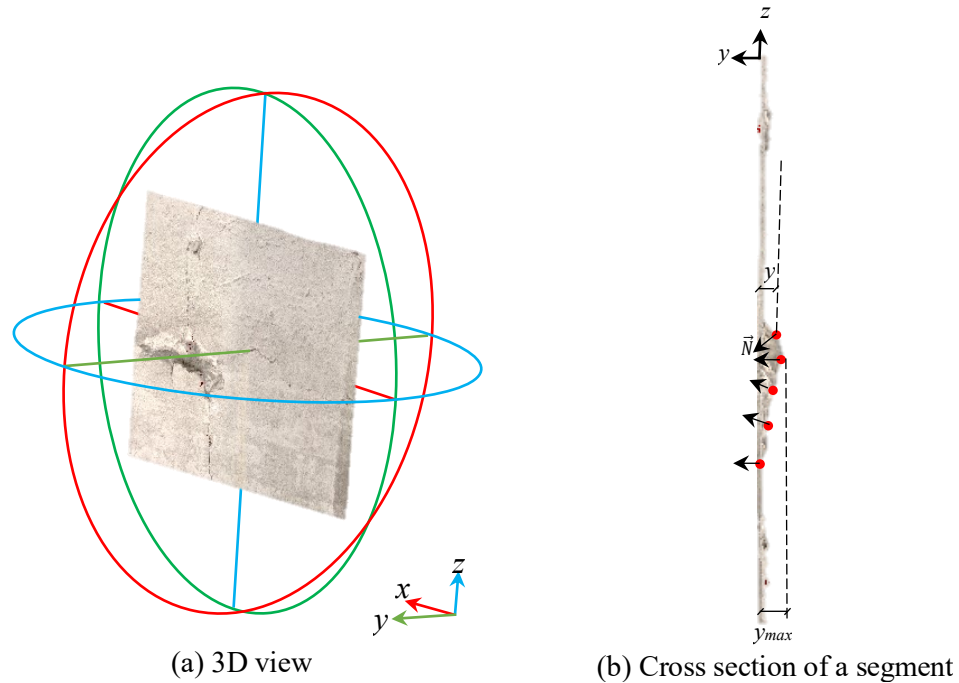


Figure 3-12. Sample of the fitted segment.

### 3.3.2.4 Annotation

Since this study aims to detect two types of concrete surface defects, cracks and spalls, the scanned point clouds were manually annotated and labeled into three classes: cracks, spalls, and no-defect. An example of an annotated segment is shown in Figure 3-11(d), where cracks, spalls, and non-defect areas are shown in blue, green, and red, respectively. The statistical information of the combined dataset is given in Table 3-3. The prepared dataset contains 102 segments (around 27 MPts) including 595 cracks (246,699 points) and 773 spalls (1,935,165 points). Based on the number of points of cracks and spalls, these classes are the minorities with a huge difference in the size of the data compared with the majority class, which is no defect. Figure 3-13 shows the distribution of the dataset based on segment density calculated using Equation 3-7.



Table 3-3. Statistics of the annotated dataset before augmentation.

Dataset	Number of segments	Number of points	Cracks		Spalls		No defect
			Number of cracks	Number of points	Number of spalls	Number of points	Number of points
Training & Evaluation	81	21,313,285	475	182,430	588	1,252,551	19,878,304
Testing	21	5,628,620	120	64,269	185	682,614	4,881,737
Total	102	26,941,905	595	246,699	773	1,935,165	24,760,041

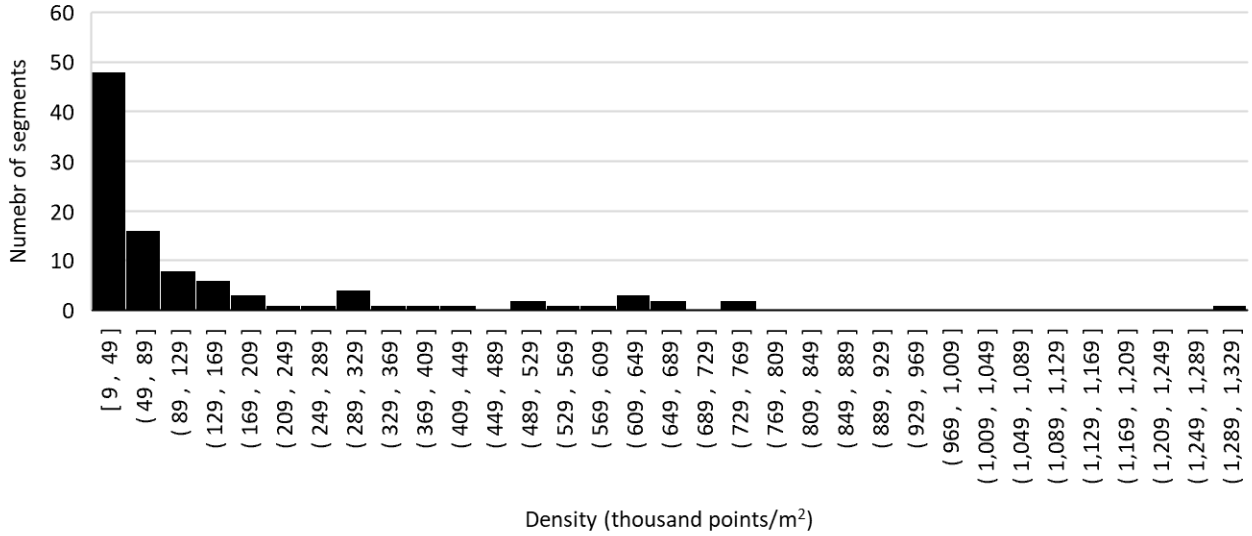


Figure 3-13. Distribution of dataset segments based on density.

### 3.3.2.5 Data Augmentation

For adapted PointNet++, the dataset, which includes location and color features, was flipped horizontally using the Python code in Appendix A.1. For SNEPointNet++, a tripled size dataset, including location, colors, and normal vector features, was generated by flipping the original dataset twice, horizontally and vertically, using the Python code in Appendix A.2.

### 3.3.2.6 Data Preprocessing

The output of the data augmentation is a point cloud dataset including several annotated segments of bridge surface in TXT file format. In this step, first, the point cloud dataset attributes and the labels were concatenated and stored into *NumPy* format files.

The original PointNet++ semantic segmentation model is designed and evaluated based on S3DIS dataset in the Hierarchical Data Format (HDF) as \*.h5 files. HDF is an abstract data management and storing model (The HDF Group, 2021). Therefore, the normalized value of  $y$  ( $Y'$ ) was calculated and added to each point and the annotated point clouds were wrapped inside the blocks and saved in HDF5 format. The number of points in each block, block sizes, and stride sizes are the variables, which can be adjusted in this step. The dataset density is not homogeneous and depends on the distance between the scanner and the surface, incidence angle, and step sizes. To

have a homogeneous dataset, the number of points in all batches was unified by down-sampling or up-sampling.

### 3.3.3 Training and Testing

The method was implemented using TensorFlow-GPU 1.15.1, Cuda 11.0, and python 3.6. The number of batches and initial learning rate were assumed to be 24 and 1e-3, respectively. The learning rate decayed 50% every eight epochs until the minimum value of 1e-5 was reached. Based on the original PointNet++ (Qi et al., 2017b), ReLU and softmax were used as activation functions and ADAM was considered as the optimization method. Sensitivity analysis was applied to find the best combination of hyperparameters and adapt the network.

Table 3-4 shows the lists of values for number of points per block, block sizes, and the density of their combinations, which were defined based on Figure 3-3. Based on Equation 3-9, the initial block size ( $BL_0$ ) was considered 40 cm, which was smaller than the smallest dimension of segments (46 cm) and the biggest defect size (60 cm) in this research. On the other hand, the density in each case should be between 9,000 pts/m<sup>2</sup> and 329,000 pts/m<sup>2</sup>, which is the density range for 85% of segments in the dataset. As shown in Table 3-4, the block size was decreased when the density was out of range (O.R.).

Table 3-4. List of number of points per block, block sizes, and their densities.

			Block Size (cm <sup>2</sup> )			
			<i>BL</i> <sub>0</sub>	<i>BL</i> <sub>1</sub>	<i>BL</i> <sub>2</sub>	<i>BL</i> <sub>3</sub>
			40 x 40	30 x 30	20 x 20	10 x 10
Number of points per block	<i>NP</i> <sub>0</sub>	8,192	51,200	91,022	204,800	819,200 (O.R.)
	<i>NP</i> <sub>1</sub>	10,240	64,000	113,778	256,000	-
	<i>NP</i> <sub>2</sub>	12,288	76,800	136,533	307,200	-
	<i>NP</i> <sub>3</sub>	14,336	89,600*	159,289*	358,400 (O.R.)	-
	<i>NP</i> <sub>4</sub>	16,384	102,400*	182,044*	-	-
	<i>NP</i> <sub>5</sub>	18,432	115,200*	204,800*	-	-
	<i>NP</i> <sub>6</sub>	20,480	128,000*	227,556*	-	-
	<i>NP</i> <sub>7</sub>	22,528	140,800*	250,311*	-	-
	<i>NP</i> <sub>8</sub>	24,576	153,600*	273,067*	-	-
	<i>NP</i> <sub>9</sub>	26,624	166,400*	295,822*	-	-
	<i>NP</i> <sub>10</sub>	28,672	179,200*	318,578*	-	-
	<i>NP</i> <sub>11</sub>	30,720	192,000*	341,333 (O.R.)	-	-
	<i>NP</i> <sub>12</sub>	32,768	204,800*	-	-	-
	<i>NP</i> <sub>13</sub>	34,816	217,600*	-	-	-
	<i>NP</i> <sub>14</sub>	36,864	230,400*	-	-	-
	<i>NP</i> <sub>15</sub>	38,912	243,200*	-	-	-
	<i>NP</i> <sub>16</sub>	40,960	256,000*	-	-	-
	<i>NP</i> <sub>17</sub>	43,008	268,800*	-	-	-
	<i>NP</i> <sub>18</sub>	45,056	281,600*	-	-	-
	<i>NP</i> <sub>19</sub>	47,104	294,400*	-	-	-
	<i>NP</i> <sub>20</sub>	49,152	307,200*	-	-	-
	<i>NP</i> <sub>21</sub>	51,200	320,000*	-	-	-

\* Cannot be applied due to computation resource limitations (i.e., RAM)

\*\* All densities are in pts/m<sup>2</sup>

### 3.3.3.1 Adapted PointNet++

Training and testing were performed on a cloud computing platform, called Compute Canada (2021), using 2 GPUs, 24 GB RAM per GPU, and 32-core CPU. Figure 3-14 shows the adapted PointNet++ fed by 7-dimensional vectors. The hierarchical point set feature learning starts with sampling  $N_1$  points out of all  $n$  points of the dataset in the first sub-layer and considering each point as the centroid of a region. Then, the points in the radius of 5 cm from the centroids are grouped and fed locally into a mini PointNet, which is comprised of three MLPs with 32, 32, and 64 nodes followed by a max pooling operation to extract new features. The learning process continues considering 10 cm, 20 cm, and 30 cm radiuses for grouping the points around  $N_2$ ,  $N_3$ , and  $N_4$  number of centroid points, respectively, in three more sub-layers. Afterwards, in the next phase, to interpolate the learning features the inverse process is applied using four mini PointNet units, which are fed by two sets of features: (1) the features from the previous MLP, which are interpolated to fit to the current MLP; and (2) the features from the corresponding mini PointNet in the previous learning set (sampling and grouping). Finally, two Fully-Connected (FC) layers are added to reach the score ( $m$ ) for  $n$  points using the weighted Softmax function.

In the preliminary sensitivity analysis, the hyperparameters were limited to the number of points per block and stride size. As shown in Table 3-5, nine cases (Case A1-A9) were defined based on three numbers of points (i.e., 8,192 points, 10,240 points, and 12,288 points) and three stride sizes (i.e., 0%, 25%, and 50%) for the largest block size (40 cm × 40 cm). The results of training, evaluation, and testing are shown in Table 3-5 and Table 3-6.

As shown in Figure 3-15, changes in the number of points and stride sizes have effects on computation time. Although the highest recalls for both cracks and spalls belong to Case A9, training this model takes the longest time (20 hours and 35 minutes) among all. Training Case A3 takes 14 hours less than Case A9 and leads to only 2% and 0.2% decreases in crack and spall recalls, respectively. Considering 10,240 points per block with 50% stride (Case A8) results in the best performance in all terms except spall recall. In comparison with Case A8, although increasing the number of points in Case A9 improves spall recall by 2.9% with no effect on crack recall, it decreases the time efficiency and performance of the network in terms of precision, F1-score, and IoU. To this end, if there is a time limitation and the only focus is on not missing defects, Case A3 would be a reliable choice for training a model. Otherwise, the overall performance of Case A8 is the best.

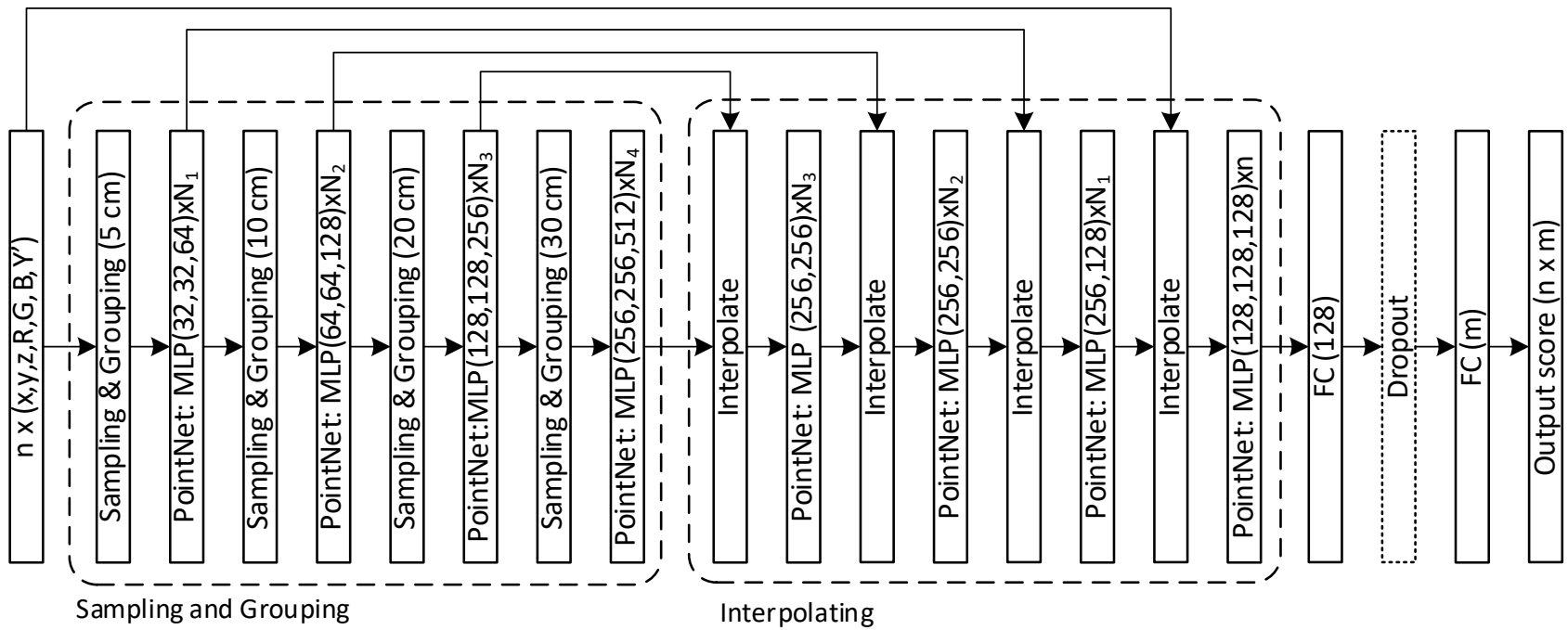


Figure 3-14. Architecture of adapted PointNet++ of the performance.

Table 3-5. Training and evaluation results of adapted PointNet++ for various stride sizes and numbers of points per block.

Case	Number of points per block	Stride size (cm × cm)	Training		Evaluation	
			Mean loss	OA (%)	Mean loss	OA (%)
A1	8,192	40×40 (0%)	0.120	98.4%	0.135	96.2%
A2	10,240		0.087	98.7%	0.102	96.8%
A3	12,288		0.094	98.7%	0.107	96.7%
A4	8,192	40×30 (25%)	0.089	98.7%	0.134	96.3%
A5	10,240		0.079	98.8%	0.114	96.7%
A6	12,288		0.074	98.9%	0.115	96.7%
A7	8,192	40×20 (50%)	0.108	98.5%	0.127	96.5%
A8	10,240		0.071	<b>99.0%</b>	0.100	<b>97.1%</b>
A9	12,288		0.086	98.7%	0.094	97.0%

Table 3-6. Testing results of adapted PointNet++ for various stride sizes and numbers of points per block.

Case	Cracks				Spalls				No defects			
	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)
A1	27.6	21.9	24.4	13.9	66.9	69.6	68.3	51.8	95.3	95	95.2	90.8
A2	38.4	45.2	41.6	26.2	67.6	77.5	72.2	56.5	96.5	94.4	95.5	91.3
A3	39.7	48.1	43.5	27.8	68.0	80.4	73.7	58.4	96.9	94.4	95.6	91.6
A4	29.5	26.5	27.9	16.2	75.8	75.7	75.7	60.9	96	96.2	96.1	92.5
A5	42.9	43.5	43.2	27.5	74.9	79.6	77.2	62.8	96.6	95.8	96.2	92.7
A6	41.7	49.3	45.2	29.2	76.6	75.6	76.1	61.4	96.2	96.1	96.1	92.6
A7	22.2	24.4	23.3	13.2	70.2	79.4	74.5	59.4	96.3	94.3	95.3	91
A8	<b>46.6</b>	<b>50.1</b>	<b>48.3</b>	<b>31.8</b>	<b>80.0</b>	77.7	<b>78.9</b>	<b>65.1</b>	96.3	<b>96.6</b>	<b>96.5</b>	<b>93.2</b>
A9	42.4	<b>50.1</b>	45.9	29.8	75.2	<b>80.6</b>	77.8	63.6	<b>96.9</b>	95.7	96.3	92.8

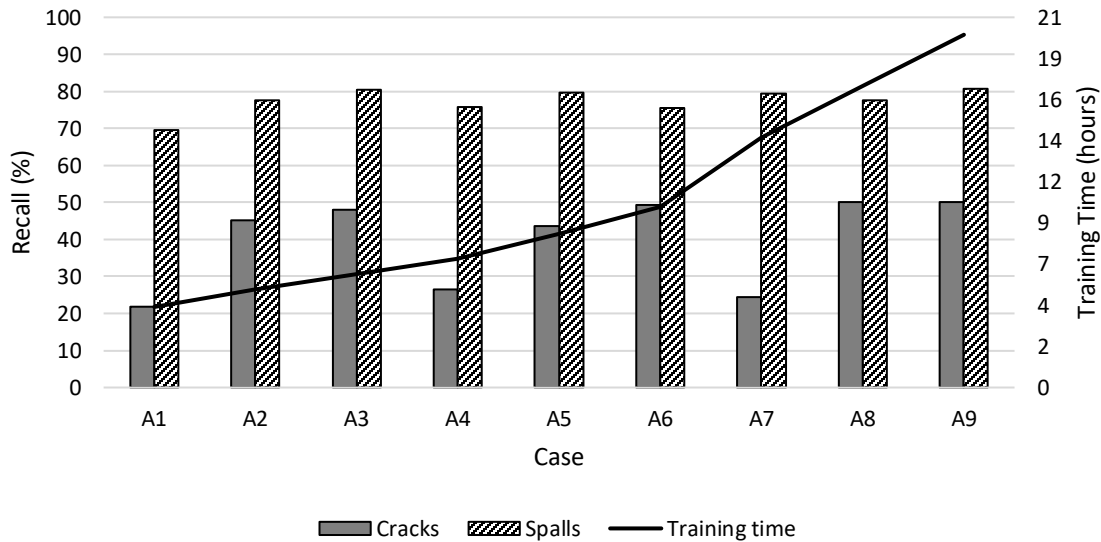
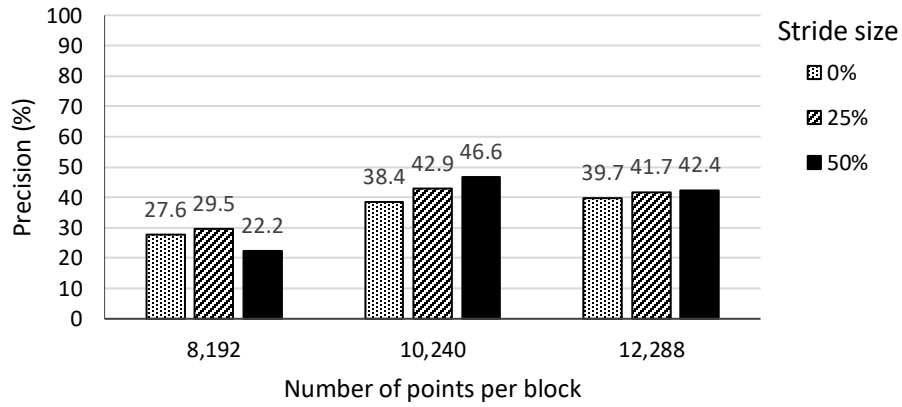


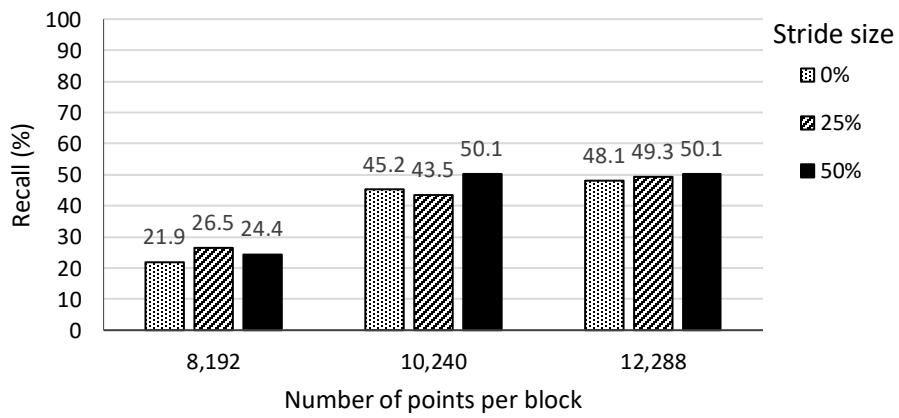
Figure 3-15. Testing results and training time of cases A1-A9.

Figure 3-16 represents the effect of stride and the number of points in each block on the adapted PointNet++ crack semantic segmentation. Based on Figure 3-16, adding the stride improves the network performance except in Case A5, which causes a decrease of 1.7% in crack recall. On the other hand, increasing the 25% stride to 50% for the blocks with 8,192 points decreases the efficiency of the model. In contrast, this change in strides causes improvement in the case of 10,240 points per block, while it has only minor impacts on training a model with 12,288 points. Therefore, the effect of strides on the overall performance of the model becomes less pronounced by increasing the number of points. This result can be interpreted that by feeding the model with more points, the model reaches its maximum learning capacity, and the provided information by strides may be redundant or lead to overfitting.

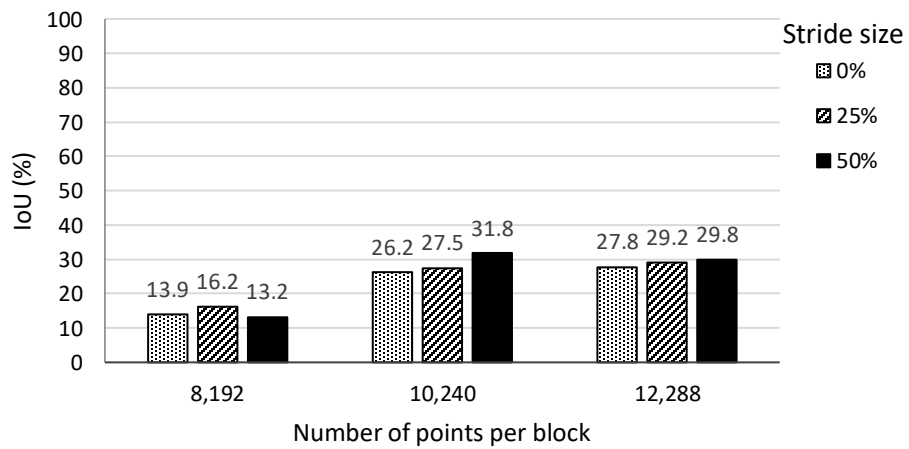
Based on the outcomes of spall semantic segmentation using adapted PointNet++ in Figure 3-17, adding 25% stride improves the network performance in terms of IoU, recall, and precision, and only in the case of having 12,288 points the spall recall decreases by 4.8%. There is no clear pattern with respect to increasing the stride to 50%. The effect of changing the number of points is more prominent in the case of increasing 8,192 points to 10,240 points. For instance, the highest performance improvement is a 10% increase in precision with 50% stride, and the number of points increases from 8,192 points to 10,240 points.



(a) Precision



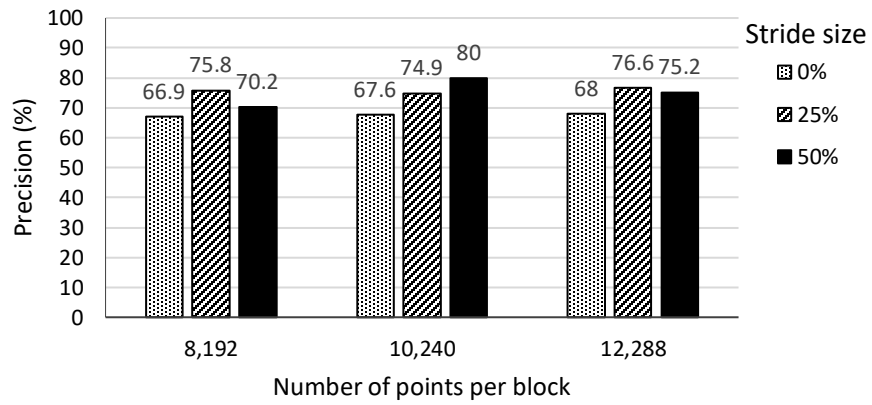
(b) Recall



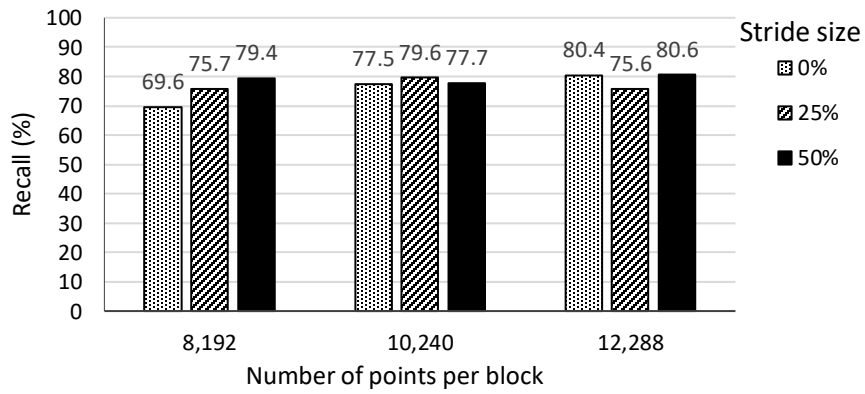
(c) IoU

Figure 3-16. Effect of stride and number of points on crack semantic segmentation using adapted PointNet++.

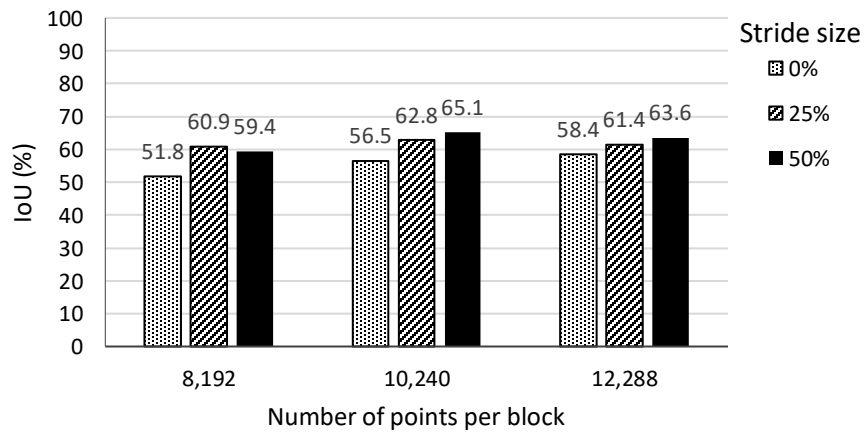




(a) Precision



(b) Recall



(c) IoU

Figure 3-17. Effect of stride and number of points on spall semantic segmentation using adapted PointNet++.

### 3.3.3.2 SNEPointNet++

Training and testing of SNEPointNet++ were performed on a LAMBDA workstation, with 3 NVIDIA RTX A6000 GPUs, 48 GB RAM per GPU, and AMD Ryzar Threadripper 3960x 48-core CPU. Most of the algorithms were developed in Python 3.8 and the environment was created in a Docker container. Based on Figure 3-5, the sensitivity analysis includes four main steps. In Step 1, various sampling sizes between 2.5 cm and 40 cm, which is the maximum value smaller than the smallest segment, for three number of sub-layers were considered to find the best architecture. In Step 2 and Step 3, the block sizes and number of points per block were selected out of the predefined values in Table 3-4. The values of hyperparameters in each step of the sensitivity analysis of the case study are shown in Table 3-7, and the detailed process is explained in the following.

Table 3-7. Hyperparameter values in each round of sensitivity analysis.

<i>Hyperparameters</i>	<i>Step 1 (Series M)</i>	<i>Step 2 (Series B)</i>	<i>Step 3 (Series N)</i>	<i>Step 4 (Series S)</i>
<i>Sub-layers &amp; sampling size</i>	Sublayers: 4,5,6 Sampling sizes: between 2.5 cm- 40 cm	Best of Series M	Best of Series M	Best of Series M
<i>Block size (cm × cm)</i>	20 × 20	20 × 20 30 × 30 40 × 40	Best of Series B	40 × 40
<i>Number of points per block</i>	12,288	12,288	8,192 10,240 12,288	8,192 10,240 12,288
<i>Stride size</i>	0%	0%	0%	0% 25% 50%

#### (a) Effect of number of sub-layers and sampling size

In Step 1, four, five, and six sublayers were considered to find the effective depth of SNEPointNet++. The widths of defects vary between 0.2 cm (i.e., hairline cracks) and 50 cm (i.e., severe spalls). Moreover, the smallest size of segments is 46 cm. Therefore, a relatively wide range of 2.5 cm to 40 cm was considered for the sampling size of each sub-layer. The training and evaluation results and the training time for the five best networks are shown in Table 3-8, while their testing results are illustrated in Table 3-9. Comparing the results by the number of layers shows that increasing the number of layers from four (M1 and M2) to five (M3 and M4) leads to higher efficiency. Compared to the best 5-layer network (M4), adding the sixth layer does not improve the learning process due to overfitting.

Considering 5-layer networks, the combination of smaller sampling sizes (M4) results in around 2% to 3% improvement over M3 in crack precision and IoU, while there are 0.8% and 0.6% reductions in recall and IoU of spalls, respectively. Considering the network performance in terms of both crack and spall semantic segmentation, M4 is selected as the most efficient architecture, which is shown in Figure 3-18.

Table 3-8. Training and evaluation results of the top five combinations of sublayers and sampling sizes.

Case	Number of layers	Sampling size in each layer (cm)						Training		Evaluation		Training time (Hrs:mins)
		1	2	3	4	5	6	Mean loss	OA (%)	Mean loss	OA (%)	
M1	4	5	10	20	30	-	-	0.072422	98.1	0.073447	97.7	27:46
M2		2.5	10	20	30	-	-	0.080615	97.8	0.080318	97.5	28:03
M3	5	5	10	20	30	40	-	0.064733	98.3	0.064739	98.1	29:52
M4		2.5	5	10	20	30	-	0.058815	98.3	0.057952	98.1	30:35
M5	6	2.5	5	10	20	30	40	0.055237	98.3	0.056810	98.1	32:43

Table 3-9. Testing results of the top five combinations of sublayers and sampling sizes.

Case	Cracks			Spalls			No defects		
	Pre. (%)	Recall (%)	IoU (%)	Pre. (%)	Recall (%)	IoU (%)	Pre. (%)	Recall (%)	IoU (%)
M1	67.4	90.6	63.9	86.3	90.1	78.8	99.2	98.5	97.7
M2	68.9	90.7	64.3	85.1	89.8	77.7	99.2	98.4	97.6
M3	71.2	91.3	66.6	90.4	91.2	<b>83.1</b>	99.4	98.5	98.0
M4	<b>73.3</b>	<b>93.0</b>	<b>69.2</b>	89.9	<b>92.0</b>	82.5	99.3	98.8	98.1
M5	70.9	93.6	67.6	<b>90.6</b>	90.5	82.8	99.2	98.8	98.1

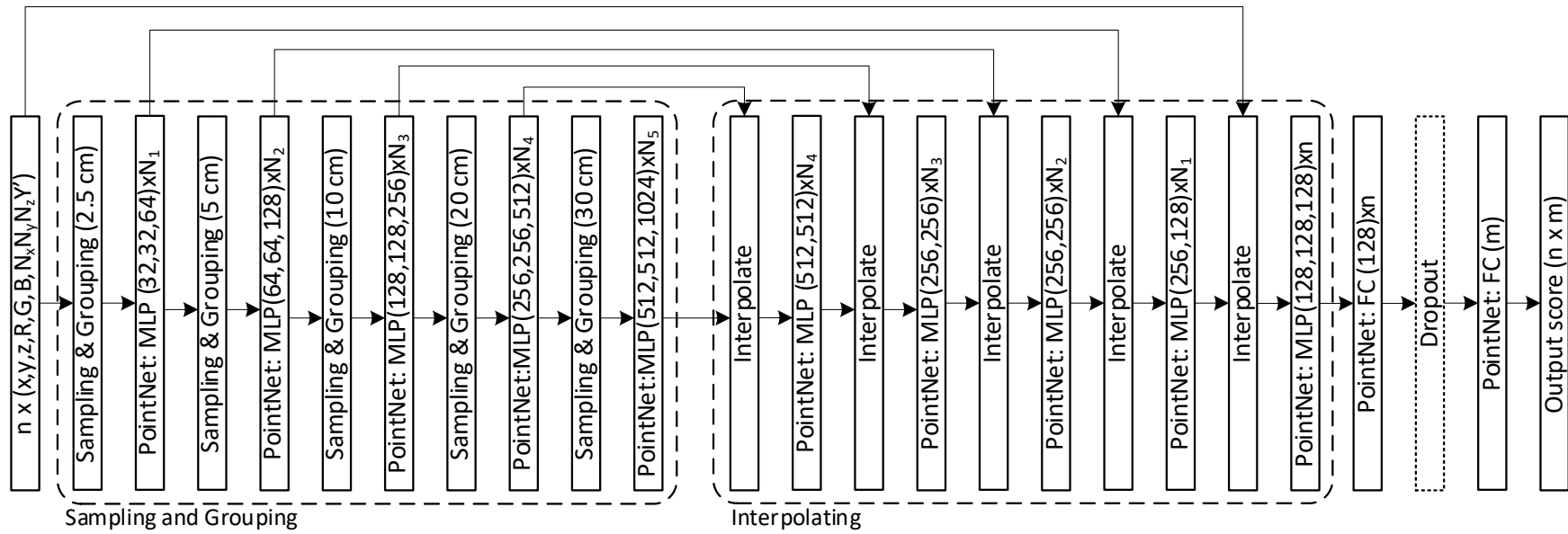


Figure 3-18. Architecture of SNEPointNet++ with the best performance (Case M4).

(b) Effect of block size and number of points per block

In Step 2 and Step 3, two series of cases (B and N) were defined to adjust block size and number of points per block, respectively. The list of the predefined values is shown in Table 3-4. As shown in Table 3-7, to find the best combination of block size and number of points per block, first block size and then number of points per block were set. In set B, considering three block sizes of 20 cm × 20 cm, 30 cm × 30 cm, and 40 cm × 40 cm with 12,288 points per block leads to uniform densities of 307,200 pts/m<sup>2</sup>, 136,533 pts/m<sup>2</sup>, and 76,800 pts/m<sup>2</sup>, respectively. As shown in Table 3-10, decreasing the block size from 40 cm × 40 cm to 30 cm × 30 cm results in better performance for both cracks and spalls. Although decreasing the block size from 30 cm × 30 cm (Case M4-B30) to 20 cm × 20 cm (Case M4-B20) improves the network performance for crack semantic segmentation, the spall recall and IoU decrease by 2.5% and 6.6%, respectively. It can be concluded that very small block sizes cannot be well-trained for larger defects (i.e., spalls). The most efficient block size was 20 cm × 20 cm due to its considerable effect on crack detection.

Table 3-11 represents the testing results of three cases defined based on different numbers of points (8,192 points, 10,240 points, and 12,288 points) per 20 cm × 20 cm block. It can be seen that more points per block result in better performance.

Table 3-10. Testing results for different block sizes.

Case	Block size (cm × cm)	Density (Pts/m <sup>2</sup> )	Cracks			Spalls			No defects		
			Pre. (%)	Recall (%)	IoU (%)	Pre. (%)	Recall (%)	IoU (%)	Pre. (%)	Recall (%)	IoU (%)
M4-B20	20 × 20	307,200	<b>73.3</b>	<b>93.0</b>	<b>69.2</b>	89.9	92.0	82.5	99.3	<b>98.8</b>	<b>98.1</b>
M4-B30	30 × 30	136,533	61.7	81.5	54.2	<b>94.0</b>	<b>94.5</b>	<b>89.1</b>	<b>99.5</b>	98.4	97.9
M4-B40	40 × 40	76,800	44.4	62.3	34.9	84.5	86.4	74.6	99.0	98.6	97.6

Table 3-11. Testing results for different numbers of points per block.

Case	Number of points per block	Density (Pts/m <sup>2</sup> )	Cracks			Spalls			No defects		
			Pre. (%)	Recall (%)	IoU (%)	Pre. (%)	Recall (%)	IoU (%)	Pre. (%)	Recall (%)	IoU (%)
M4-B20-N12	12,288	307,200	<b>73.3</b>	<b>93.0</b>	<b>69.2</b>	<b>89.9</b>	<b>92.0</b>	<b>82.5</b>	<b>99.3</b>	<b>98.8</b>	<b>98.1</b>
M4-B20-N10	10,240	256,000	62.2	85.8	57.9	83.0	87.7	76.2	99.3	98.5	97.5
M4-B20-N8	8,192	204,800	51.0	81.7	49.2	80.4	88.2	72.6	99.5	98.4	97.9

(c) Effect of stride size

In Step 4, the stride could not be applied on 20 cm × 20 cm due to the computation resource limitations (i.e., RAM size). Therefore, it was decided to consider the minimum possible block size (40 cm × 40 cm) and investigate the effect of stride size (0%, 25%, and 50%) on the results for three different number of points per block (8,192 points, 10,240 points, and 12,288 points). The testing results of the SNEPointNet++ for nine cases are tabulated in Table 3-12.

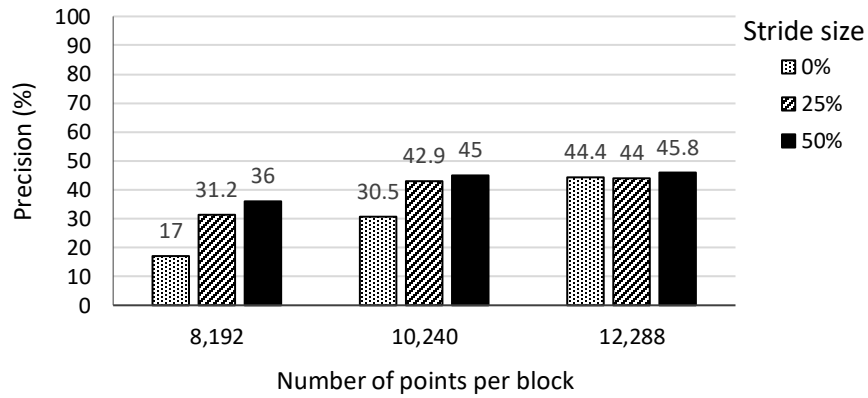
Table 3-12 and Figure 3-19 show the effect of increasing the stride size for different numbers of points in the case of crack semantic segmentation. It can be seen that by adding the strides (Cases M4-S4, M4-S5, and M4-S6), the performance mostly improves in terms of precision and IoU. These improvements are more dominant for the smaller number of points because there is more missing information, which can be provided using strides. Unlike IoU and precision, adding stride does not have any noticeable effect on recall for small number of points (Case M4-S4), while it improves the performance for larger number of points (Cases M4-S5 and M4-S6). The main reason for this trend can be the negative impact of increasing the number of points in the blocks with no stride (Cases M4-S1, M4-S2, and M4-S3). To evaluate the effect of increasing the stride size on model performance, Cases M4-S4, M4-S5, and M4-S6 are compared with Cases M4-S7, M4-S8, and M4-S9, respectively. The results demonstrate gradual improvements in crack IoUs and precisions and minor fluctuations in recalls.

Figure 3-20 shows the testing results of spall semantic segmentation for Cases M4-S1 to M4-S9. The greatest improvement due to adding strides belongs to Case S6 with 12,288 points, where recall and IoU increased 5.9% and 5.7%, respectively. Although increasing the stride size from 25% to 50% results in a 6% increase in spall precision of Case M4-S8, it does not lead to a significant improvement in the performance, and even in two cases, the spall recalls decrease. In general, compared to cracks, spall semantic segmentation performance is less sensitive to the stride size.

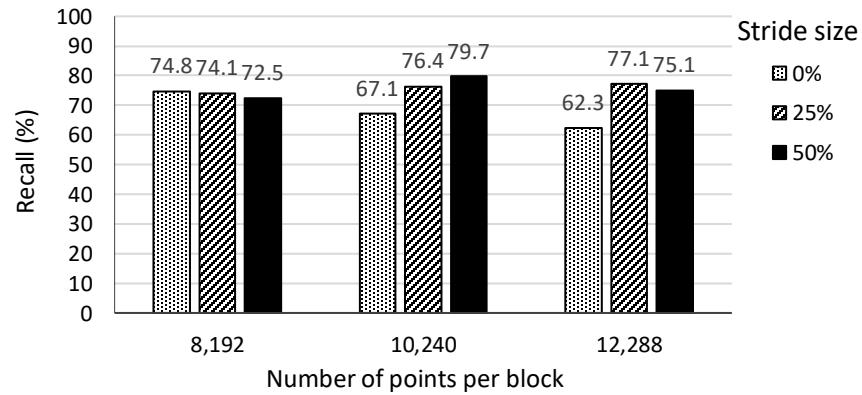
Comparing the results of Case A8 in Table 3-6 with Case M4-S8 in Table 3-12 shows that the modifications in the SNEPointNet++ (without decreasing the number of block sizes) lead to 29.6% and 13.7% increases in cracks and spalls recalls, respectively, while they result in 8.6% and 16.8% performance improvement for crack and spall semantic segmentation, respectively, in terms of IoU.

Table 3-12. Testing results for various stride sizes and numbers of points per block.

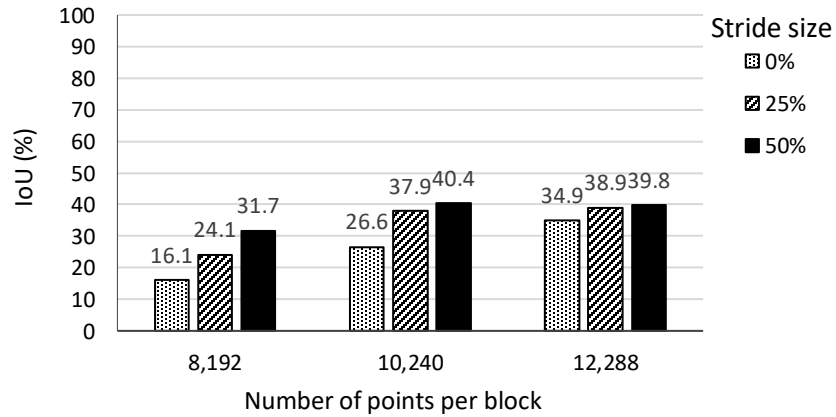
Case	Number of points per block	Stride size (cm × cm)	Cracks				Spalls				No defects			
			Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)
M4-S1	8,192	40×40 (0%)	17.0	74.8	27.7	16.1	85.7	90.3	87.9	78.5	99.5	98.6	99.0	98.1
M4-S2	10,240		30.5	67.1	41.9	26.6	82.5	90.0	86.1	75.6	99.4	98.4	98.9	97.8
M4-S3	12,288		44.4	62.3	51.8	34.9	84.5	86.4	85.4	74.6	99.0	98.6	98.8	97.6
M4-S4	8,192	40×30 (25%)	31.2	74.1	43.9	24.1	85.3	92.9	88.9	80.1	99.5	98.2	98.9	97.8
M4-S5	10,240		42.9	76.4	55.0	37.9	82.8	92.4	87.4	77.6	99.4	98.2	98.8	97.7
M4-S6	12,288		44.0	77.1	56.0	38.9	86.1	92.3	89.1	80.3	99.4	98.6	99.0	98.0
M4-S7	8,192	40×20 (50%)	36.0	72.5	48.1	31.7	86.4	<b>94.1</b>	90.1	82.0	99.5	98.5	99.0	98.0
M4-S8	10,240		45.0	<b>79.7</b>	<b>57.6</b>	<b>40.4</b>	88.8	91.4	90.1	81.9	<b>99.4</b>	<b>98.9</b>	<b>99.1</b>	<b>98.3</b>
M4-S9	12,288		<b>45.8</b>	75.1	56.9	39.8	<b>89.1</b>	91.4	<b>90.3</b>	<b>82.3</b>	<b>99.4</b>	<b>98.9</b>	<b>99.1</b>	<b>98.3</b>



(a) Precision

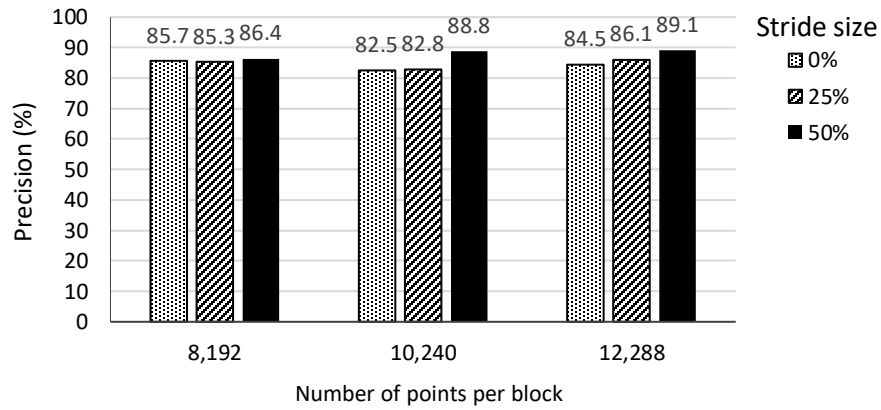


(b) Recall

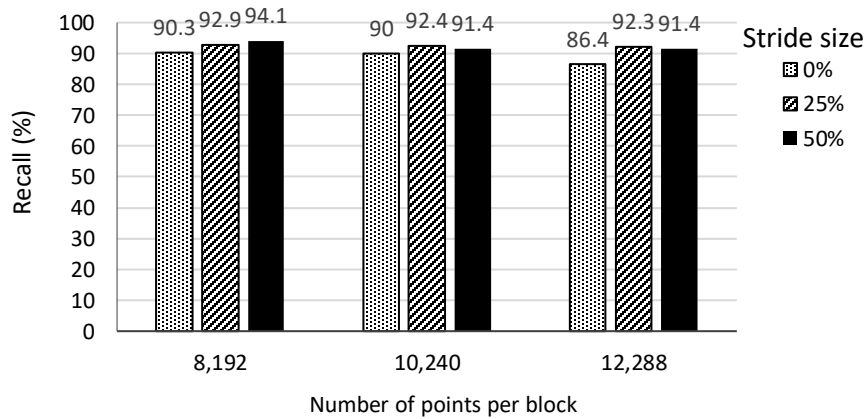


(c) IoU

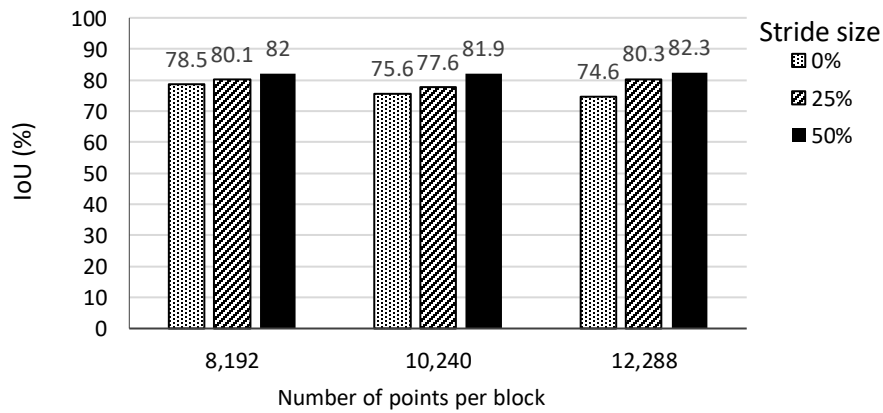
Figure 3-19. Effect of stride on crack semantic segmentation using SNEPointNet++.



(a) Precision



(b) Recall



(c) IoU

Figure 3-20. Effect of stride on spall semantic segmentation using SNEPointNet++.



## 3.4 Discussion

### 3.4.1 Training, Evaluations, and Testing Results

Table 3-13 summarizes the parameters and hyperparameters of the original PointNet++ and the two adjusted networks. The results of the training, evaluation, and testing of the best models of adapted PointNet++ and SNEPointNet++ are tabulated in Table 3-14 and Table 3-15. Using adapted PointNet++ was useful as an initial step to ensure the feasibility of applying PointNet++ in this research domain. Compared to adapted PointNet++, SNEPointNet++ performance in detecting the cracks is 43% and 37% better in terms of recall and IoU, respectively. Moreover, in SNEPointNet++, the recall and IoU of spalls are 14% and 17% higher than in the adapted PointNet++, respectively. Furthermore, the SNEPointNet++ improves the semantic segmentation performance of cracks more than spalls. The ground truth and the predicted results of three segments are visualized in Figure 3-21, where spalls, cracks, and no-defect classes are shown in green, blue, and red, respectively.

As mentioned before, the datasets play an important role in the learning process. The main differences between the two adjusted networks are data augmentation and input features. The adapted PointNet++ used only horizontal flipping of the collected dataset. In contrast, the dataset used in SNEPointNet++ is three times bigger than the raw dataset due to flipping horizontally and vertically. Comparing Sample 1 in Figure 3-21(b) with Figure 3-21(c), most of the unpredicted crack points in the adapted PointNet++ were considered as no defects. On the other hand, as shown in Figure 3-21(d), the normal vector can be a helpful factor in the learning process. Another reason for the improved results of SNEPointNet++ is the deeper network and adding the smaller sample sizes (e.g., 2.5 cm) without removing other sample sizes. Hence, cracks have more chances to be detected.

Figure 3-21(c) is a sample of network performance in differentiating between the defects and human-based color changes on the surface (i.e., inspectors' marking, graffiti). It can be seen that the network efficiently performed in detecting a line, which looks like a crack, as no defect. This is a good example of the advantage of point clouds over images.

Table 3-13. Comparison between PointNet++ and adjusted networks (best configurations).

Parameter	PointNet++	Adapted PointNet++	SNEPointNet++
Classes	Building indoor objects (11 classes)	Cracks, spalls, no-defect	Cracks, spalls, no-defect
Data augmentation	Random rotation	Flipping horizontally	Flipping horizontally and vertically
Convolving direction	XY surface	XZ surface	XZ surface
Size of blocks (m)	$1.5 \times 1.5 \times Z_{max}$	$0.4 \times Y_{max} \times 0.4$	$0.2 \times Y_{max} \times 0.2$
Stride	N.A.	50%	0%
Input Variables	$x, y, z, R, G, B, X', Y', Z'$	$x, y, z, R, G, B, Y'$	$x, y, z, R, G, B, N_x, N_y, N_z, Y'$
Number of points in each block	8,192 pts	10,240 pts	12,288 pts
Number of sub-layers	4	4	5
Sampling sizes (cm)	10, 20, 40, 80	5, 10, 20, 30	2.5, 5, 10, 20, 30
Number of epochs	200	50	50
Learning rate	1e-3 (decays exponentially to a minimum of 1e-5)		

Table 3-14. Training and evaluation results.

Method	Training		Evaluation	
	Mean loss	OA (%)	Mean loss	OA (%)
Adapted PointNet++	0.071	99.01	0.100	97.12
SNEPointNet++	0.059	98.3	0.058	98.1

Table 3-15. Testing results.

Method	Cracks				Spalls				No defects			
	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)	Pre. (%)	Recall (%)	F1 Score (%)	IoU (%)
Adapted PointNet++	46.6	50.1	48.3	31.8	80.0	77.7	78.9	65.1	96.3	96.6	96.5	93.2
SNEPointNet++	<b>73.3</b>	<b>93.0</b>	<b>82.0</b>	<b>69.2</b>	<b>89.9</b>	<b>92.0</b>	<b>90.9</b>	<b>82.5</b>	<b>99.3</b>	<b>98.8</b>	<b>99.0</b>	<b>98.1</b>

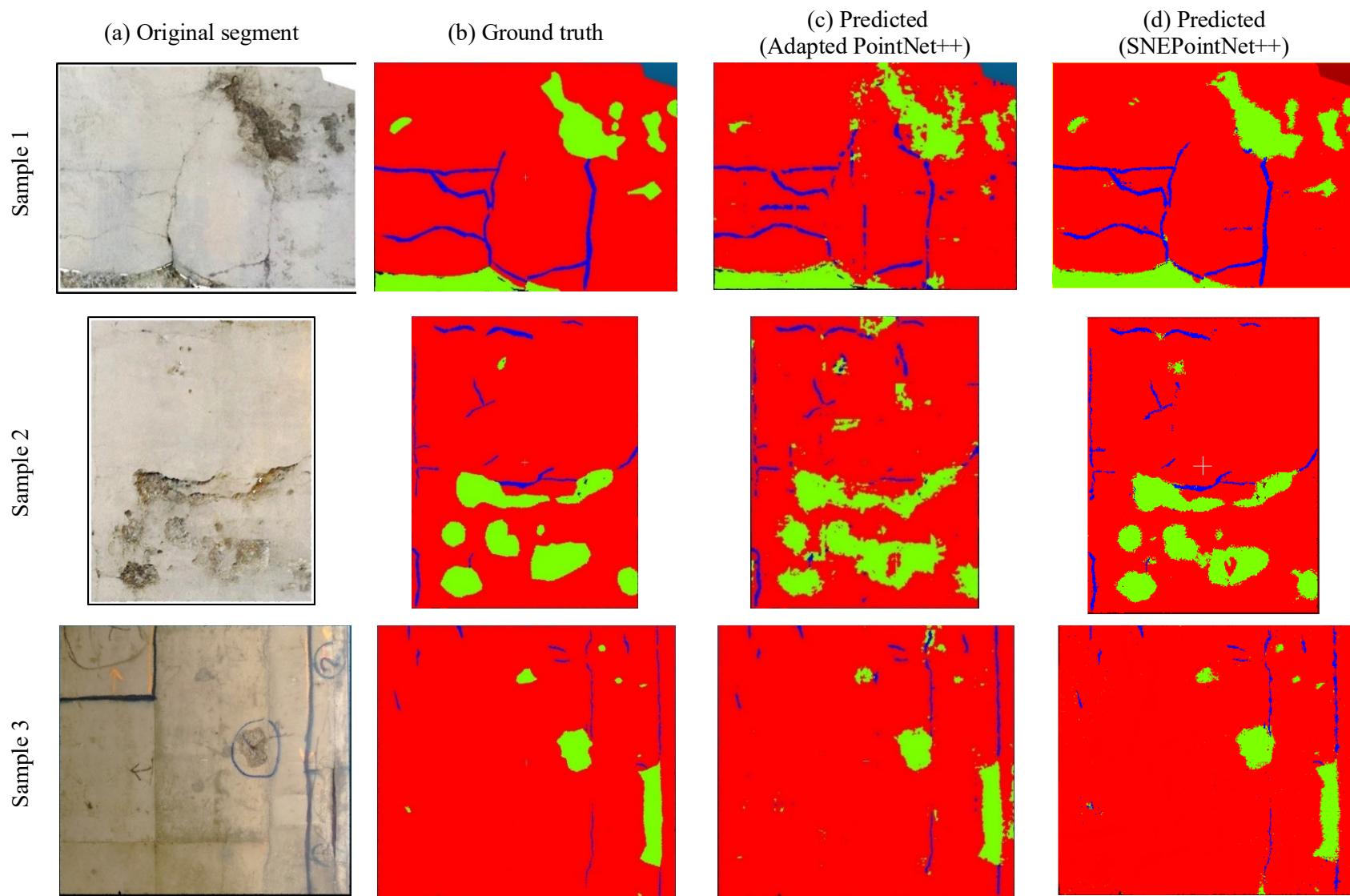


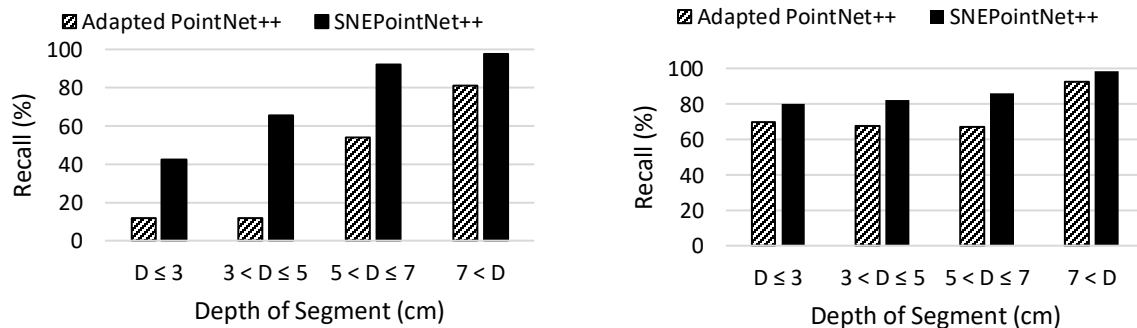
Figure 3-21. Three samples of semantic segmentation.

### 3.4.2 Testing Results Based on Segment Depth

Testing results of the adapted PointNet++ (Case A8) and SNEPointNet++ (Case M4-B20-N12) are classified based on the maximum depth of each sample (segment) in Table 3-16. As shown in Figure 3-22, despite the low recall for segments with less than 5 cm depth, SNEPointNet++ modifications significantly improve the performance of semantic segmentation of deeper defects, especially cracks. The results show that the deepest segments have the highest recalls for both cracks (81% recall and 64% F1-Score) and spalls (93% recall and 87% F1-Score) using adapted PointNet++. Using SNEPointNet++ improves the performance up to 98% and 99% recalls for semantic segmentation of cracks and spalls, respectively. Moreover, the cracks of deeper segments can be detected using SNEPointNet++ with 98% recall and 96% F1-score. These results validate the basic hypothesis of this research about the ability of point cloud-based DL methods to better detect defects by exploiting the depth information.

Table 3-16. Classifying the testing results based on the segment depth.

Depth (cm)	Defect type	Adapted PointNet++		SNEPointNet++		Number of segments ratio (%)
		Recall (%)	F1-score (%)	Recall (%)	F1-score (%)	
$D \leq 3$	Cracks	11.8	15.7	42.6	36.2	9.5
	Spalls	69.9	71.33	79.9	79.8	
$3 < D \leq 5$	Cracks	12.0	15.3	65.3	53.9	14.3
	Spalls	67.6	69.0	82.2	76.0	
$5 < D \leq 7$	Cracks	54.0	51.3	92.3	89.3	42.9
	Spalls	66.9	75.3	85.7	89.7	
$7 < D$	Cracks	81.1	63.9	97.5	96.3	33.3
	Spalls	92.6	87.0	98.6	97.7	



(a) Cracks (b) Spalls  
Figure 3-22. Classifying testing results based on depth of segments.

### 3.4.3 Comparison with Image-Based Methods

The recent image-based methods have reached around 98% (Le et al., 2021) to 99% (Vignesh et al., 2021) recalls in concrete surface defect classification. However, classification is not the appropriate approach to find the semantic information of each point individually, which is the objective of this research.

The similar image-based semantic segmentation results are tabulated and compared with SNEPointNet++ in Table 3-17. Compared to this research, the results of Lopez Droguett et al. (2020) were 25% and 4% higher in terms of IoU and recall, respectively, for crack semantic segmentation. A huge dataset, which was built using image processing from the captured videos of five bridge structures, is one of the main reasons of their high network performance. Although Lee et al. (2019) trained a model with the 14% higher precision in crack semantic segmentation, SNEPointNet++ leads to 19% and 2% higher recall and F1-score, respectively. Considering the recall as the most beneficial performance metric in bridge inspection, SNEPointNet++ is more practical than Lee et al. (2019). Although Fu et al. (2021) and Wang et al. (2022) used a bigger dataset than SNEPointNet++, their networks results are 18% and 20% lower, respectively, in terms of IoU crack semantic segmentation. Compared to SNEPointNet++, Hoskere et al. (2020) detected cracks and spalls with 1% and 2% less IoU, respectively. To this end, the performance of SNEPointNet++ is better than the image-based methods, except Lopez Droguett et al. (2020), which used a very large dataset.

Table 3-17. Comparison between SNEPointNet++ and image-based DL semantic segmentation methods.

Reference	Type of Defect	Sample size	Results (%)			
			IoU	Recall	Precision	F1-score
Lee et al. (2019)	Crack	242 cracks		74	87	80
Hoskere et al. (2020)	Crack	341 cracks	68			
	Spall	324 spalls	81			
Lopez Droguett et al. (2020)	Crack	256,115 cracks	94	97 (Av.)		97 (Av.)
Mohammed Abdelkader et al. (2021)	Spall	60 spalls				92
Fu et al. (2021)	Crack	5,000 cracks	65			
Wang et al. (2022)	Crack	2,446 images	63	68		
SNEPointNet++	Crack	595 cracks	69	93	73	82
	Spall	773 spalls	83	92	90	91

### 3.4.4 Comparison with Point Cloud-Based Methods

The results of the similar point cloud-based methods and SNEPointNet++ are summarized in Table 3-18. To compare point cloud-based surface defect detection methods, they are categorized into three groups based on the types of defects (i.e., cracks, spalls, and both). The results of crack detection methods are incomparable because the papers used different metrics (i.e., error) or visualization approaches to show the results. Although the dataset of Valença et al. (2017) included cracks with 0.1 mm to 4 mm width, the minimum detectable crack width was 1 mm. Turkan et al. (2018) proposed the only DL-based method in this category, which is not comparable due to showing the results based on errors.

In the case of spall detection, the best result of Kim et al. (2015b) was 92% average recall and 97% average precision for ten samples with the sizes between 10 mm × 10 mm × 4 mm and 100 mm × 100 mm × 7 mm. Due to scanning larger spalls in an ideal situation in terms of incidence angle and distance, better results are expected. Moreover, another experiment showed that their proposed method was not able to detect spalls with less than 4 mm depth.

The last group, which belongs to spall and crack detection methods, includes two studies. Unlike the method of Guldur and Hajjar (2017), the adapted DGCNN (Bahreini and Hammad, 2021) and the proposed networks of this research were evaluated using the same dataset. The efficiency of SNEPointNet++ is 10% and 24% higher than the adapted DGCNN in terms of IoU of spalls and cracks, respectively.

Table 3-18. Comparison between SNEPointNet++ and similar point cloud-based methods.

Reference	Method		Type of Defect	Sample size/ dimensions (cm)	Results (%)				
	DL	Non-DL			IoU	Recall	Pre.	F1	N.C.*
Truong-Hong et al. (2016)		✓	Crack	L: 1.8 – 0.8; W: 0.4– 1.2					✓
Valença et al. (2017)		✓	Crack	L: 3.0; W:0.1-0.4					✓
Turkan et al. (2018)	✓		Crack	L: 1.0 – 3.5; W: 0.9 – 1.6; D: 0.1-0.5					✓
Kim et al. (2015b)		✓	Spall	Minimum: 1.0×1.0×0.4 Maximum: 10×10×0.7		68 to 92	83 to 97		
McLaughlin et al. (2020)	✓	✓	Spall	A: 394 – 3,277					✓
Guldur and Hajjar (2017)		✓	Crack	L: 4.8–30.4; W :2.5					✓
			Spall	L: 29.7; W: 9.4					✓
Bahreini and Hammad (2021)	✓		Crack	Minimum: W: 0.2; D:0.1	45	55	70	62	
			Spall	W:0.5 - 60	73	89	79	84	
SNEPointNet++	✓		Crack	Minimum W: 0.2; D:0.1	69	93	73	82	
			Spall	W:0.5 - 60	83	92	90	91	

\*N.C.: Not Comparable; W: Width; L: Length; A: Area; D: Depth; F1: F1-score; Pre: Precision

### 3.5 Summary and Conclusions

This chapter proposed SNEPointNet++, which is a novel point cloud-based method for semantic segmentation of two types of surface defects (i.e., spalls and cracks) simultaneously and without transforming the point cloud into other representations. This method is able to focus on two main characteristics related to surface defects: normal vector and depth. It also considers the challenges related to the size of the dataset and imbalanced classes. Sensitivity analysis is applied to capture the best combination of hyperparameters and investigate their effects on network performance. A case study was conducted using the collected point cloud dataset from four bridges to validate the proposed method. The created high-quality point cloud dataset, including 1,785 cracks and 2,319 spalls with a minimum width of 2 mm and 5 mm, respectively, is publicly available at <https://github.com/neshatbln/SNEPointNet2/tree/main/Data> for future research in concrete surface defect detection.

Based on the case study, it can be concluded that: (1) in SNEPointNet++, using the main features related to surface defects (i.e., depth, color, and normal vectors) and taking into account the issues related to the point cloud dataset (i.e., small size of the dataset, imbalanced dataset) resulted in 93% (IoU:69.2%) and 92% (IoU:82.5%) recalls for semantic segmentation of cracks and spalls, respectively. Moreover, this network can detect the spalls and cracks of the segments deeper than 7 cm, which have very severe defects, with 99% and 98% recalls, respectively; (2) SNEPointNet++ is more accurate than other DL and non-DL point cloud-based methods. Although one recent DL image-based semantic segmentation method (Lopez Droguett et al., 2020) has higher performance, this can be explained by the much larger dataset used in that research; (3) SNEPointNet++ with the best performance has five sublayers with 2.5 cm, 5 cm, 10 cm, 20 cm, and 30 cm sampling sizes; (4) based on the sensitivity analysis, for the 20 cm x 20 cm blocks, more points per block results in higher efficiency. Moreover, considering 12,288 points per block, decreasing the block size improves the network performance for crack semantic segmentation due to increasing the density of each block, and consequently, providing more crack points. However, the optimum

block size for spall semantic segmentation is 50% of size of the biggest defect and using smaller blocks leads to a drop in the efficiency of the network because of missing the boundaries of spalls in smaller blocks. Additionally, having stride improves the network performance in terms of IoU. However, high stride values are not beneficial in the case of higher number of points per block (i.e., 12,288); and (5) SNEPointNet++ is invariant to the resolution setting of the LiDAR due to using different settings during data collection (i.e., 1/1, 1/2, and 1/4 resolutions). However, in order to detect cracks with width as small as 2 mm, using the highest resolution is required.



# CHAPTER 4 LIDAR-EQUIPPED UAV PATH PLANNING CONSIDERING POTENTIAL LOCATIONS OF DEFECTS FOR BRIDGE INSPECTION

## 4.1 Introduction

According to Section 1.4, in addition to minimizing the flight time of a collision-free path for the UAV, it is crucial to ensure that all critical bridge surfaces are covered more than once from near perpendicular views. Several parameters can influence the accuracy of the data, such as the incidence angle of the laser beam and the distance between the scanner and the structure. Small incidence angles and minimum distances are required to achieve high-quality data and avoid missing some high-risk defects. As mentioned in Section 2.6.4, doing a structural analysis before path planning provides a good perspective about the high-risk spots. The proposed method extends available path planning methods to consider the *level of criticality* of different areas and collect more accurate data from these areas. The proposed path planning method is fully explained in Section 4.3, followed by implementing the method for a hypothetical three-span bridge. The last section includes the conclusions and future work.

## 4.2 Considerations, Requirements, and Constraints

Considering the requirements and constraints gives a better perspective on selecting the most appropriate UAV for bridge inspection. In this section, several factors which are considered directly or indirectly in path planning are explained.

(1) *Mounting location*: Most commercially available solutions mount the scanner under the UAV because they are designed for surveying purposes (Figure 4-1(b)). However, for structural inspection purposes, the LiDAR can be mounted either on top (Figure 4-1(a)) or under the UAV (Figure 4-1(b)), depending on the relative location of the inspected area of the structure.



(a) MIT RANGE (Bachrach et al., 2011)



(b) Bigone 8 Hsepro LiDAR (Italdron Inc., 2019)

Figure 4-1. Scanner position on top of the UAV in (a) and under (b).

(2) *Metrology method*: There are two types of metrology methods for LiDAR: Time-of-Flight (ToF) and Phase-Shift (PS) (Kim et al., 2015b). ToF, sometimes called Pulse-Based, sends a straight laser pulse toward the object with constant power. It can calculate the device-object distance with the measured time the laser pulse takes to travel between the point of emission and the scanned surface. The PS system modulates the power of the beam and sends off a continuous sinusoidal laser beam. To compute the distance, the difference between the emittance phase and reflected laser beams is measured (Pfeifer and Briese, 2007). ToF is useful for measuring long ranges with an accuracy of 4-10 mm at 100 m, while PS functions best in short-range cases of 2-4

mm at 20 m (Kim et al., 2015b). PS scanners have lower ranges, higher measurement rates, and usually better precision than scanners based on ToF (González-Jorge et al., 2012).

(3) *Maximum takeoff weight and payload*: the maximum takeoff weight is the maximum weight at which a UAV is allowed to fly. The payload is the combined weight of all carried devices (e.g., scanner, minicomputer, batteries, and Global Positioning System (GPS)). The payload affects the UAV flight time, as carrying a heavier payload consumes more energy. Scanner weight is one of the major loads in the UAV payload and should be carefully considered. An accurate light scanner is expensive, and choosing the best option is contingent on the available budget.

(4) *Size of UAV*: The shorter arms of smaller UAVs allow them to fly closer to the inspected surface and collect more accurate data. However, the downside of using smaller UAVs is their reduced capacity to carry the scanner and other mounted equipment due to their takeoff-weight limitations.

(5) *Minimum ( $d_{min}$ ) and maximum distances ( $d_{max}$ )*: A specific distance range should be considered during inspection regarding safety issues and scanner characteristics. The density of the scanned point cloud decreases with longer distances.

(6) *Battery capacity*: The battery capacity impacts the flight time. Adding more batteries helps the UAV fly further but increases system weight. The battery capacity may not be enough for scanning all the structure surfaces in one trip. Therefore, several two-way trips should be considered to cover the entire surface (Guerrero and Bestaoui, 2013).

(7) *Vibration*: Vibration of the LiDAR-equipped UAV during inspection causes scanning angle errors, placement angle errors, and flight path errors, which result in an inaccurate 3D point cloud dataset. Designing a suitable LiDAR-equipped UAV platform with an appropriate engine, body shape, and installed dampers can decrease vibration (Li et al., 2015). Moreover, error correction methods consider and analyze the impact of vibration on accuracy (Li et al., 2019).

(8) *DoFs*: As shown in Figure 4-2, the UAV has six DoFs: three displacements ( $x$ ,  $y$ ,  $z$ ) and three rotations (yaw, pitch, and roll). UAV pitch and roll are generally constrained to keep the vehicle horizontally. At least three DoFs should be considered in 3D path planning. Some studies focused on optimizing yaw rotation as well as displacements (Bircher et al., 2015).

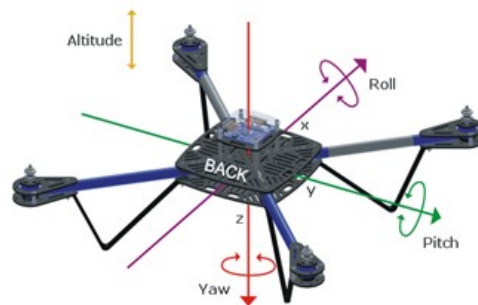


Figure 4-2. Six degrees of freedom of a UAV (Getbestcopter, 2016).

(9) *LiDAR parameters*: The 3D scanners have two FoVs. The FoV is an important parameter in visibility analysis. Other important parameters of the scanner are step size ( $\Delta\theta$ ), incidence angle

( $\alpha_1$ ), and beam diameter ( $d_b$ ). These parameters are shown in Figure 4-3. The accuracy of a point cloud depends mainly on measurement resolution, angular resolution, and scanning speed.

(10) *Safety*: Before the operation, any safety risk related to equipment (i.e., weight, speed, noise), environment (i.e., wind, temperature), jobsite (i.e., location, traffic), mission (i.e., distance to structures) should be determined and managed (Xu and Turkan, 2022). For instance, the center of gravity must be fixed since the UAV stability may change when mounting additional devices. Depending on the controlling method, interruption of GPS signals or remote-control signals may lead to accidents. Moreover, the flight should be conducted in wind-free conditions.

(11) *Regulations*: The safety rules should be followed to ensure safe operation. Flight operation permission may be needed depending on the UAV size to ensure the safety of people, equipment, and buildings. According to Appendix C, which is an example of the application for special flight operation certification, flight permission requires several documents including scan plan, certificates, insurance.

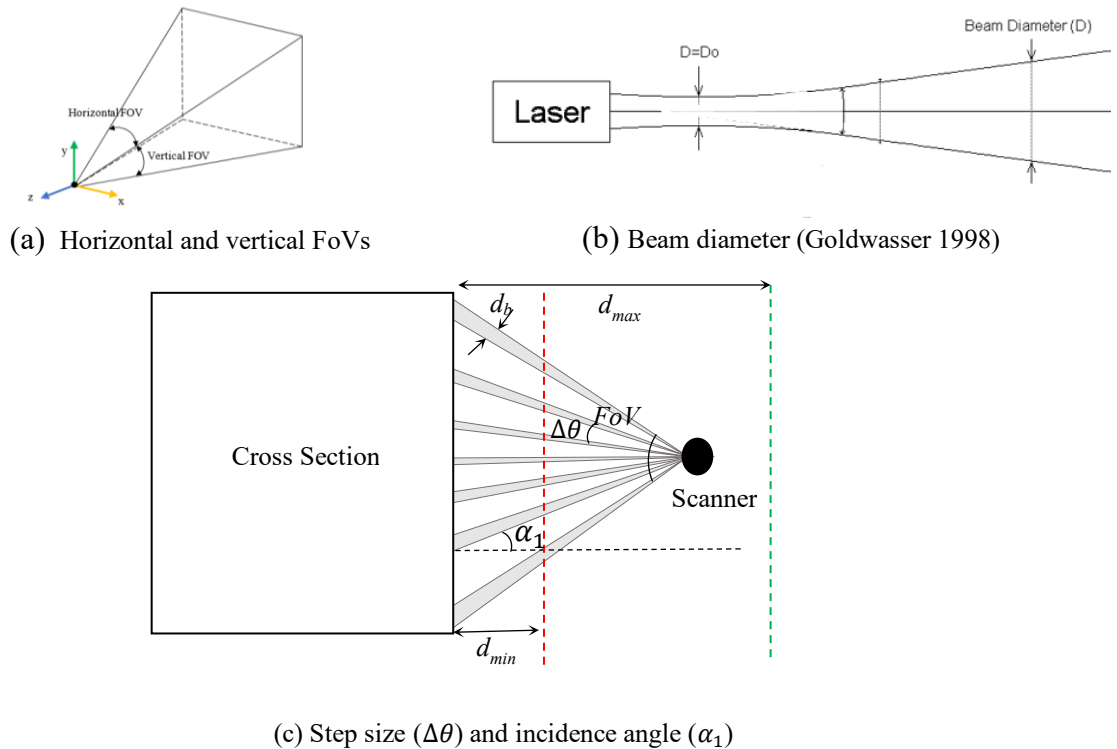


Figure 4-3. LiDAR specifications.

### 4.3 Developing Path Planning Considering Potential Locations of Defects

As mentioned in Sections 1.4, to improve bridge inspection using LiDAR-equipped UAV, a new path planning method is proposed in this research where a UAV will fly according to a generated path to scan the surface and collect point clouds considering several requirements and constraints.

The proposed method defines potentially damaged areas using structural analysis and IVs matching the levels of criticality (low, medium, high). IVs are assigned to cells created on the bridge surface. Next, VPIs are determined, through which the UAV should pass. VPIs help the

scanner collect data at corresponding locations from the shortest allowable distances and near perpendicular angles. Surface visibility from the VPIs is calculated using ray tracing. Finding the optimal path requires two steps: (1) calculating the path length matrix using A\* to find a collision-free path between two VPIs separated by an obstacle; and (2) solving the TSP using GA, taking all VPIs into consideration. The objective function for evaluating the optimal path is the minimum path length with acceptable visibility. Figure 4-4 shows the proposed framework for planning an obstacle-free path accounting for potential locations of defects. Further details are explained below.

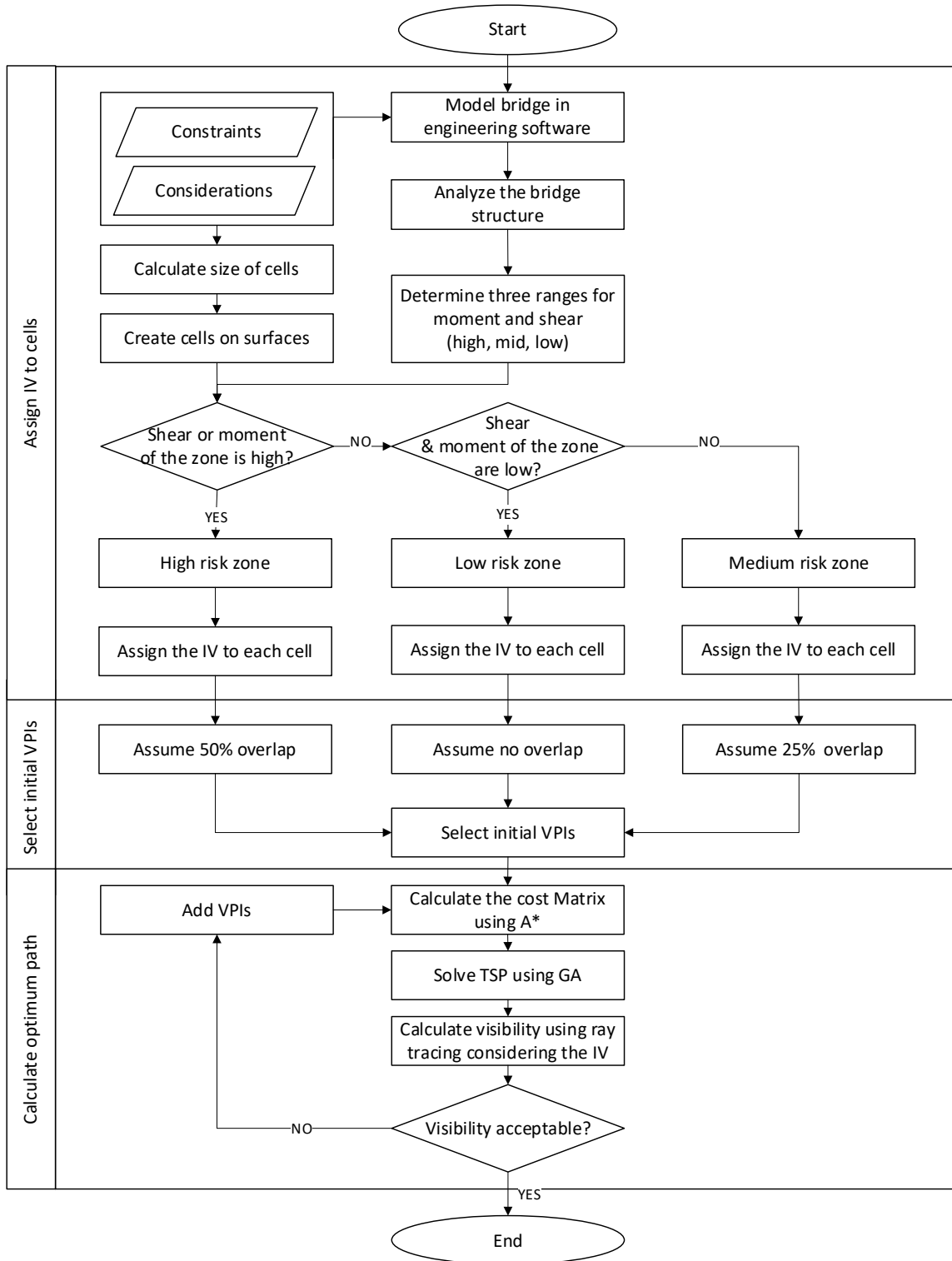


Figure 4-4. Proposed framework for path planning.

Since the proposed method is model-based, the bridge model should be accessible. If the BrIM model is available and compatible with the structural analysis software, it can be imported into the software. Otherwise, the 3D structure should be modeled based on available documents and

information. The proposed method has three main parts: (1) Assign IVs to cells, (2) select initial VPIs, and (3) calculate an optimum path.

### 4.3.1 Assign IVs to Cells

First, the cell size ( $C_s$ ) is computed prior to bridge surface division into cells for detailed visibility analysis. As Figure 4-5 demonstrates, the minimum cell size (equivalent to minimum scan spacing) can be calculated via Equation 4-1. This size should be small enough to achieve high accuracy in proportion to scanner resolution.

$$C_{s,min} = \Delta_{min} = \frac{d_{min} \times \alpha_R}{\cos(\alpha_1)} \quad \text{Equation 4-1}$$

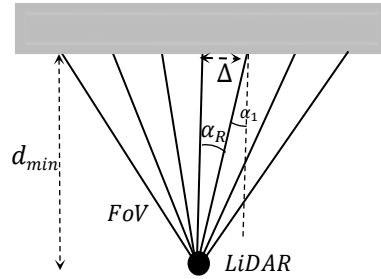


Figure 4-5. Calculating minimum cell size.

where  $\Delta_{min}$ : minimum scan spacing (m);  $d_{min}$ : minimum scan distance between inspected surface and scanner (m);  $\alpha_R$ : angular resolution (radian); and  $\alpha_1$ : incidence angle (degree). The maximum cell size ( $C_{s,max}$ ) depends on the FoV and can be calculated based on Equation 4-2.

$$C_{s,max} = 2 \tan\left(\frac{FoV_{min}}{2}\right) d_{min} \quad \text{Equation 4-2}$$

Then, the bridge deck surface, which should be inspected, is meshed into equal cells based on the calculated size. Figure 4-6 shows the bottom surface of a 2-span bridge deck meshed into equal cells. The total number of cells ( $C_t$ ) can be calculated using Equation 4-3.

$$C_t = \sum_{i=1}^k \frac{W_D \times l_i}{C_s^2} \quad \text{Equation 4-3}$$

where  $k$  is the number of spans, and  $W_D$  and  $l_i$  are the width of the deck and the length of the  $i^{th}$  bridge span (unsupported part), respectively.

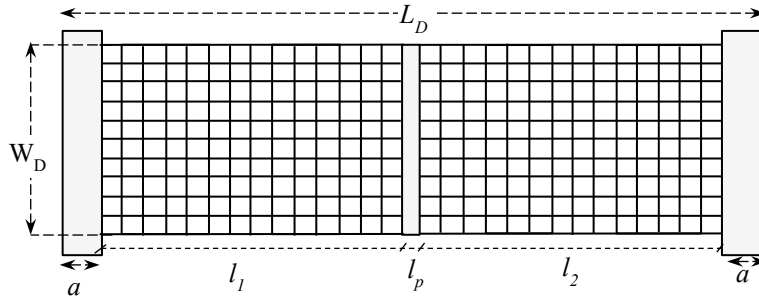


Figure 4-6. The meshed bridge surface.

On the other hand, based on bridge design codes (e.g., American Association of State Highway and Transportation Officials (AASHTO), Canadian Standards Association (CSA) S6), loads (i.e., dead loads, vehicle live load, pedestrian live load) are applied to the bridge model to perform the structural analysis and find the bridge deformation, bending moment, and shear diagrams. The absolute values of the calculated moment ( $M$ ) and shear ( $V$ ) on the bridge are split into three equal ranges (low, medium, high) as follows.

Table 4-1. Determine three ranges for shear and bending moment.

Range	Shear (kN)	Bending Moment (kN.m)
High	$\frac{2}{3}  V_{max}  \leq  V $	$\frac{2}{3}  M_{max}  \leq  M $
Medium	$\frac{1}{3}  V_{max}  \leq  V  < \frac{2}{3}  V_{max} $	$\frac{1}{3}  M_{max}  \leq  M  < \frac{2}{3}  M_{max} $
Low	$ V  < \frac{1}{3}  V_{max} $	$ M  < \frac{1}{3}  M_{max} $

As shown in Figure 4-7, each row of cells is considered as one zone and its level of criticality (low, medium, or high) is defined based on the magnitude of the moment and shear at the middle of the row as follows: (1) A zone in high moment or high shear range is considered high-risk. (2) If both shear and moment are low in a zone, it is assumed a low-risk zone; and (3) All remaining zones are considered medium-risk. It is assumed that the zones with the high level of criticality have more potential for surface defects. Each cell inherits an IV based on the zone it occupies (Figure 4-7). Figure 4-8 shows the cells grouped into three zones: high-risk (red), medium-risk (yellow), and low-risk (green), where  $L_D$ ,  $a$ ,  $l_p$ ,  $W_D$ , and  $l_i$  are the total bridge length, the abutment width, the piers width, the deck width, and the length of  $i^{th}$  deck, respectively

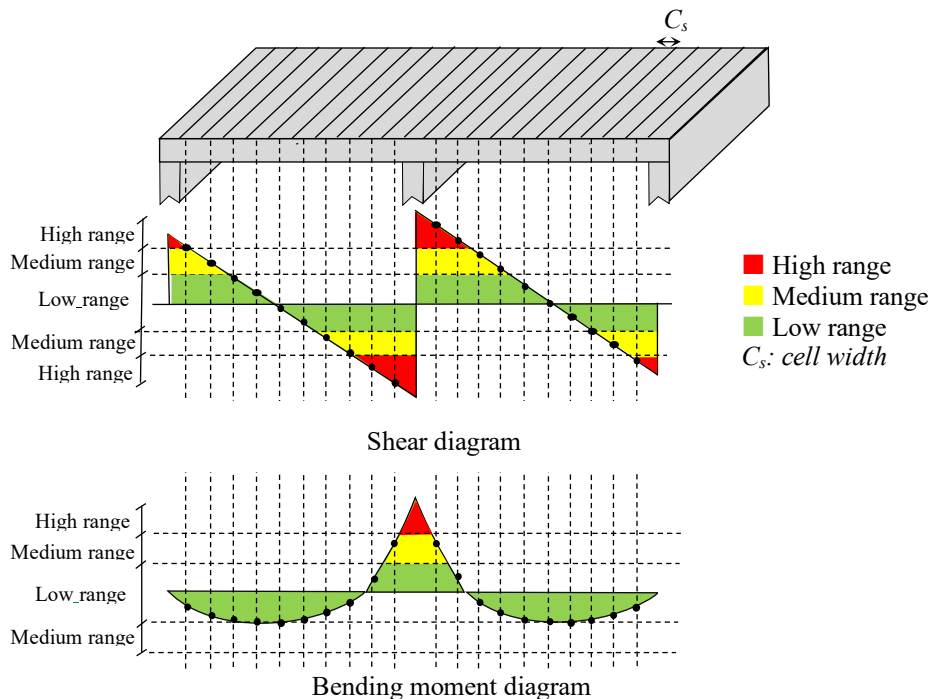


Figure 4-7. Determining the level of criticality of each zone.

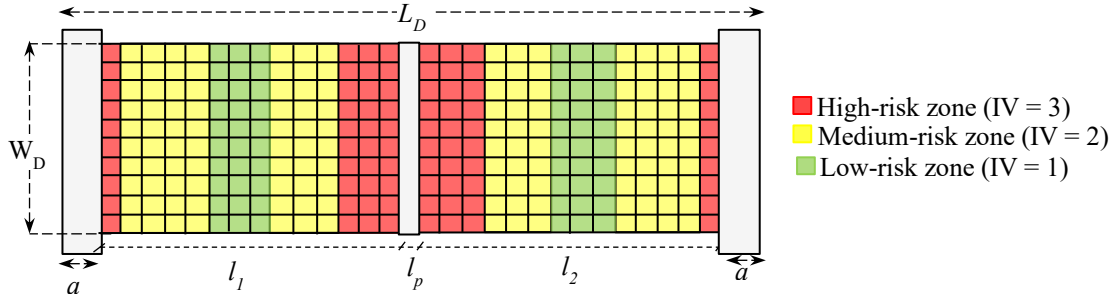


Figure 4-8. Assigning IVs to cells.

### 4.3.2 Select Initial VPIS

As mentioned before, there are two steps for solving TSP. A set of VPIS is generated; then, the path, which passes through each VPI once and returns to the first point, is found. Despite many methods based on minimizing the number of VPIS, the proposed method focuses on selecting the best set of VPIS with respect to the level of criticality. To have an accurate data collection, the laser beams have to reach the defect edges that may not be possible in some cases, such as the large incidence angle, obstacle existence between the defect and LiDAR, and large step size. Consequently, to decrease the errors caused by these issues and to improve the accuracy of VPIS selection, two factors are considered:

(a) In the case of perpendicular views of the inspected surface, the laser beam can reach all parts of the defects, raising accuracy in detecting defect size.

Considering perpendicular view for each VPI leads to providing the smallest possible incidence angles in each FoV. Moreover, smaller incidence angle and consequently denser point clouds result in higher accuracy in detecting the size of the defect. As shown in Figure 4-9, the actual width ( $w$ ) and depth ( $d$ ) of defect B, located far from the VPI<sub>1</sub> (the center of the LiDAR), are not equal the detected width ( $w'$ ) and depth ( $d'$ ), while the actual and detected sizes of defect A are the same (incidence angle =  $0^\circ$ ). In addition, both defect A and defect B are mis-detected if an obstacle blocks some LiDAR rays as shown in Figure 4-10.

(b) Another factor is overlapping views. Scanning defects from more than one VPI and providing overlapping views give a better estimation of the defect size because the possibility of the laser beam reaching the defect edges is increased, as shown in Figure 4-9. Moreover, providing overlapping views leads to collecting denser point clouds, which results in higher accuracy. Therefore, to increase the collected data accuracy, the overlapping views can be considered in VPI selection. Since providing overlapping views for the entire surface is time-consuming, this factor is considered only for high-risk zones. Moreover, too many overlaps may lead to errors due to overfitting in the registration of point clouds (Tam et al., 2013).



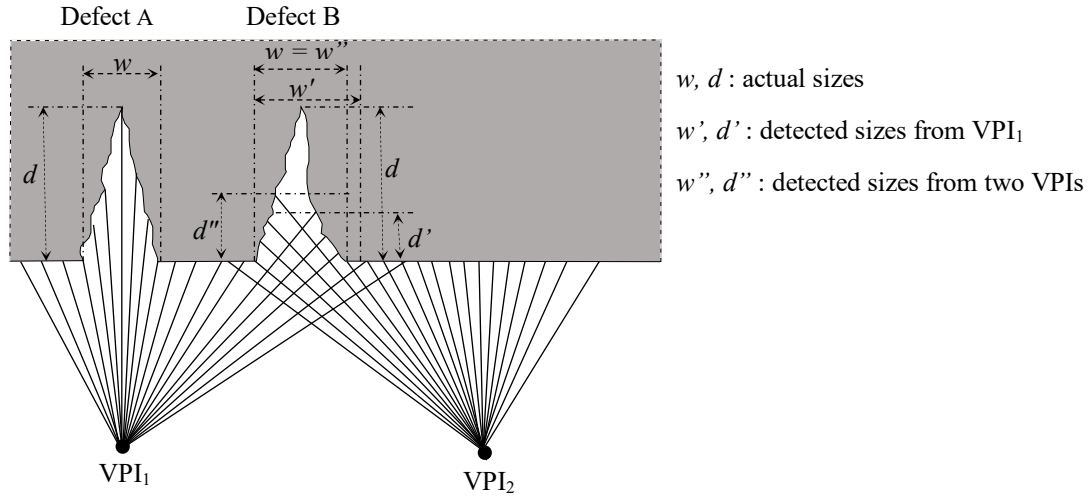


Figure 4-9. Scanning defects.

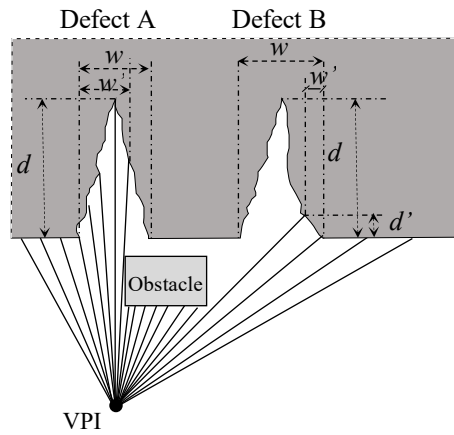


Figure 4-10. Scanning defects in case of an obstacle.

As Figure 4-11 shows, the maximum visible area from a minimum distance ( $d_{min}$ ) translates to  $l_H \times l_V$ , where  $l_H$  and  $l_V$  are calculated using Equations 4-4 and 4-5, respectively.

$$l_H = 2d_{min} \tan(FoV_H/2) \quad \text{Equation 4-4}$$

$$l_V = 2d_{min} \tan(FoV_V/2) \quad \text{Equation 4-5}$$

where  $FoV_H$  is horizontal and  $FoV_V$  vertical.

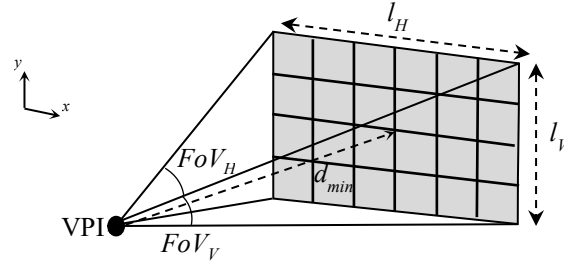


Figure 4-11. Maximum visible area from VPI located at minimum distance from surface.

Figure 4-12 details VPIs distribution under a bridge deck. This distribution follows the following rules:

(a) For full coverage in the direction of the length of the deck (x-axis), VPIs are placed on several rows along the length of the deck separately a distance of  $l_H$ . Equation 4-6 can calculate the number of rows ( $P_x$ ) using  $L_D$  (the deck length),  $a$  (the abutment width), and  $l_p$  (the piers width).

$$P_x = (L_D - a - l_p) / l_H \quad \text{Equation 4-6}$$

(b) Selecting the number of VPIs and their distances in the y-direction depends on the criticality level of their corresponding row. Considering a 50% overlap view for VPIs in high-risk zones with perpendicular views, each cell can be scanned at least twice during data collection. Consequently, the distance between those VPIs should be  $l_V/2$ . In the mid-risk zones, VPIs are located  $3l_V/4$  far from each other to provide 25% overlap. To cover all low-risk zones without overlap, distances between corresponding VPIs are considered equal to  $l_V$ .

Equation 4-7 calculates the number of VPIs per row ( $P_y$ ), and Equation 4-8 calculates the total number of VPIs ( $P_{tot}$ ).

$$P_y = \begin{cases} 2W_D/l_V & \text{for High risk zone} \\ 4W_D/3l_V & \text{for Mid risk zone} \\ W_D/l_V & \text{for low risk zone} \end{cases} \quad \text{Equation 4-7}$$

$$P_{tot} = \sum_1^{P_x} P_y \quad \text{Equation 4-8}$$

where  $W_D$  is the deck width.

The average overlapping count indicates the average number of times a cell appears from scanning from different angles and can be calculated using Equation 4-9.

$$\text{Overlapping count} = \sum_{i=1}^N CS_i / N \quad \text{Equation 4-9}$$

where  $CS_i$  is equal to the number of times that cell  $i$  is seen and  $N$  is the total number of cells.

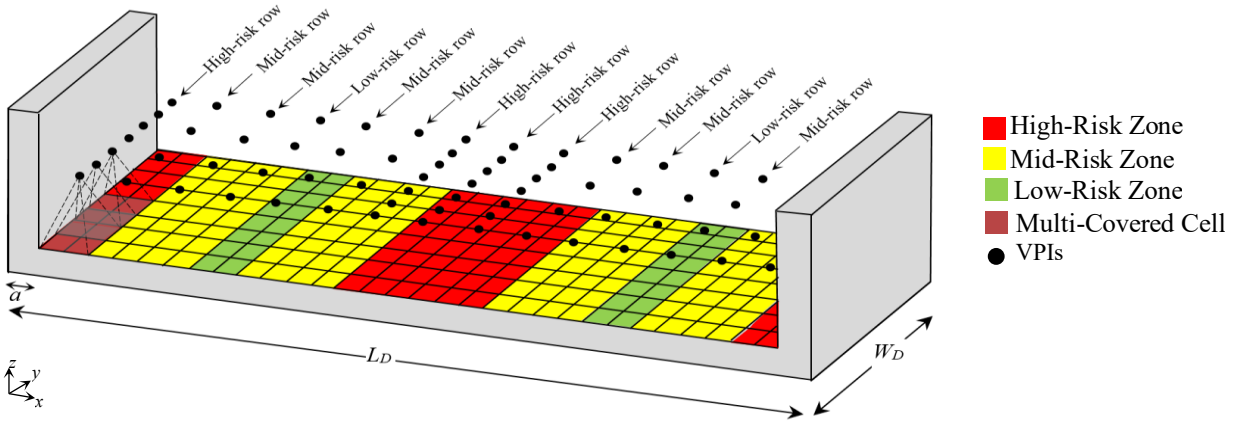


Figure 4-12. Selecting VPIs based on criticality levels.

### 4.3.3 Calculate the Optimum Path

In the proposed method, path planning is based on solving TSP using GA. The fitness function in this method minimizes path length, computed based on Equation 4-10.

$$\text{Minimum path length} = \min \sum_{k=1}^{N-1} l_k \quad \text{Equation 4-10}$$

where  $N$  is the total number of VPIs through which the path should pass.  $l_k$  is the obstacle-free path length between two VPIs, selected from  $N(N-1)$  paths between all VPIs.

Therefore, the path length matrix is first calculated as shown in Figure 4-13, where  $l_{ij}$  and  $L_{N \times N}$  are the distance between points  $i$  and  $j$  and the path-length matrix, respectively. If there is no obstacle between two VPIs, the path length, which is a direct line between point  $i$  and  $j$ , is equal to  $\sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ . Otherwise, the path length is calculated based on the generated path between two points using A\*.

---

**Algorithm: path length matrix calculation**

---

```
1:  $i, j \leftarrow 0$ 
2: while  $i < N$  do
3:    $l_{ij} = 0$ 
3:    $j \leftarrow i+1$ 
4:   while  $j < N$  do
5:     if there is an obstacle between point  $i$  and point  $j$ 
6:       Calculate  $l_{ij} = l_{ji}$  using A* between points  $i$  and point  $j$ 
7:     else
8:        $l_{ij} = l_{ji} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ 
9:        $j \leftarrow j+1$ 
10:    end while
11:     $i \leftarrow i+1$ 
12: end while
13: return path length matrix
```

---

Figure 4-13. Path length matrix calculation pseudocode.

The main GA operators are selection, crossover, and mutation. The GA steps are:

Step 1. Population initialization: many feasible paths, which can be parents for the next generation, are generated, and the fitness value (path length) of each path is computed.

Step 2. Selection: two parents are picked from the population to produce off-springs.

Step 3. Crossover: an operator for generating new children from parents.

Step 4. Mutation: a random process based on swapping the sequence of points. Occasionally, two points are picked and their location on the path is switched.

Step 5. Evaluation: evaluating the off-springs based on fitness values.

Step 6. Replacement: if the new offspring is fitter than the previous fittest path, it is considered as the fittest path.

Step 7. Repeating: the number of generations ( $N$ ) is pre-determined and steps 2 to 6 are repeated  $N$  times.

After selecting the shortest path, to calculate total coverage, the visibility from each VPI is calculated using ray tracing and considering levels of criticality. This method is based on previous work (Albahri and Hammad, 2017) for calculating surveillance cameras coverage in buildings and is modified to calculate LiDAR-equipped UAV coverage in this research. To reflect the level of criticality in this step, weighted values are calculated, which depend on the number of cells and their corresponding IV. Therefore, covering the most critical areas is prioritized over any other part of the inspected surface. Total coverage is computed as in the following equations:

$$W_{a_i} = IV_i \sum_{j=1}^n C_{ij} \quad \text{Equation 4-11}$$

$$CC_{a_i} = IV_i \sum_{v=1}^n C_{iv} \quad \text{Equation 4-12}$$

$$\text{Total Coverage} = \frac{\sum_{i=1}^3 CC_{a_i}}{\sum_{i=1}^3 W_{a_i}} \quad \text{Equation 4-13}$$

where  $i$  (1:3) is the number of zone levels of criticality (low, medium, and high).  $W_{a_i}$  is the weight of the area  $a_i$ ,  $C_{ij} = 1$  represents the cell  $j$  in the  $i^{\text{th}}$  level, and  $n$  is the total number of cells in the  $i^{\text{th}}$  zone level.  $IV_i$  is the importance value assigned to all cells in zone level  $i$ ,  $C_{iv} = 1$  is the covered

cell  $v$  in zone level  $i$  and  $n'$  is the total number of covered cells in  $i^{th}$  zone level.  $CC_{a_i}$  is the weighted covered cells in area  $a_i$ . If there is inadequate visibility, new VPIs should be added to the low-risk zone and the path updated.

#### 4.4 Implementation and Case Study

To demonstrate the applicability of the proposed algorithm to typical bridge inspection problems, a four-span concrete bridge, located in Alberta, Canada, is used in the case study. As shown in Figure 4-14, the bridge consists of four lanes with two sidewalks, covering 15.6 m in width and 62 m in length. A Velodyne LiDAR PUCK mounted on a UAV (Matrice 100) facing upward is hypothetically considered as the equipment to scan the bottom surface of the bridge (Velodyne LiDAR Inc., 2015). The payload of the UAV including the LiDAR (590 g) and other parts is around 1 kg (Papachristos et al., 2019). Based on Table 4-2, although the flight range can be further increased using an additional battery, this is not possible because the extra battery will exceed the payload of the UAV. The relation between the flight time of a Matrice 100 and its payload using a TB48D battery is shown in Figure 4-15. The figure also shows that, with the LiDAR weight, a maximum flight time of 16 minutes can be achieved for the payload 1 kg.



Figure 4-14. Inspected bridge in Alberta, Canada.

Table 4-2. Takeoff weight and flight time for Matrix 100.

	Max Takeoff Weight (g)	Weight with (battery/ies) (g)	Max payload (g)	Flight time (no payload) (min)
with one battery	3,600	2,355	1,245	28
with two batteries	3,600	3,031	569	40

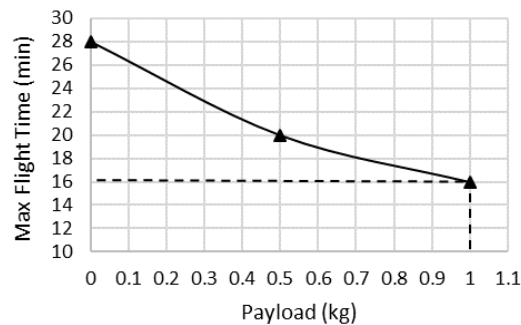


Figure 4-15. Relation between flight time and payload for Matrix 100.

CSiBridge 2017 (Computers and Structures Inc., 2017), a structural bridge design software that supports importing and exporting of International Foundation Class (IFC) file format, is used to model the bridge. Based on the CSA S6-14 – Canadian Highway Bridge Design Code (Canadian Standards Association, 2014), the loads are applied to calculate the bending moment ( $M$ ) and shear ( $V$ ) of the sections. Figure 4-16 and Figure 4-17 show the modeled bridge structure and the structural analysis results including bending moment and shear diagrams, respectively.



Figure 4-16. Modeled bridge structure.

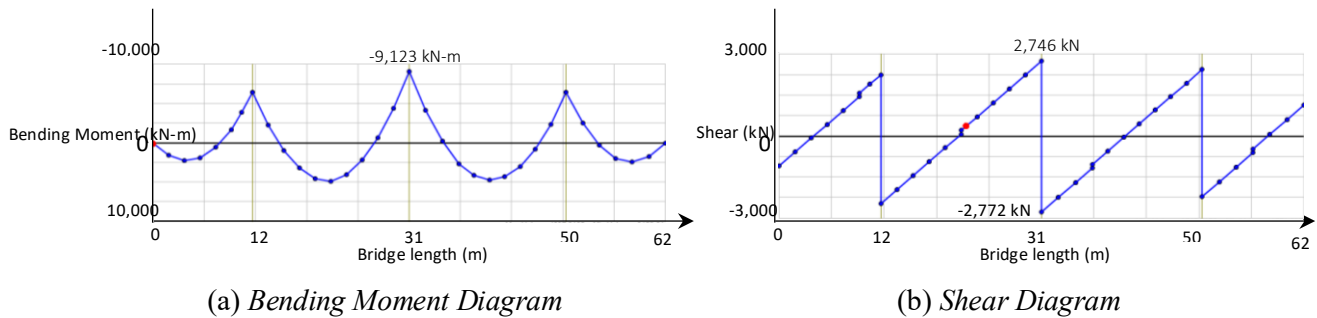


Figure 4-17. Results of bridge structural analysis.

The calculated bending moments and shear forces are classified into three groups based on Table 4-1.

$$\begin{cases} 0 \leq |M| < 3041 \text{ kN.m} & : \text{low } M \\ 3041 \text{ kN.m} \leq |M| < 6082 \text{ kN.m} & : \text{medium } M \\ 6082 \text{ kN.m} \leq |M| & : \text{high } M \end{cases} \quad \text{Equation 4-14}$$

$$\begin{cases} 0 \leq |V| < 924 \text{ kN} & : \text{low } V \\ 924 \text{ kN} \leq |V| < 1848 \text{ kN} & : \text{medium } V \\ 1848 \text{ kN} \leq |V| & : \text{high } V \end{cases} \quad \text{Equation 4-15}$$

The bottom surface of the bridge deck is divided into rows with the maximum 3.9-meter width, calculated using Equation 4-4. Based on the criticality levels, three defined IVs (1, 2, 3) are assigned to the rows including 6 rows in high-risk zones, and 8 and 2 in mid- and low-risk zones, respectively. Assuming 50%, 25%, and 0% overlaps for high-, mid-, and low-risk zones, respectively, and using Equations 4-4 to 4-8, 90 VPIs are sampled and distributed in the 16 rows as shown in Figure 4-18.

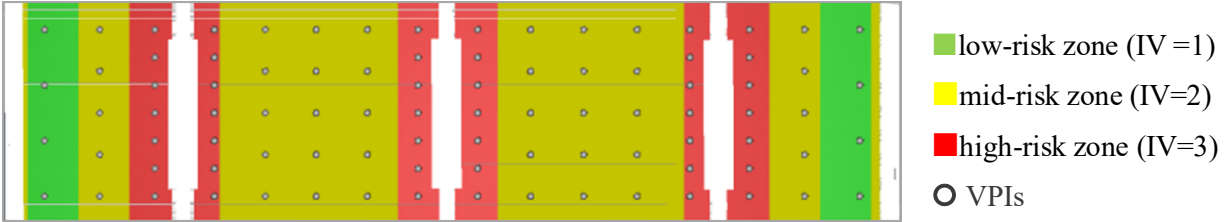


Figure 4-18. Risk zones and sampled VPIs (bottom view of bridge).

The path planning phase is done in a cross-platform game engine called Unity 3D (Unity Technologies, 2018), which is supported in Windows using the programming language C#. In Unity, the files can be imported as assets including the attributes of all elements separately (e.g., name, size). Also, Unity includes a class, called NavMesh, for pathfinding between two objects in the meshed area using the A\* graph search algorithm. In this phase, Revit 2017 (Autodesk Inc., 2019) works as an intermediary software between CSiBridge and Unity 3D. The bridge structure is imported into Revit 2017 in RVT format and then exported as a file in Film Box (FBX) format for use in Unity 3D (Figure 4-19).

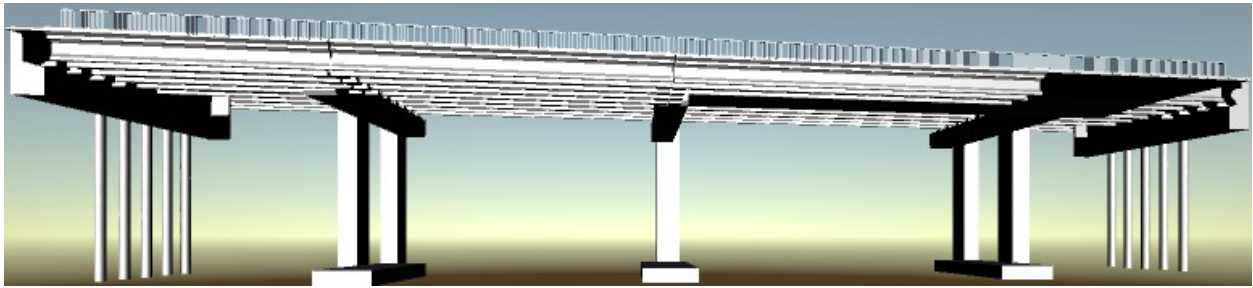


Figure 4-19. Importing the model into Unity.

A safe flight requires a minimum distance from the bridge ( $d_{min,z}$ ) of 1 m. According to the UAV arm size and the space required for rotation,  $d_{min,x}$  and  $d_{min,y}$  are assumed as 60 cm. The selected value for the angular resolution  $\alpha_R$  is  $2^\circ$ . Assuming the horizontal and vertical FoVs are equal to  $105^\circ$  and using Equations 4-1 and 4-2, the cell size should be between 3.5 cm and 285.6 cm. To speed up the calculations, the surfaces under the bridge are divided into  $50 \times 50 \text{ cm}^2$  cells. Figure 4-20 shows 4,816 cells created on the surfaces of the beams and the deck, of which 602, 2838, and 1376 cells belong to low-, mid-, and high-risk zones, respectively. 3,024 cells are created on horizontal surfaces (the lower surfaces of the deck and beams) and 1,792 cells on vertical surfaces (the side surfaces of the beams).

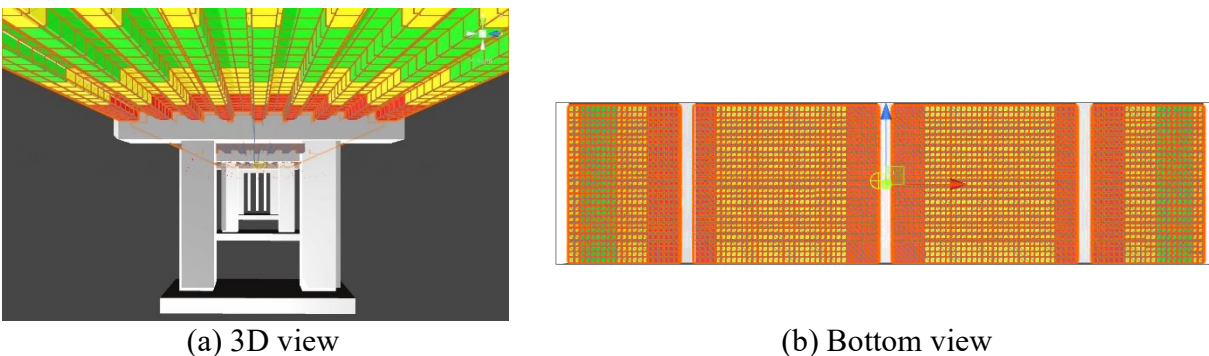


Figure 4-20. Meshed surface of the bridge.

A  $90 \times 90$  path length matrix is calculated using A\*, which is used to solve TSP and determine a path passing through all VPIs. As shown in Figure 4-21, the results of running the optimization four times with 15,000 generations, where the size of the population is 100, vary between 275.5 m and 276.8 m. Each iteration takes less than 1 minute on a personal computer with Intel®Core™i7-8700 CPU (3.20 GHz) and 16 GB RAM. The time complexity is  $O(mn)$ , where  $m$  and  $n$  are the population size and the number of VPIs, respectively (Kang et al., 2016). The pink line in Figure 4-22 is the visualization of the shortest 3D obstacle-free path with 275.5 m length and 100% visibility. The visible area is shown in blue. Assuming a 0.5 m/s speed, the UAV flight would take 9.2 min to scan the bridge deck. Based on Figure 4-15, this time is within the flying time (i.e., 16 min).

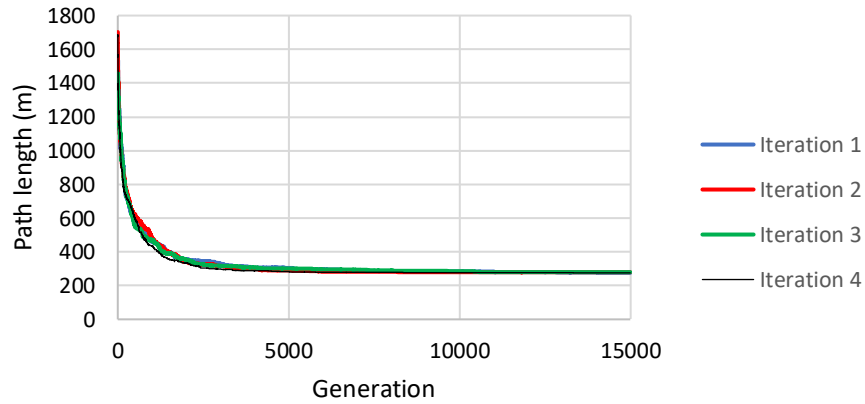
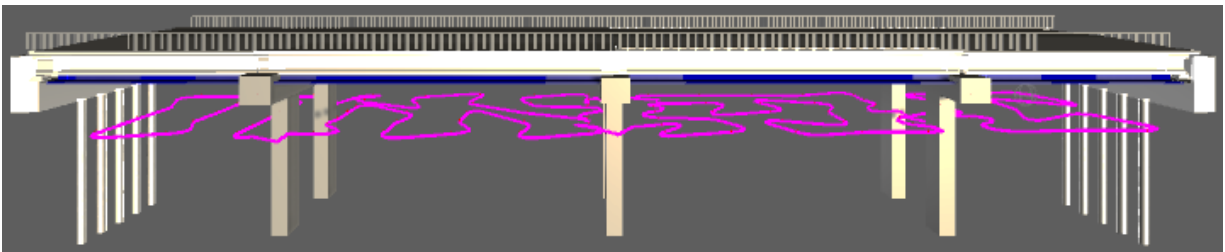


Figure 4-21. Four optimization results.



(a) 3D view



(b) Bottom view

Figure 4-22. Visual representation for the calculated path.

As explained in Section 4.3, a smaller incidence angle results in higher accuracy in detecting the size of the defect. Because the assumed FoVs of the LiDAR are  $105^\circ$ , the possible range of the incidence angle for vertical surfaces of the beams is between  $37.5^\circ$  and  $90^\circ$  and for horizontal



surfaces is between  $0^\circ$  and  $52.5^\circ$ . Figure 4-23 shows the cumulative distribution of the percentage of cells in each zone at any given incidence angle. As expected, the maximum incidence angle in the high-risk zones is  $52^\circ$  which is less than that of the mid- and low-risk zones ( $62^\circ$ ). Increasing the overlapping views resulted in better incidence angles for both horizontal and vertical cells.

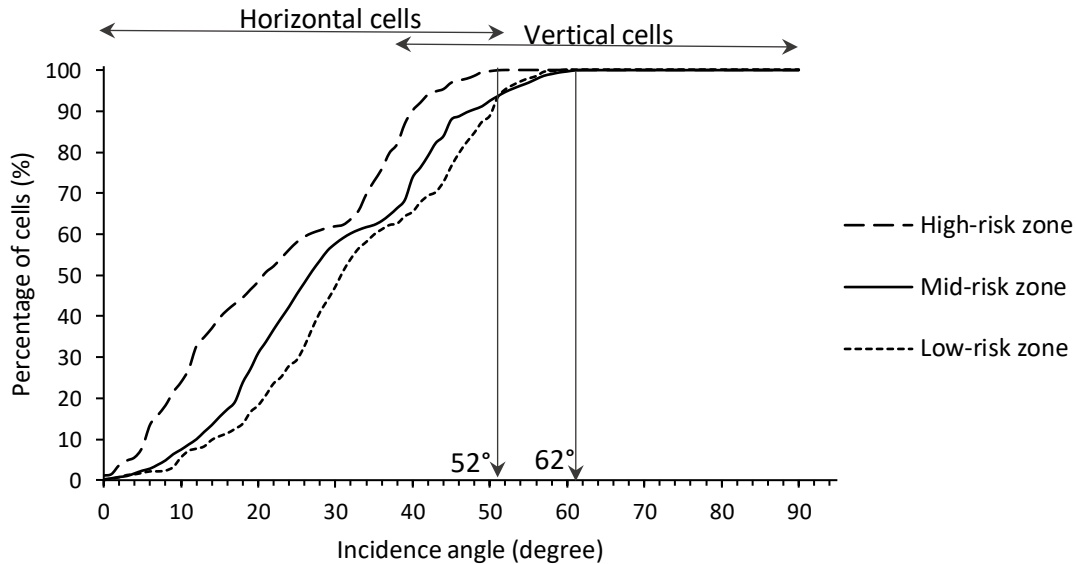


Figure 4-23. Percentage of cells in each zone at any given incidence angle.

To examine the influence of the overlapping percentages on the path quality, which is evaluated based on incidence angles and overlapping counts, three sets of overlapping percentages are defined: (1) 50% for all zones, (2) 50%, 25%, and 0% for high-, mid-, and low-risk zones, and (3) no overlapping views. The number of VPIs and the calculated path lengths are shown in Table 4-3. As shown in Table 4-3, decreasing the overlapping percentages leads to fewer VPIs, resulting in shorter paths. Although minimizing the path is one of the main objectives of path planning, the accuracy of the collected data is important. For the first set of VPIs, the average incidence angles and overlapping counts for all zones are  $22.9^\circ$  and 2, respectively. Compared to the first set, considering no overlaps for all zones in the third set results in a 12% decrease in the path length and a 43% increase in the average incidence angles.

Moreover, in the first set, the overlapping counts for all zones is double that of the third set. Although scanning a cell from different views increases the accuracy of the point cloud, it also increases the size of the point cloud. The second set is defined to take advantage of overlapping views based on the level of criticality. Compared to the third set, the path length of the second set increases by 9%, while the average incidence angles decrease by 43% and 11% for high- and mid-risk zones, respectively. More overlapping counts means higher density and accuracy for the point clouds belonging to more risky areas.

Table 4-3. Path planning results for three cases.

Set	Overlapping percentage			Average incidence angle (°)			Average overlapping count			Path length (m)	Number of VPIs
	Low-risk	Mid-risk	High-risk	Low-risk	Mid-risk	High-risk	Low-risk	Mid-risk	High-risk		
1	50	50	50	22.9	22.9	22.9	2	2	2	284.6	112
2	0	25	50	32.8	29.5	22.9	1	1.3	2	275.5	90
3	0	0	0	32.8	32.8	32.8	1	1	1	249.4	64

## 4.5 Conclusions

The path planning method is proposed to inspect bridges and detect surface defects. One of the main advantages of the proposed method is that LiDAR and UAV are used to collect high-quality 3D point clouds safely at close distances. In order to have a safer inspection, the point cloud is collected using an obstacle-free path for a UAV. Moreover, automated approaches in all three main steps of the proposed methodology, finding minimum path length (using GA), obstacle avoidance (using A\*) and coverage calculation (using ray tracing), leads to shorter processing time with less workload. The conclusions of this research are as follows: (1) The proposed method can find an obstacle-free path with minimum length and acceptable visibility. The result of this method is near-optimal, reliable, and time-effective. (2) the method gives the inspector the perspective on potential locations of surface defects to provide perpendicular and overlapping views efficiently in sampling the VPIs. Considering the highest overlapping views for high-risk zones improves the accuracy of the detected size of defects and avoids spending too much time on scanning the medium- and low-risk zones. (3) Calculating the visibility with respect to the level of criticality leads to prioritizing covering the high-risk zones.

As shown in the proposed framework, the algorithm includes two main steps: defining the viewpoints and finding the shortest path. Path planning can be applied to any type of sensor (e.g., camera) and is not specific to LiDAR. However, the first step has been tailored for LiDAR by considering the minimum incidence angle and perpendicular view in selecting VPIs, which can be adjusted in case of using a camera.

## CHAPTER 5 SUMMARY, CONTRIBUTIONS, AND FUTURE WORK

### 5.1 Summary of Research

This research aims to improve bridge inspection by proposing an efficient and automated data collection method and a DL-based defect detection method. Its focus was placed on semantic segmentation of concrete surface defects (i.e., cracks and spalls) using point clouds and LiDAR-equipped UAV path planning. This research covered a comprehensive review of the related literature and identified the research gaps in this domain. The proposed methods were explained in detail and their applicability was validated in case studies.

In Chapter 2, the literature review, different concrete surface defect detection techniques using point clouds and semantic segmentation DL methods were reviewed. Then, the concept of path planning methods and the related previous studies were discussed to find the best approach to bridge the gaps.

Chapter 3 focus was on creating a publicly available dataset and developing a semantic segmentation method for concrete surface defects (i.e., spalls, cracks) using an adjusted network called SNEPointNet++. The aspects considered in the development of SNEPointNet++ based on PointNet++ were mainly regarding the differences between the application and classes of these networks, such as the small size of defects, the most beneficial features of defects (i.e., depth, normal vectors), and less points in the classes with higher priority (i.e., cracks, spalls). To this end, a point cloud dataset was first created based on the scans from four bridges in Montreal. The dataset was then augmented by flipping horizontally and vertically. This dataset, which contained the coordinates, color, normal vector, and depth of the points, was used as the input of SNEPointNet++. The model was trained and evaluated using 60% and 20% of the dataset, respectively, and tested using the remaining unseen part. Sensitivity analysis was subsequently applied to capture the best values for the hyperparameters of the network.

The focus of Chapter 4 was on the path planning of a LiDAR-equipped UAV considering the potential locations of defects. The proposed path planning method was applied and simulated on a 3-span bridge in Alberta, Canada. A\* and GA were used to find the shortest path through VPis selected based on the scanning requirements and the level of criticality of different parts of the bridge.

### 5.2 Research Contributions and Conclusions

This research has the following contributions:

- (1) Developing a point cloud-based DL method for semantic segmentation of concrete bridge surface defects (e.g., cracks and spalls), called SNEPointNet++, considering the main features for defects (i.e., color, depth, and normal vector) and taking into account the issues related to the point cloud dataset (i.e., smaller size of the dataset, imbalanced dataset). The following conclusions can be drawn from this contribution:
  - SNEPointNet++ was able to detect spalls and cracks simultaneously and without transforming the point cloud into other representations with 93% (IoU:69.2%) and 92% (IoU:82.5%) recalls, respectively. Moreover, the results demonstrated the effectiveness of the proposed network in detecting defects of deeper segments (i.e., up to 99% recall for spalls in the segments deeper than 7 cm).

- SNEPointNet++ is more efficient than other point cloud-based methods as well as most of the currently available image-based DL semantic segmentation methods.
  - Based on the sensitivity analysis, it can be concluded that the most efficient SNEPointNet++ has five sublayers with 2.5 cm, 5 cm, 10 cm, 20 cm, and 30 cm sampling sizes, where the maximum value of the number of points per block (12,288 points per block) and minimum block size (20 cm × 20 cm) are considered. In addition, the network performance can be improved by adding stride.
  - Although SNEPointNet++ is invariant to resolution, a dataset collected by the highest resolution is required to detect small cracks of 2 mm width.
- (2) Developing an obstacle-free path planning method to inspect bridges safely and detect surface defects using LiDAR-equipped UAV considering potential locations of defects. The automated approaches in all three main steps of the proposed methodology, finding minimum path length using GA, obstacle avoidance using A\*, and coverage calculation using ray tracing, lead to shorter processing time. Regarding this contribution, the following conclusions can be drawn:
- The proposed method can find an obstacle-free path with minimum length and acceptable visibility. The result of this method is near-optimal, reliable, and time-effective.
  - The method gives perspective on the potential locations of surface defects to provide perpendicular and overlapping views efficiently in sampling the VPIs. Considering the highest overlapping views for high-risk zones improves the accuracy of the detected size of defects and avoids spending too much time on scanning the medium- and low-risk zones.
  - Calculating the visibility with respect to the level of criticality leads to giving priority to covering the high-risk zones.
- (3) Providing a point cloud dataset to detect two main concrete surface defects, spalls and crack, which is publicly available at SNEPointNet++ Dataset (Bolourian, 2022). The lack of a point cloud dataset for bridge inspection, particularly surface defect detection, is one of the main obstacles of point cloud-based DL methods in this field. By making our dataset publicly available, other researchers can benefit from the dataset for future improvements in this field.

### 5.3 Limitations and Future Work

Despite the above-mentioned contributions, there are some limitations in this research that should be addressed in the future. These limitations are as follows:

- (1) While the proposed LiDAR-equipped UAV path planning method is promising to meet the objectives of this research, the calculated path could not be applied in the site experiments due to the lack of an affordable high-quality light 3D LiDAR that can be mounted on the available UAV (Matric100). To this end, site experiment using an appropriate LiDAR-equipped UAV is one of the future works. Moreover, using LiDAR-equipped UAV flying based on the planned path is highly recommended for future point cloud dataset expansion due to the high efficiency and accuracy of this technique resulting from close range scanning and consideration of several factors (i.e., overlapping views, levels of criticality) in path planning.

- (2) One of the shortcomings of point cloud-based semantic segmentation in this research is the limited size of the dataset. Therefore, the future effort to increase the dataset can focus on collecting more point clouds from different parts of the damaged bridges using UAV and TLS. Moreover, the point clouds can be generated using Generative Adversarial Networks (GAN) (Li et al., 2018) and synthetic point clouds (Mohammadi et al., 2019).
- (3) The defect detection case study is limited to planar surfaces. Although this method was not evaluated on curved surfaces (e.g., columns), it is expected to be still applicable due to consideration of the normal vector. To evaluate the performance of SNEPointNet++, a dataset including curved surfaces should be provided by scanning specific bridge elements such as circular columns or caps of piers. If the performance is not acceptable, considering the curvature as an input may improve the network performance.
- (4) One of the limitations in training SNEPointNet++ was the shortage of computation storage (i.e., RAM). In the future, having more resources will make the opportunity to expand the sensitivity analysis and improve the network performance. For example, since increasing the number of points per block from 10,240 points to 12,288 points significantly improves the network performance, the same trend is expected for larger values of the number of points per block. Moreover, strides can be added to smaller blocks (e.g., 20 cm × 20 cm). Besides, other hyperparameters (i.e., depth and width of PointNet units, number of points in each sample) can be adjusted in SNEPointNet++.
- (5) Inspectors require specific information about defects (i.e., size, severity). Therefore, future efforts should aim to:
  - Classify the dataset into more classes considering their levels of severity (i.e., medium, severe) or type of the cracks (i.e., shear cracks). This approach requires more data in terms of variety and quantity.
  - Extract the defects individually using supervised instance segmentation, clustering, or other non-ML methods, and then measure their dimensions. Moreover, the extracted defects can be transferred into a BrIM model to be used in path planning. Comparing the new as-is BrIM model and the previous one is an efficient approach for evaluating the defect propagation and bridge condition assessment.
- (6) SNEPointNet++ performance was validated based on an augmented dataset including four features (i.e., 3D coordinates, colors, normal vector, depth). However, the effects of each feature and each of the aspects, which were considered for model improvement, were not investigated separately. In the future, these aspects can be applied in separate steps to quantify their individual impact on the network performance.
- (7) The proposed LiDAR-equipped UAV path planning can be further improved by considering operation conditions (i.e., wind, vibration, speed) and more DoFs of the UAV. Moreover, localization and mapping methods (e.g., Simultaneous Localization and Mapping (SLAM)) will be evaluated in case of losing GPS signal under the bridge.

## REFERENCES

- Adhikari, R. S., Moselhi, O., and Bagchi, A. (2014). "Image-based retrieval of concrete crack properties for bridge inspection." *Automation in Construction*, 39 (1): 180–194. DOI: 10.1016/j.autcon.2013.06.011
- Ahlborn, T. M., Shuchman, R., Sutter, L. L., Brooks, C. N., Harris, D. K., Burns, J. W., Endsley, K. A., Evans, D. C., Vaghefi, K., and Oats, R. C. (2010). *An evaluation of commercially available remote sensors for assessing highway bridge condition*. Michigan: Michigan Technological University
- AlBahnassi, H., and Hammad, A. (2011). "Near real-time motion planning and simulation of cranes in construction: Framework and system architecture." *Journal of Computing in Civil Engineering*, 26 (1): 54–63. American Society of Civil Engineers. DOI: 10.1.1.404.477
- Albahri, A. H., and Hammad, A. (2017). "A novel method for calculating camera coverage in buildings using BIM." *Journal of Information Technology in Construction (ITcon)*, 22 (2): 16–33. DOI: <http://www.itcon.org/2017/2>
- Almadhoun, R., Taha, T., Dias, J., Seneviratne, L., and Zweiri Y. (2019). "Coverage Path Planning for Complex Structures Inspection Using Unmanned Aerial Vehicle (UAV)." *Intelligent Robotics and Applications*, 243–266. Switzerland: Springer
- Almadhoun, R., Taha, T., Seneviratne, L., Dias, J., and Cai, G. (2016). "A survey on inspecting structures using robotic systems." *International Journal of Advanced Robotic Systems*, 13 (6): 1–18. DOI: 10.1177/1729881416663664
- Alves, R., Silva de Morais, J., and Yamanaka, K. (2020). "Speeding Up On-Line Route Scheduling for an Autonomous Robot Through Pre-Built Paths." *Robotica*, 1–13. DOI: 10.1017/S0263574720000594
- Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2011). *The traveling salesman problem: a computational study*. Princeton, New Jersey: Princeton university press
- Armeni, I., Sax, S., Zamir, A. R., and Savarese, S. (2017). "Joint 2d-3d-semantic data for indoor scene understanding." *Computer Vision and Pattern Recognition*, 2 (1702.01105). DOI: [doi.org/10.48550/arXiv.1702.01105](https://doi.org/10.48550/arXiv.1702.01105)
- Aryan, A., Bosché, F., and Tang, P. (2021). "Planning for terrestrial laser scanning in construction: A review." *Automation in Construction*, 125: 103551. DOI: 10.1016/j.autcon.2021.103551
- Autodesk Inc. (2019). "Revit." <https://www.autodesk.com/products/revit/overview>
- Autodesk Inc. (2021). "ReCap." <https://www.autodesk.ca/en/products/recap/>
- Bachrach, A., Prentice, S., He, R., and Roy, N. (2011). "RANGE—Robust autonomous navigation in GPS-denied environments." *Journal of Field Robotics*, 28 (5): 644–666. Wiley Online Library. DOI: 10.1002/rob.20400
- Bahreini, F., and Hammad, A. (2021). "Point Cloud Semantic Segmentation of Concrete Surface Defects Using Dynamic Graph CNN." *38th International Symposium on Automation and Robotics in Construction*, 379–386. Dubai, UAE
- Barazzetti, L., and Scaioni, M. (2009). "Crack measurement: Development, testing and applications of an automatic image-based algorithm." *ISPRS Journal of Photogrammetry and Remote Sensing*, 64 (3): 285–296. Elsevier
- Bennett, T. (2015). "Exploring Reality Computing for Civil Infrastructure." <https://informedinfrastructure.com/12620/reality-computing-for-civil-infrastructure/>
- Bien, J., and Zwolski, J. (2007). "Dynamic tests in bridge monitoring—systematics and applications." *25th Conference & Exposition on Structural Dynamics*, 1–10. Orlando, Florida

- Bircher, A., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., and Siegwart, R. (2015). “Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics.” *Proceedings - IEEE International Conference on Robotics and Automation*, 6423–6430
- Bircher, A., Kamel, M., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., and Siegwart, R. (2016a). “Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots.” *Autonomous Robots*, 40 (6): 1059–1078. DOI: 10.1007/s10514-015-9517-1
- Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., and Siegwart, R. (2016b). “Receding horizon” next-best-view” planner for 3Dexploration.” *International Conference on Robotics and Automation (ICRA)*, 1462–1468. Stockholm, Sweden
- Bolourian, N. (2022). SNEPointNet++ Dataset, Accessed June 2022. <https://github.com/neshatbln/SNEPointNet2/tree/main/Data>
- Boon, K., and Lovelace, D. C. (2014). *The domestic use of unmanned aerial vehicles*. USA: Oxford University Press
- Borna, K., and Khezri, R. (2015). “A combination of genetic algorithm and particle swarm optimization method for solving traveling salesman problem.” *Cogent Mathematics*, 2 (1): 1048581. DOI: 10.1080/23311835.2015.1048581
- Boulch, A., Guerry, J., Le Saux, B., and Audebert, N. (2018). “SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks.” *Computers & Graphics*, 71: 189–198
- Brownlee, J. (2014). “An Introduction to Feature Selection - Machine Learning Mastery.” *Data Preparation*. Accessed May 24, 2021. <https://machinelearningmastery.com/an-introduction-to-feature-selection/>
- Brownlee, J. (2016). “How To Improve Deep Learning Performance.” Accessed May 23, 2021. <https://machinelearningmastery.com/improve-deep-learning-performance/>
- Burri, M., Nikolic, J., Hürzeler, C., Caprari, G., and Siegwart, R. (2012). “Aerial service robots for visual inspection of thermal power plant boiler systems.” *2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, 70–75. Zurich, Switzerland
- Canadian Standards Association. (2014). *CSA S6-14 – Canadian Highway Bridge Design Code*. Canada: CSA Group
- Cao, C., Zhang, J., Travers, M., and Choset, H. (2020). “Hierarchical Coverage Path Planning in Complex 3D Environments.” *International Conference on Robotics and Automation (ICRA)*, 3206–3212. Paris, France: IEEE
- Castellanos, J. A., Tardós, J. D., and Schmidt, G. (1997). “Building a global map of the environment of a mobile robot: The importance of correlations.” *International Conference on Robotics and Automation*, 1053–1059. Albuquerque, New Mexico
- Chandrashekar, G., and Sahin, F. (2014). “A survey on feature selection methods.” *Computers and Electrical Engineering*, 40 (1): 16–28. Pergamon. DOI: 10.1016/j.compeleceng.2013.11.024
- Chen, X., Golovinskiy, A., and Funkhouser, T. (2009). “A benchmark for 3D mesh segmentation.” *Acm transactions on graphics (tog)*, 73. ACM
- Chow, J. K., Liu, K., Tan, P. S., Su, Z. S., Wu, J., Li, Z., and Wang, Y.-H. (2021). “Automated defect inspection of concrete structure.” *Automation in Construction*, 132: 103959. DOI: 10.1016/j.autcon.2021.103959
- Compute Canada Org. (2021). “Compute Canada.” <https://www.computecanada.ca/>

- Computers and Structures Inc. (2017). “CsiBridge.” Accessed January 20, 2007. <https://www.csiamerica.com/products/csibridge>
- Cook, D., Vardy, A., and Lewis, R. (2014). “A survey of AUV and robot simulators for multi-vehicle operations.” *Autonomous Underwater Vehicles Journal*, 1–8. DOI: 10.1109/AUV.2014.7054411
- Crosby, C. (2016). “Introduction to LiDAR, Terrestrial Laser Scanning Applications”
- Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. (2019). “Class-Balanced Loss Based on Effective Number of Samples.” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9268–9277. Xi’an, China
- Dawood, T., Zhu, Z., and Zayed, T. (2017). “Machine vision-based model for spalling detection and quantification in subway networks.” *Automation in Construction*, 81: 149–160. Elsevier. DOI: 10.1016/j.autcon.2017.06.008
- Demir, N., and Baltasvias, E. (2012). “Automated modeling of 3D building roofs using image and LiDAR data.” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35–40. Melbourne, Australia
- Dijkstra, E. W. (1959). “A note on two problems in connexion with graphs.” *Numerische mathematik*, 1 (1): 269–271. Springer
- Dong, Y. (2015). “What’s the difference between RRT and RRT\* and which one should be used.” [https://www.youtube.com/watch?v=JeEk\\_CWcRFI](https://www.youtube.com/watch?v=JeEk_CWcRFI)
- Dorafshan, S., and Maguire, M. (2018). “Bridge inspection: human performance, unmanned aerial systems and automation.” *Journal of Civil Structural Health Monitoring*, 8 (3): 443–476. Springer
- Dornhege, C., Kleiner, A., Hertle, A., and Kolling, A. (2016). “Multirobot Coverage Search in Three Dimensions.” *Journal of field Robotics*, 33 (4): 537–558
- Dronocode Project Inc. (2017). “A linux foundation collaborative project.” <https://www.dronocode.org>
- Eberhart, R., and Kennedy, J. (1995). “A new optimizer using particle swarm theory.” *Sixth International Symposium on Micro Machine and Human Science*, 39–43. Nagoya, Japan: IEEE
- Englot, B., and Hover, F. (2010). “Inspection planning for sensor coverage of 3D marine structures.” *International Conference on Intelligent Robots and Systems (IROS)*, 4412–4417. Taipei, Taiwan
- Englot, B., and Hover, F. S. (2012). “Sampling-based coverage path planning for inspection of complex structures.” *22nd International Conference on Automated Planning and Scheduling*, 29–37. Atibaia, Brazil
- Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). “The pascal visual object classes challenge: A retrospective.” *International journal of computer vision*, 111 (1): 98–136. Springer
- Facelift Ltd. (2017). “Bronto S46 XDT.” <http://www.facelift.co.uk/piclibrary/picture.cfm?pic=331>
- FARO Technologies Inc. (2011). “User Manual FARO Laser Scanner Focus 3D”
- FARO Technologies Inc. (2012). *Faro Laser Scanner Focus 3D X 130 -NEO-Tech*
- FARO Technologies Inc. (2017). “Quality Setting Function on the Focus3D.” Accessed May 24, 2021. [https://knowledge.faro.com/Hardware/3D\\_Scanners/Focus/Quality\\_Setting\\_Function\\_on\\_the\\_Focus3D](https://knowledge.faro.com/Hardware/3D_Scanners/Focus/Quality_Setting_Function_on_the_Focus3D)



- Freimuth, H., and König, M. (2018). “Planning and executing construction inspections with unmanned aerial vehicles.” *Automation in Construction*, 96: 540–553. DOI: 10.1016/j.autcon.2018.10.016
- Fu, B., Chen, L., Zhou, Y., Zheng, D., Wei, Z., Dai, J., and Pan, H. (2018). “An improved A\* algorithm for the industrial robot path planning with high success rate and short length.” *Robotics Autonomous System*, 106: 26–37
- Fu, H., Meng, D., Li, W., and Wang, Y. (2021). “Bridge Crack Semantic Segmentation Based on Improved Deeplabv3+.” *Journal of Marine Science and Engineering*, 9 (6): 671
- Gagnon, M., Gaudreault, V., and Overton, D. (2008). *Age of public infrastructure: A provincial perspective*. Statistics Canada
- Galceran, E., and Carreras, M. (2013). “A survey on coverage path planning for robotics.” *Robotics and Autonomous Systems*, 61 (12): 1258–1276. DOI: 10.1016/j.robot.2013.09.004
- Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., and Garcia-Rodriguez, J. (2017). “A review on deep learning techniques applied to semantic segmentation.” *Computer Vision and Pattern Recognition*, arXiv, 1: 1704.06857. DOI: 10.48550/arXiv.1704.06857
- Getbestcopter. (2016). “How to build a quadcopter with Arduino?” <http://www.getbestcopter.com/how-to-build-a-quadcopter-with-arduino/>
- Gharib, A., Benhra, J., and Chaouqi, M. (2015). “A performance comparison of PSO and GA applied to TSP.” *International Journal of Computer Applications*, 130 (15): 34–39. Foundation of Computer Science
- Girardeau-Montaut, D. (2020). “CloudCompare”
- Goldbarg, E. F. G., de Souza, G. R., and Goldbarg, M. C. (2006). “Particle Swarm Optimization for the bi-objective degree constrained minimum spanning tree.” *Congress on Evolutionary Computation (CEC)*, 420–427. Vancouver, BC, Canada: IEEE
- González-Jorge, H., Gonzalez-Aguilera, D., Rodriguez-Gonzalvez, P., and Arias, P. (2012). “Monitoring biological crusts in civil engineering structures using intensity data from terrestrial laser scanners.” *Construction and Building Materials*, 31: 119–128. Elsevier. DOI: 10.1016/j.conbuildmat.2011.12.053
- Government of Canada. (2021). “Part D - Bridge Inspection.” *Guideline for Bridge Safety Management*. Accessed May 24, 2021. [https://tc.canada.ca/en/rail-transportation/guidelines/guideline-bridge-safety-management/part-bridge-inspection#\\_3.3.1\\_-\\_Visual](https://tc.canada.ca/en/rail-transportation/guidelines/guideline-bridge-safety-management/part-bridge-inspection#_3.3.1_-_Visual)
- Grefenstette, J., Gopal, R., Rosmaita, B., and Van Gucht, D. (1985). “Genetic algorithms for the traveling salesman problem.” *the first International Conference on Genetic Algorithms and their Applications*, 160–168. Lawrence Erlbaum
- Grilli, E., Menna, F., and Remondino, F. (2017). “A review of point clouds segmentation and classification algorithms.” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 339–345. Nafplio, Greece
- Guan, H., Li, J., Yu, Y., Chapman, M., and Wang, C. (2015). “Automated Road Information Extraction From Mobile Laser Scanning Data.” *IEEE Transactions on Intelligent Transportation Systems*, 16 (1): 194–205. DOI: 10.1109/TITS.2014.2328589
- Guerrero, J. A., and Bestaoui, Y. (2013). “UAV path planning for structure inspection in windy environments.” *Journal of Intelligent & Robotic Systems*, 69 (1): 297–311. DOI: 10.1007/s10846-012-9778-2

- Guldur, B., and Hajjar, J. F. (2016). “Automated classification of detected surface damage from point clouds with supervised learning.” *34th International Symposium on Automation and Robotics in Construction (ISARC)*, 1–7. Taipei, Taiwan: Vilnius Gediminas Technical University,
- Guldur, B., and Hajjar, J. F. (2017). “Laser-based surface damage detection and quantification using predicted surface properties.” *Automation in Construction*, 83: 285–302. Elsevier. DOI: 10.1016/j.autcon.2017.08.004
- Guldur, B., Yan, Y., and Hajjar, J. F. (2015). “Condition Assessment of Bridges Using Terrestrial Laser Scanners.” *Structures Congress*, 355–366
- Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M. (2020). “Deep learning for 3D point clouds: A survey.” *IEEE transactions on pattern analysis and machine intelligence*, 1–27. DOI: 10.1109/tpami.2020.3005434
- Habel, R. (2017). “Laser Road Imaging System.” <http://www.pavemetrics.com>
- Hackel, T., Wegner, J. D., and Schindler, K. (2016). “Contour detection in unstructured 3D point clouds.” *the IEEE Conference on Computer Vision and Pattern Recognition*, 1610–1618. Las Vegas, NV, USA
- Hamledari, H., Sajedi, S., McCabe, B., and Fischer, M. (2021). “Automation of Inspection Mission Planning Using 4D BIMs and in Support of Unmanned Aerial Vehicle–Based Data Collection.” *Journal of Construction Engineering and Management*, 147 (3): 04020179. American Society of Civil Engineers. DOI: 10.1061/(ASCE)CO.1943-7862.0001995
- Hampel, U., and Maas, H.-G. (2009). “Cascaded image analysis for dynamic crack detection in material testing.” *Journal of Photogrammetry and Remote Sensing*, 64 (4): 345–350. Elsevier. DOI: 10.1016/j.isprsjprs.2008.12.006
- Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., and Malik, J. (2011). “Semantic contours from inverse detectors.” *IEEE International Conference on Computer Vision*, 991–998. Barcelona, Spain
- Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). “A formal basis for the heuristic determination of minimum cost paths.” *Transactions on Systems Science and Cybernetics*, 4 (2): 100–107. IEEE. DOI: 10.1109/TSSC.1968.300136
- Haurum, J. B., Allahham, M. M., Lynge, M. S., Henriksen, K. S., Nikolov, I. A., and Moeslund, T. B. (2021). “Sewer Defect Classification using Synthetic Point Clouds.” *16th International Conference on Computer Vision Theory and Applications*, 891–900. Vienna, Austria
- Helsgaun, K. (2000). “An effective implementation of the Lin–Kernighan traveling salesman heuristic.” *European Journal of Operational Research*, 126 (1): 106–130. DOI: 10.1016/S0377-2217(99)00284-2
- Hoffman, K. L., Padberg, M., and Rinaldi, G. (2013). “Traveling Salesman Problem.” *Encyclopedia of operations research and management science*, 1573–1578. Springer. DOI: 10.1007/978-1-4419-1153-7\_1068
- Hoskere, V., Narazaki, Y., Hoang, T. A., and Spencer Jr., B. F. (2020). “MaDnet: multi-task semantic segmentation of multiple types of structural materials and damage in images of civil infrastructure.” *Journal of Civil Structural Health Monitoring*, 10: 757–773
- Hou, T.-C., Liu, J.-W., and Liu, Y.-W. (2017). “Algorithmic clustering of LiDAR point cloud data for textural damage identifications of structural elements.” *Measurement*, 108: 77–90. DOI: <https://doi.org/10.1016/j.measurement.2017.05.032>

- Hover, F. S., Eustice, R. M., Kim, A., Englot, B., Johannsson, H., Kaess, M., and Leonard, J. J. (2012). "Advanced perception, navigation and planning for autonomous in-water ship hull inspection." *The International Journal of Robotics Research*, 31 (12): 1445–1464. Sage Publications. DOI: 10.1109/IROS.2010.5648908
- Hsien-Ke, L., Jallow, M., Nie-Jia, Y., Ming-Yi, J., Jyun-Hao, H., Cheng-Wei, S., and Po-Yuan, C. (2017). "Comparison of Bridge Inspection Methodologies and Evaluation Criteria in Taiwan and Foreign Practices." *the International Symposium on Automation and Robotics in Construction*, 1–8. Taiwan: IAARC
- Huang, H., Zhang, C., and Hammad, A. (2021). "Effective Scanning Range Estimation for Using TLS in Construction Projects." *Journal of Construction Engineering and Management*, 147 (9): 04021106. American Society of Civil Engineers. DOI: 10.1061/(ASCE)CO.1943-7862.0002127
- Hussain, A., Shad Muhammad, Yousaf Nauman Sajid, M., Hussain, I., Mohamd Shoukry, A., and Gani, S. (2017). "Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator." *Computational intelligence and neuroscience*, 2017: 1–8. DOI: 10.1155/2017/7430125
- Hutchinson, T. C., and Chen, Z. (2006). "Improved Image Analysis for Evaluating Concrete Damage." *Journal of Computing in Civil Engineering*, 20 (3): 210–216. American Society of Civil Engineers. DOI: 10.1061/ASCE0887-3801200620:3210
- Italdron Inc. (2019). "Bigone 8 Hsepro LiDAR." <http://www.italdron.com/professionals-drones-and-accessories/professionals-drones/bigone-8hse-pro-laser-scan>
- Janoušek, P., and Faigl, J. (2013). "Speeding up coverage queries in 3D multi-goal path planning." *International Conference on Robotics and Automation (ICRA)*, 5082–5087. Karlsruhe, Germany
- Jiang, H., Yan, F., Cai, J., Zheng, J., and Xiao, J. (2020). "End-to-End 3D Point Cloud Instance Segmentation Without Detection." 12796–12805
- Kachitvichyanukul, V. (2012). "Comparison of three evolutionary algorithms: GA, PSO, and DE." *Industrial Engineering and Management Systems*, 11 (3): 215–223. Korean Institute of Industrial Engineers
- Kang, S., Kim, S.-S., Won, J., and Kang, Y.-M. (2016). "GPU-based parallel genetic approach to large-scale travelling salesman problem." *The Journal of Supercomputing*, 72 (11): 4399–4414. DOI: doi.org/10.1007/s11227-016-1748-1
- Khallaf, R., and Khallaf, M. (2021). "Classification and analysis of deep learning applications in construction: A systematic literature review." *Automation in construction*, 129: 1–16. DOI: 10.1016/j.autcon.2021.103760
- Khaloo, A., Lattanzi, D., Cunningham, K., Dell'Andrea, R., and Riley, M. (2018). "Unmanned aerial vehicle inspection of the Placer River Trail Bridge through image-based 3D modelling." *Structure and Infrastructure Engineering*, 14 (1): 124–136. Taylor & Francis. DOI: 10.1080/15732479.2017.1330891
- Kim, H., and Kim, C. (2020). "Deep-Learning-Based Classification of Point Clouds for Bridge Inspection." *Remote Sensing*, 12 (22): 3757. DOI: doi.org/10.3390/rs12223757
- Kim, H., Yoon, J., and Sim, S.-H. (2020). "Automated bridge component recognition from point clouds using deep learning." *Structural Health Monitoring*, 27 (9): e2591. DOI: doi.org/10.1002/stc.2591

- Kim, I.-H., Jeon, H., Baek, S.-C., Hong, W.-H., and Jung, H.-J. (2018). “Application of Crack Identification Techniques for an Aging Concrete Bridge Inspection Using an Unmanned Aerial Vehicle.” *Sensors*, 18 (6): 1881. Multidisciplinary Digital Publishing Institute
- Kim, J. W., Kim, S. B., Park, J. C., and Nam, J. W. (2015a). “Development of crack detection system with unmanned aerial vehicles and digital image processing.” *Advances in structural engineering and mechanics*, 33 (3): 25–29
- Kim, M. K., Sohn, H., and Chang, C.-C. (2015b). “Localization and Quantification of Concrete Spalling Defects Using Terrestrial Laser Scanning.” *Journal of Computing in Civil Engineering*, 29 (6): 1–12. DOI: 10.1061/(ASCE)CP.1943-5487.0000415
- Koch, C., and Brilakis, I. (2011). “Pothole detection in asphalt pavement images.” *Advanced Engineering Informatics*, 25 (3): 507–515. Elsevier. DOI: 10.1016/j.aei.2011.01.002
- Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., and Fieguth, P. (2015). “A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure.” *Advanced Engineering Informatics*, 29: 196–210. DOI: 10.1016/j.aei.2015.01.008
- Koo, B., Jung, R., Yu, Y., and Kim, I. (2021). “A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models.” *Journal of Computational Design and Engineering*, 8 (1): 239–250
- Kroll, A., Baetz, W., and Peretzki, D. (2009). “On autonomous detection of pressured air and gas leaks using passive IR-thermography for mobile robot application.” *International Conference on Robotics and Automation (CRA)*, 921–926. Kobe, Japan: IEEE
- Kuffner, J. J., and LaValle, S. M. (2000). “RRT-connect: An efficient approach to single-query path planning.” *International Conference on Robotics and Automation (CRA)*, 995–1001. San Francisco, USA
- Kulling, K. C. (2009). “Optimal and receding-horizon path planning algorithms for communications relay vehicles in complex environments.” Massachusetts Institute of Technology
- Laefer, D. F., Truong-Hong, L., Carr, H., and Singh, M. (2014). “Crack detection limits in unit based masonry with terrestrial laser scanning.” *NDT and E International*, 62: 66–76. DOI: 10.1016/j.ndteint.2013.11.001
- Langari, S. M. (2015). “Enhanced Path Planning Method for Improving Safety and Productivity of Excavation Operations.”, Master Thesis, Concordia University
- Lattanzi, D., and Miller, G. R. (2015). “3D scene reconstruction for robotic bridge inspection.” *Journal of Infrastructure Systems*, 21 (2): 04014041. ascelibrary.org. DOI: 10.1061/(ASCE)IS.1943-555X.0000229
- LaValle, S. M., and Kuffner Jr, J. J. (2001). “Randomized kinodynamic planning.” *The International Journal of Robotics Research*, 20 (5): 378–400. SAGE Publications. DOI: 10.1177/02783640122067453
- Le, T., Gibb, S., Pham, N., La, H. M., Falk, L., and Berendsen, T. (2017). “Autonomous robotic system using non-destructive evaluation methods for bridge deck inspection.” *International Conference on Robotics and Automation (ICRA)*, 3672–3677. Singapore
- Le, T.-T., Nguyen, V.-H., and Le, M. V. (2021). “Development of Deep Learning Model for the Recognition of Cracks on Concrete Surfaces.” *Applied Computational Intelligence and Soft Computing*, e8858545. Hindawi. DOI: 10.1155/2021/8858545

- Lee, D., Kim, J., and Lee, D. (2019). “Robust concrete crack detection using deep learning-based semantic segmentation.” *International Journal of Aeronautical and Space Sciences*, 20 (1): 287–299
- Lehtomäki, M., Jaakkola, A., Hyyppä, J., Lampinen, J., Kaartinen, H., Kukko, A., Puttonen, E., and Hyyppä, H. (2016). “Object Classification and Recognition From Mobile Laser Scanning Point Clouds in a Road Environment.” *IEEE Transactions on Geoscience and Remote Sensing*, 54 (2): 1226–1239. DOI: 10.1109/TGRS.2015.2476502
- Li, C.-L., Zaheer, M., Zhang, Y., Poczos, B., and Salakhutdinov, R. (2018). “Point Cloud GAN.” *arXiv:1810.05795 [cs, stat]*
- Li, M., Hu, Y., Zhao, N., and Qian, Q. (2019). “Modeling and analyzing point cloud generation in missile-borne LiDAR.” *Defence Technology*, 16 (1): 69–76. DOI: doi.org/10.1016/j.dt.2019.10.003
- Li, Z., Yan, Y., Jing, Y., and Zhao, S. G. (2015). “The design and testing of a LiDAR platform for a UAV for heritage mapping.” *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40 (1): 17. DOI: 10.5194/isprsarchives-XL-1-W4-17-2015
- Lin, G., and Shen, W. (2018). “Research on convolutional neural network based on improved Relu piecewise activation function.” *Procedia Computer Science*, Recent Advancement in Information and Communication Technology:, 131: 977–984. DOI: 10.1016/j.procs.2018.04.239
- Liu, P., Chen, A., Huang, Y.-N., Han, J.-Y., Lai, J.-S., Kang, S.-C., Wu, T.-H., Wen, M.-C., and Tsai, M.-H. (2014). “A review of rotorcraft Unmanned Aerial Vehicle (UAV) developments and applications in civil engineering.” *Smart Structure and Systems*, 13 (6): 1065–1094. DOI: 10.12989/sss.2014.13.6.1065
- Liu, W., Chen, S., Boyajian, D., and Hauser, E. (2010). “Application of 3D LiDAR scan of a bridge under static load testing.” *Materials Evaluation*, 68 (12): 1359–1367
- Liu, W., Chen, S., and Hauser, E. (2011). “LiDAR-based bridge structure defect detection.” *Experimental Techniques*, 35 (6): 27–34. Wiley Online Library. DOI: 10.1111/j.1747-1567.2010.00644.x
- Lopez Droguett, E., Tapia, J., Yanez, C., and Boroschek, R. (2020). “Semantic segmentation model for crack images from concrete bridges for mobile devices.” *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 1–14. DOI: https://doi.org/10.1177/1748006X20965111
- Luckai, J., Polak, M. A., and Walbridge, S. (2014). “A methodology for evaluating the effects of spalling on the structural capacity of reinforced concrete bridge.” *Canadian Journal of Civil Engineering*, 41 (3): 197–205. DOI: 10.1139/cjce-2011-0263
- Lumelsky, V. J., and Stepanov, A. A. (1987). “Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape.” *Algorithmica*, 2 (1): 403–430. Springer. DOI: 10.1007/BF01840369
- Luo, X., Li, X., Yang, Q., Wu, F., Zhang, D., Yan, W., and Xi, Z. (2017). “Optimal path planning for UAV-based inspection system of large-scale photovoltaic farm.” *Chinese Automation Congress (CAC)*, 4495–4500. Jinan, China: IEEE
- Maguire, M., Dorafshan, S., and Thomas, R. (2018). “SDNET2018: A concrete crack image dataset for machine learning applications.” *Digital Commons*. https://digitalcommons.usu.edu/all\_datasets/48/

- Martin-Abadal, M., Piñar-Molina, M., Martorell-Torres, A., Oliver-Codina, G., and Gonzalez-Cid, Y. (2021). "Underwater Pipe and Valve 3D Recognition Using Deep Learning Segmentation." *Journal of Marine Science and Engineering*, 9 (1): 5
- Maru, M., Bekele, D., Lee, K., Demissie, T., and Seunghee, P. (2021). "Comparison of Depth Camera and Terrestrial Laser Scanner in Monitoring Structural Deflections." *Sensors*, 21 (1): 201. DOI: <https://doi.org/10.3390/s21010201>
- Marzouk, M., and Ali, H. (2013). "Modeling safety considerations and space limitations in piling operations using agent based simulation." *Expert Systems with Applications*, 40 (12): 4848–4857. Elsevier. DOI: 10.1016/j.eswa.2013.02.021
- Maturana, D., and Scherer, S. (2015). "Voxnet: A 3D convolutional neural network for real-time object recognition." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928. Hamburg, Germany: IEEE
- Mcful, H. (2018). "Drones are Breaking Ground in Bridge Inspection"
- McLaughlin, E., Charron, N., and Narasimhan, S. (2020). "Automated Defect Quantification in Concrete Bridges Using Robotics and Deep Learning." *Journal of Computing in Civil Engineering*, 34 (5): 04020029
- Metni, N., and Hamel, T. (2007). "A UAV for bridge inspection: Visual servoing control law with orientation limits." *Automation in Construction*, 17 (1): 3–10. Elsevier. DOI: 10.1016/j.autcon.2006.12.010
- Michalski, R. S., Carbonell, J. G., and Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer-Verlag Berlin Heidelberg
- Mirza, S. (2006). "Durability and sustainability of infrastructure — a state-of-the-art report." *Canadian Journal of Civil Engineering*, 33 (6): 639–649. DOI: 10.1139/106-049
- Mizoguchi, T., Koda, Y., Iwaki, I., Wakabayashi, H., Kobayashi, Y., Shirai, K., Hara, Y., and Lee, H. S. (2013). "Quantitative scaling evaluation of concrete structures based on terrestrial laser scanning." *Automation in Construction*, 35: 263–274. Elsevier B.V. DOI: 10.1016/j.autcon.2013.05.022
- Mohammadi, M. E., Wood, R. L., and Wittich, C. E. (2019). "Non-Temporal Point Cloud Analysis for Surface Damage in Civil Structures." *ISPRS International Journal of Geo-Information*, 8 (12): 527. DOI: <https://doi.org/10.3390/ijgi8120527>
- Mohammed Abdelkader, E., Moselhi, O., Marzouk, M., and Zayed, T. (2021). "Entropy-Based Automated Method for Detection and Assessment of Spalling Severities in Reinforced Concrete Bridges." *Journal of Performance of Constructed Facilities*, 35 (1): 04020132
- Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). "On the number of linear regions of deep neural networks." *Advances in Neural Information Processing Systems*, 27: 1–9
- Morgan, D., and Falkner, E. (2001). *Aerial mapping: methods and applications*. Florida, USA: CRC Press
- Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., and Yuille, A. (2014). "The role of context for object detection and semantic segmentation in the wild." *IEEE Conference on Computer Vision and Pattern Recognition*, 891–898. Columbus, OH, USA
- Nasir, J., Islam, F., Malik, U., Ayaz, Y., Hasan, O., Khan, M., and Muhammad, M. S. (2013). "RRT\*-SMART: A rapid convergence implementation of RRT." *International Journal of Advanced Robotic Systems*, 10 (7): 299–311. SAGE Publications Sage UK: London, England. DOI: 10.5772/56718

- Nasrollahi, M. (2019). “Automated Bridge Inspection for Concrete Surface Defect Detection Using Deep Neural Network Based on LiDAR Scanning.” Master Thesis, Concordia University
- Nasrollahi, M., Bolourian, N., and Hammad, A. (2019). “Concrete Surface Defect Detection Using Deep Neural Network Based on LiDAR Scanning.” *CSCE Conference*, 1–10. Laval
- Nasrollahi, M., Bolourian, N., Zhu, Z., and Hammad, A. (2018). “Designing LiDAR-equipped UAV platform for structural inspection.” *35th International Symposium on Automation and Robotics in Construction (ISARC)*, 1–6. Berlin, Germany: IEEE
- Noreen, I., Khan, A., and Habib, Z. (2016). “Optimal Path Planning using RRT\* based Approaches: A Survey and Future Directions.” *International Journal of Advanced Computer Science and Applications*, 7 (11): 97–107. DOI: 10.14569/IJACSA.2016.071114
- NTSB. (2008). *Collapse of I-35W Highway Bridge Minneapolis, Minnesota August 1, 2007. National Transportation Safety Board*, 1–178. Washington DC
- Oh, J. S., Choi, Y. H., Park, J. B., and Zheng, Y. F. (2004). “Complete coverage navigation of cleaning robots using triangular-cell-based map.” *IEEE Transactions on Industrial Electronics*, 51 (3): 718–726. IEEE. DOI: 10.1109/TIE.2004.825197
- Olsen, M. J., Kuester, F., Chang, B. J., and Hutchinson, T. C. (2010). “Terrestrial Laser Scanning-Based Structural Damage Assessment.” *Journal of Computing in Civil Engineering*, 24 (3): 264–272. DOI: 10.1061/(asce)cp.1943-5487.0000028
- Ongsulee, P. (2018). “Artificial intelligence, machine learning and deep learning.” *15th International Conference on ICT and Knowledge Engineering (ICT&KE)*, 1–6. Bangkok, Thailand: IEEE
- Ontario Ministry of Transportation. (2008). *Ontario Structure Inspection Manual (OSIM)*. St. Catharines, Ontario: Policy, Planning & Standards Division Engineering Standards Branch Bridge Office
- Papachristos, C., Khattak, S., Mascari, F., and Alexis, K. (2019). “Autonomous navigation and mapping in underground mines using aerial robots.” *Aerospace Conference*, 1–8. Montana, USA: IEEE
- Pauly, M., Gross, M., and Kobbelt, L. P. (2002). “Efficient simplification of point-sampled surfaces.” *conference on Visualization (VIS)*, 163–170. Boston, MA, USA: IEEE Computer Society
- Perron, L., and Furnon, V. (2020). “Or-tools.” *Google*. <https://developers.google.com/optimization/>
- Pfeifer, N., and Briese, C. (2007). “Laser scanning—principles and applications.” *GeoSiberia-International Exhibition and Scientific Congress*, cp-59. Novosibirsk, Russia: European Association of Geoscientists & Engineers
- Phung, M. D., Hoang, V. T., Dinh, T. H., and Ha, Q. (2017a). “Automatic crack detection in built infrastructure using unmanned aerial vehicles.” *The 34th International Symposium on Automation and Robotics in Construction*, 823–829. Taipei, Taiwan
- Phung, M. D., Quach, C. H., Dinh, T. H., and Ha, Q. (2017b). “Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection.” *Automation in Construction*, 81: 25–33. DOI: 10.1016/j.autcon.2017.04.013
- Pierdicca, R., Paolanti, M., Martini, F., Massimo, M., Morbidoni, C., Malinverni, E. S., Frontoni, E., and Lingua, A. M. (2020). “Point cloud semantic segmentation using a deep learning framework for cultural heritage.” *Remote Sensing*, 12 (6): 1005

- Prest, A., Leistner, C., Civera, J., Schmid, C., and Ferrari, V. (2012). “Learning object class detectors from weakly annotated video.” *Computer Vision and Pattern Recognition (CVPR)*, 3282–3289. Providence, RI, USA: IEEE
- Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). “Pointnet: Deep learning on point sets for 3d classification and segmentation.” *Computer Vision and Pattern Recognition (CVPR)*, 4–23. Honolulu, HI, USA: IEEE
- Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.” *Advances in Neural Information Processing Systems*, 1–14. Neural information processing systems foundation
- Quadros, A., Underwood, J. P., and Douillard, B. (2012). “An occlusion-aware feature for range images.” *International Conference on Robotics and Automation (ICRA)*, 4428–4435. St. Paul, MN, USA: IEEE
- Quin, P., Paul, G., Alempijevic, A., Liu, D., and Dissanayake, G. (2013). “Efficient neighborhood-based information gain approach for exploration of complex 3D environments.” *International Conference on Robotics and Automation (ICRA)*, 1343–1348. Karlsruhe, Germany
- Rabbani, T., Van Den Heuvel, F., and Vosselmann, G. (2006). “Segmentation of point clouds using smoothness constraint.” *International archives of photogrammetry, remote sensing and spatial information sciences*, 36 (5): 248–253. DOI: 10.1.1.118.5346
- Rafanavicius, V., Cimmperman, P., Taluntis, V., Man, K. L., Volkvicius, G., Jurkynas, M., and Bezaras, J. (2017). “Efficient path planning methods for UAVs inspecting power lines.” *International Conference on Platform Technology and Service (PlatCon)*, 1–6. Busan, Korea
- Rahman, M. A., and Wang, Y. (2016). “Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation.” *Advances in Visual Computing*, 234–244. Cham: Springer
- Rashidi, M., Mohammadi, M., Sadeghlou, S., Abdolvand, M., Truong-Hong, L., and Samali, B. (2020). “A Decade of Modern Bridge Monitoring Using Terrestrial Laser Scanning: Review and Future Directions.” *Remote Sensing*, 12 (22): 3796
- Reshetyuk, Y. (2006). “Investigation and calibration of pulsed time-of-flight terrestrial laser scanners.” Ph.D. Stockholm, Sweden: Royal Institute of Technology (KTH), Department of Transport and Economics
- Scott, W. R. (2009). “Model-based view planning.” *Machine Vision and Applications*, 20 (1): 47–69. Springer. DOI: 10.1007/s00138-007-0110-2
- Scott, W. R., Roth, G., and Rivest, J.-F. (2003). “View planning for automated three-dimensional object reconstruction and inspection.” *Computing Surveys (CSUR)*, 35 (1): 64–96. Association for Computing Machinery (ACM). DOI: 10.1145/641865.641868
- Sharma, S., Sharma, S., and Athaiya, A. (2017). “Activation functions in neural networks.” *Towards data science*, 6 (12): 310–316
- Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). “Indoor segmentation and support inference from rgb-d images.” *European Conference on Computer Vision*, 746–760. Florence, Italy: Springer
- Silverman, R. (2002). *Modern calculus and analytic geometry*. Dover Publications
- Sun, M., Song, Z., Jiang, X., Pan, J., and Pang, Y. (2017). “Learning Pooling for Convolutional Neural Network.” *Neurocomputing*, 224: 96–104. DOI: 10.1016/j.neucom.2016.10.049



- Synergy Positioning Systems Ltd. (2018). “Faro in the air, underground, everywhere 3D scanning.” Accessed July 6, 2020. <https://synergypositioning.co.nz/news/faro-in-the-air-underground-everywhere-3D-scanning>
- Tam, G. K. L., Cheng, Z. Q., Lai, Y. K., Langbein, F. C., Liu, Y., Marshall, D., Martin, R. R., Sun, X. F., and Rosin, P. L. (2013). “Registration of 3d point clouds and meshes: A survey from rigid to Nonrigid.” *IEEE Transactions on Visualization and Computer Graphics*, 19 (7): 1199–1217. DOI: 10.1109/TVCG.2012.310
- Taylor, K., and LaValle, S. M. (2009). “I-Bug: An intensity-based bug algorithm.” *IEEE International Conference on Robotics and Automation*, 3981–3986. Kobe, Japan
- Te, G., Hu, W., Zheng, A., and Guo, Z. (2018). “Rgcnn: Regularized graph CNN for point cloud segmentation.” *the 26th ACM international conference on Multimedia*, 746–754. Seoul, Republic of Korea
- Teza, G., Galgaro, A., and Moro, F. (2009). “Contactless recognition of concrete surface damage from laser scanning and curvature computation.” *Non-Destructive Testing and Evaluation International*, 42 (4): 240–249. Elsevier. DOI: 10.1016/j.ndteint.2008.10.009
- The HDF Group. (2021). “Ensuring long-term access and usability of HDF data and supporting users of HDF technologies.” Accessed May 24, 2021. <https://www.hdfgroup.org/>
- The Vertikal Press. (2015). “Fatal Underbridge Inspection.” Accessed July 5, 2020. <https://www.craneaccidents.com/2015/08/report/fatal-underbridge-inspection/>
- Thrun, S. (1998). “Learning metric-topological maps for indoor mobile robot navigation.” *Artificial Intelligence*, 99 (1): 21–71. Elsevier. DOI: 10.1016/S0004-3702(97)00078-7
- Transports Canada. (2012). “Guideline for bridge safety management”
- Trimble Inc. (2020). “Trimble Software.” <https://go2.trimble.com/downloadTBC.html>
- Truong-Hong, L., Falter, H., Lennon, D., and Laefer, D. F. (2016). “Framework for bridge inspection with laser scanning.” *EASEC-14 Structural Engineering and Construction*, 1–8. Ho Chi Minh City, Vietnam
- Turkan, Y., Hong, J., Laflamme, S., and Puri, N. (2018). “Adaptive wavelet neural network for terrestrial laser scanner-based crack detection.” *Automation in Construction*, 94: 191–202. DOI: <https://doi.org/10.1016/j.autcon.2018.06.017>
- Unity Technologies. (2018). “Unity 3D Game Engine.” <https://unity3d.com/unity>
- Valença, J., Puente, I., Júlio, E., González-Jorge, H., and Arias-Sánchez, P. (2017). “Assessment of cracks on concrete bridges using image processing supported by laser scanning survey.” *Construction and Building Materials*, 146: 668–678. DOI: 10.1016/j.conbuildmat.2017.04.096
- Velodyne LiDAR Inc. (2015). “Velodyne LiDAR PUCK™.” Accessed July 31, 2019. <https://www.amtechs.co.jp/product/VLP-16-Puck.pdf>
- Vignesh, R., Narenthiran, B., Manivannan, S., Arul Murugan, R., and RajKumar, V. (2021). “Concrete Bridge Crack Detection Using Convolutional Neural Network.” *Materials, Design, and Manufacturing for Sustainable Environment*, 797–812. Springer
- Walsh, S. B., Borello, D. J., Guldur, B., and Hajjar, J. F. (2013). “Data processing of point clouds for object detection for structural engineering applications.” *Computer-Aided Civil and Infrastructure Engineering*, 28 (7): 495–508. Wiley Online Library. DOI: 10.1111/mice.12016
- Wang, J., Liu, Y., Nie, X., and Mo, Y. L. (2022). “Deep convolutional neural networks for semantic segmentation of cracks.” *Structural Control and Health Monitoring*, 29 (1): e2850

- Wang, Z., and Lu, F. (2016). “VoxSegNet: Volumetric CNNs for semantic part segmentation of 3D shapes.” *Transactions on Visualization and Computer Graphics*, 26 (9): 2919–2930
- Weber, C., Hahmann, S., and Hagen, H. (2010). “Sharp feature detection in point clouds.” *Shape Modeling International Conference (SMI)*, 175–186. Beijing, China: IEEE
- Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). “3d shapenets: A deep representation for volumetric shapes.” *IEEE conference on computer vision and pattern recognition*, 1912–1920. Boston, USA
- Xia, T., Yang, J., and Chen, L. (2022). “Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning.” *Automation in Construction*, 133: 103992. DOI: doi.org/10.1016/j.autcon.2021.103992
- Xiao, J., Owens, A., and Torralba, A. (2013). “Sun3d: A database of big spaces reconstructed using sfm and object labels.” *IEEE International Conference on Computer Vision (ICCV)*, 1625–1632. Sydney, Australia
- Xu, X., Yang, H., and Neumann, I. (2015). “Concrete Crack Measurement and Analysis Based on Terrestrial Laser Scanning Technology.” *Sensors and Transducers Journal*, 186 (3): 5
- Xu, Y., and Turkan, Y. (2019). “Bridge Inspection Using Bridge Information Modeling (BrIM) and Unmanned Aerial System (UAS).” *Advances in Informatics and Computing in Civil and Construction Engineering*, 617–624. Springer International Publishing
- Xu, Y., and Turkan, Y. (2022). “Risk Assessment for Using UAS in Construction: A Fuzzy Analytical Hierarchy Process.” *Construction Research Congress*, 441–451. Arlington, USA: American Society of Civil Engineers
- Yang, C.-H., Tsai, M.-H., Kang, S.-C., and Hung, C.-Y. (2018). “UAV path planning method for digital terrain model reconstruction—A debris fan example.” *Automation in Construction*, 93: 214–230. Elsevier. DOI: 10.1016/j.autcon.2018.05.024
- Yavartanoo, M., Kim, E. Y., and Lee, K. M. (2018). “Spnet: Deep 3d object classification and retrieval using stereographic projection.” *Asian Conference on Computer Vision*, 691–706. Berlin, German: Springer
- Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., and Guibas, L. (2016). “A scalable active framework for region annotation in 3D shape collections.” *ACM Transactions on Graphics*, 35 (6): 210:1-210:12. DOI: 10.1145/2980179.2980238
- Yin, C., Wang, B., Gan, V. J., Wang, M., and Cheng, J. C. (2021). “Automated semantic segmentation of industrial point clouds using ResPointNet++.” *Automation in Construction*, 130: 103874
- Yoder, L., and Scherer, S. (2016). “Autonomous exploration for infrastructure modeling with a micro aerial vehicle.” *Field and service robotics*, 427–440. Springer. DOI: 10.1007/978-3-319-27702-8\_28
- Zhang, B., Tang, L., and Roemer, M. (2014). “Probabilistic weather forecasting analysis for unmanned aerial vehicle path planning.” *Journal of Guidance, Control, and Dynamics*, 37 (1): 309–312. DOI: 10.2514/1.61651
- Zhao, R., Pang, M., and Wang, J. (2018). “Classifying airborne LiDAR point clouds via deep features learned by a multi-scale convolutional neural network.” *International journal of geographical information science*, 32 (5): 960–979
- Zhu, Z., and Brilakis, I. (2008). “Detecting air pockets for architectural concrete quality assessment using visual sensing.” *Electronic Journal of Information Technology in Construction*, 13: 86–102. Citeseer

## APPENDIX A. DATA AUGMENTATION

### A.1. Flipping Horizontally without Normal Vector

```
import os
import glob
import sys
import numpy as np
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
ROOT_DIR = os.path.dirname(BASE_DIR)
sys.path.append(BASE_DIR)

data_dir = os.path.join(ROOT_DIR, 'data')
mirrored_dir = os.path.join(data_dir, 'mirrored')
if not os.path.exists(mirrored_dir):
    os.mkdir(mirrored_dir)
anno_paths = [line.rstrip() for line in open(os.path.join(BASE_DIR, 'meta/anno_paths_.txt'))]

for anno_path in anno_paths:
    print(anno_path)
    elements = anno_path.split('/')
    area = os.path.join(mirrored_dir+'/'+elements[-3])
    if not os.path.exists(area):
        os.mkdir(area)
    part = os.path.join(area+'/'+elements[-2])
    if not os.path.exists(part):
        os.mkdir(part)
    output_dir = os.path.join(part+'/'+'+Annotations')
    if not os.path.exists(output_dir):
        os.mkdir(output_dir)
    input = np.loadtxt(data_dir+'/'+'+bridge'++'/'+elements[-3]+'/' + elements[-2]+'/' + elements[-2]+'+'.txt', dtype=np.float, delimiter=' ')
    num=len(input)
    out = input
    for i in range(num):
        out[i,0] = -input[i,0]
    np.savetxt(part+'/'+elements[-2]+'+'.txt', out, fmt='%0.3f %0.3f %0.3f %d %d %d')
    for f in glob.glob(os.path.join(data_dir, 'bridge', anno_path, '*.txt')):
        out_filename = os.path.basename(f)
        input = np.loadtxt(f, dtype=np.float, delimiter=' ')
        print (f)
        num=len(input)
        print (num)
        out = input
```

```
for i in range(num):  
    out[i,0] = -input[i,0]  
np.savetxt(output_dir+'/'+out_filename, out, fmt='%.3f %.3f %.3f %d %d %d')
```

## A.2. Flipping Horizontally and Vertically with Normal Vector

```
import os
import glob
import sys
import numpy as np
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
ROOT_DIR = os.path.dirname(BASE_DIR)
sys.path.append(BASE_DIR)

data_dir = os.path.join(ROOT_DIR, 'data')
mirrored_dir = os.path.join(data_dir, 'mirrored')
if not os.path.exists(mirrored_dir):
    os.mkdir(mirrored_dir)
anno_paths = [line.rstrip() for line in open(os.path.join(BASE_DIR, 'meta/anno_paths.txt'))]

for ann_path in anno_paths:
    print(ann_path)
    elements = ann_path.split('/')
    area_h = os.path.join(mirrored_dir+'/'+elements[-3]+'h')
    area_v = os.path.join(mirrored_dir+'/'+elements[-3]+'v')

    if not os.path.exists(area_h):
        os.mkdir(area_h)
    if not os.path.exists(area_v):
        os.mkdir(area_v)

    part_h = os.path.join(area_h+'/'+elements[-2])
    part_v = os.path.join(area_v+'/'+elements[-2])
    if not os.path.exists(part_h):
        os.mkdir(part_h)
    if not os.path.exists(part_v):
        os.mkdir(part_v)

    output_dir_h = os.path.join(part_h+'/'+ 'Annotations')
    output_dir_v = os.path.join(part_v+'/'+ 'Annotations')

    if not os.path.exists(output_dir_h):
        os.mkdir(output_dir_h)
    if not os.path.exists(output_dir_v):
        os.mkdir(output_dir_v)
    input = np.loadtxt(data_dir+'/'+ 'bridge'+ '/' +elements[-3]+ '/' +elements[-2]+ '/' +elements[-2]+ '.txt', dtype=np.float, delimiter=' ')
```



## APPENDIX B. DATA ARRANGEMENT

### B.1. Step 1: Preparing Dataset

The collected data should be got ready to use in the learning process. In ML, more data lead to better results and a more reliable model. Therefore, instead of using each scanned bridge as one input, their surfaces are divided to make many point cloud sets. Each set may include one or more defects, which should be annotated manually. In this step, preparing point cloud data sets, the following rules should be considered:

- The reference plane is on  $XZ$  plane, and the surface defects such as cracks and spalling are located behind the plane. It means the  $y$  coordination of defects is negative.
- Only the coordinates  $(x, y, z)$  and RGB of each point should be kept in raw data. Other parameters, three scalars, are deleted in CloudCompare software.
- Since the blocks have a box shape, the scanned surfaces are divided into rectangular segments
- Each segment should be big enough to contain different sizes of defects and small enough to avoid covering many undamaged areas.
- In the segments with lower density the distance between points are higher, which lead to lower accuracy in annotation and adding more points randomly during up-sampling.
- The same pattern should be considered in data annotation.
- The points are classified into three classes: spall, crack, and nondefect.

Several rectangular segments are selected and cut from each scan of the bridges using the following considerations: 1) the minimum size of the segments was considered around  $0.6\text{ m} \times 0.6\text{ m}$  and the average size was about  $2\text{ m} \times 2\text{ m}$ .; 2) The number of points should not be less than 150,000 or more than 400,000 points. Figure B-1 shows a sample of a segment of the bridge, which is selected for annotation as an input.



*Figure B-1. A segment of a concrete bridge including defects in Cloud Compare.*

The file is named as “part\_segment number”. The file is saved in \*.txt format (e.g. part\_01.txt) in a folder with the same name (e.g. part\_01). Moreover, “Annotations” folder should be created in this folder (Figure B-2).

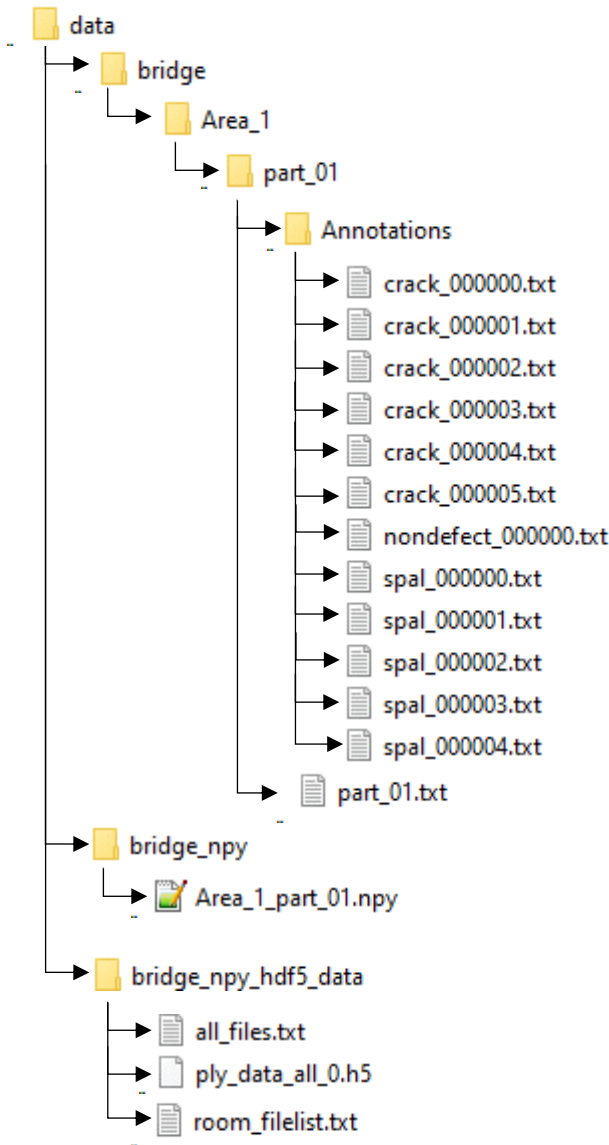







Figure B-2. Arrangement of dataset.

After separating and saving all segments, open the \*.txt file in CloudCompare and use the translate/rotate tool  to put the surface in the y direction and segment in  all spalls and cracks one by one using segment tool . Select all cracks on the left column, save  them in folder Annotations as “crack”. Do the same for spalls and save them as “Spal”. The numbering is done automatically starting from 000000. Any remaining point is assigned to “nondefect” class and saved as one file: “nondefect\_000000”.

Note: during segmentation, you can use “scalar” or RGB mode. But, before saving the annotated segments, delete the scalar using . In training and testing, only this parameter is not used.

Note: If the segmented piece is not located in the main “selected” piece, the error “there is not enough memory” will appear.



Note: Follow the exact rule for naming the files.

## B.2. Step 2: Preprocessing Dataset



As shown in Figure B-2, the raw data is stored in “bridge” and two empty folders “bridge\_npy” and “bridge\_npy\_h5” are created to store numpy and h5 files, respectively. First, based on some given parameters (e.g., size of the block, stride size) in `indoor_3dutil.py`, all the point clouds which are listed in “`/sem_seg/meta/anno_paths.txt`”, are converted into \*.npy format using `collect_indoor3d_data.py`. The name of the numpy files should be listed as “`/sem_seg/meta/all_data_label.txt`”. Then, some parameters (e.g. number of points, block size, stride size) are set in “`gen_indoor3d_h5.py`”, which is used to generate the h5 files based on the npy. The list of h5 files should be written in “`all_files.txt`” file as shown in Figure B-2. These files are used as the input of the training model.

## B.3. Step 3: Training and Testing

“`train_and_test_multigpu.py`” is used for training the dataset and testing it on multi GPUs. The results including a model, the confusion matrix, and the required calculated parameters during training and testing, are stored in “log” folder.

# APPENDIX C. APPLICATION FOR FLIGHT OPERATION CERTIFICATION

## C.1. Application Form

 Transport Canada / Transports Canada	
<b>APPLICATION FOR A SPECIAL FLIGHT OPERATIONS CERTIFICATE (SFOC) FOR THE OPERATION OF A UNMANNED AIR VEHICLE (UAV) SYSTEM IN CANADIAN AIRSPACE</b>	
This form can either be filled in on screen then printed, signed and submitted to the appropriate regional office OR, be printed then completed in BLOCK CAPITALS using black or dark blue ink and submitted to the appropriate regional office	
<b>References:</b> 1. Aeronautics Act; 2. Canadian Aviation Regulations; 3. Staff Instruction SI 623-001 SI Issue No. 02; 4. Advisory Circular AC 600-004 Issue No. 02; and 5. TP 15263 Knowledge Requirements for Pilots of Unmanned Air Vehicle Systems under 25 Kg	
(*) Indicates there is a Note in the Amplifying Note section	
<b>SECTION 1 – BASIC INFORMATION</b>	
1. Application date (yyyy-mm-dd)  <b>2017-07-27</b>	2. Type of application * Select one of the following: <input type="radio"/> Simplified <input checked="" type="radio"/> Complex <input type="radio"/> Compliant <input checked="" type="radio"/> Site-Specific <input type="radio"/> Standing
3. Name and address of the Company or Private Operator *  <b>Concordia University, 1455 de Maisonneuve Blvd. West, EV-9.215, Montreal, QC H3G 1M8</b>	4. Nationality *  <input checked="" type="radio"/> Canadian UAV Operator <input type="radio"/> Foreign UAV Operator
<b>SECTION 2 – PERSONNEL INFORMATION</b>	
<b>5. APPLICANT *</b>	
6. Title <b>Dr.</b>	7. First name <b>Amin</b>
10. Position title <b>Professor</b>	11. Candian Civil Licence Number * <b>N/A</b>
12. Email <b>Hammad@ciise.concordia.ca</b>	13. Telephone number (999-999-9999) <b>514-848-2424</b>
14. Cell (999-999-9999) <b>514-261-5640</b>	15. Alternate contact phone number(s) * <b>(514) 848-2424 - ext : 5800</b>
<b>16. OPERATION MANAGER * (Operation Manager's qualifications to be attached with the application)</b>	
<input type="checkbox"/> Same as Applicant	
17. Title <b>Ms.</b>	18. First name <b>Neshat</b>
21. Position title <b>PhD. Student</b>	22. Candian Civil Licence Number * <b>N/A</b>
23. Email <b>n_bolour@encs.concordia.ca</b>	24. Telephone number (999-999-9999) <b>514-848-2424</b>
25. Cell (999-999-9999) <b>514-946-6065</b>	26. Alternate contact phone number(s) * <b>(514) 848-2424 - ext : 7074</b>
<b>27. GROUND SUPERVISOR * (On site Ground Supervisor's qualifications to be attached with the application)</b>	
<input checked="" type="checkbox"/> Same as Applicant <input type="checkbox"/> Same as Operation Manager	
28. Title <b>Dr.</b>	29. First name <b>Amin</b>
32. Email <b>Hammad@ciise.concordia.ca</b>	33. Telephone number (999-999-9999) <b>514-848-2424</b>
34. Cell (999-999-9999) <b>514-261-5640</b>	35. Alternate contact phone number(s) * <b>(514) 848-2424 - ext : 5800</b>
26-0835E (1706-01) Page 1 of 12	
	

<b>36. PRIMARY UAV PILOT * (All UAV Pilot's qualifications to be attached with the application)</b>			
<input type="checkbox"/> Same as Applicant <input checked="" type="checkbox"/> Same as Operation Manager <input type="checkbox"/> Same as Ground Supervisor			
37. Title <b>Ms.</b>	38. First name <b>Neshat</b>	39. Middle name(s)	40. Last name <b>Bolourian</b>
41. Position title <b>PhD. Student</b>		42. Candian Civil Licence Number * <b>N/A</b>	
43. Email <b>n_bolour@encs.concordia.ca</b>		44. Telephone number (999-999-9999) <b>514-848-2424</b>	45. Cell (999-999-9999) <b>514-946-6065</b>
<b>46. PRIMARY VISUAL OBSERVER * (All Visual Observer's qualifications to be attached with the application)</b>			
<input type="checkbox"/> Same as Applicant <input checked="" type="checkbox"/> Same as Operation Manager <input type="checkbox"/> Same as Ground Supervisor			
47. Title <b>Ms.</b>	48. First name <b>Negar</b>	49. Middle name(s)	50. Last name <b>Salimzadeh</b>
<b>51. SYSTEM MAINTAINER * (All Visual Observer's qualifications to be attached with the application)</b>			
<input type="checkbox"/> Same as Applicant <input checked="" type="checkbox"/> Same as Operation Manager <input type="checkbox"/> Same as Ground Supervisor			
52. Title <b>Ms.</b>	53. First name <b>Neshat</b>	54. Middle name(s)	55. Last name <b>Bolourian</b>
<b>SECTION 3 – UAV INFORMATION (if multiple UAV's, add their information in Section 6)</b>			
56. UAV System	UAV #1	UAV #2	UAV #3
57. UAV Manufacturer *	<b>DJI Matices 100</b>		
58. UAV Name	<b>Matices 100</b>		
59. UAV Serial Number *	<b>N/A</b>		
60. UAV Maximum Take-Off Weight (MTOW)	<b>3600 g</b>		
61. Command and Control Frequency *	<b>5.725~5.825 GHz</b>		
62. Modifications done on the UAV System * (Description to be attached with the application)	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input type="radio"/> No
63. Maintenance procedure in place *	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input type="radio"/> No
<b>SECTION 4 – LIABILITY INSURANCE INFORMATION</b>			
64. Insurance company name		65. Name of insured *	
66. Policy number		67. Policy value	
68. Effective date (yyyy-mm-dd)		69. Expiry date (yyyy-mm-dd)	
<b>SECTION 5 – OPERATIONAL INFORMATION</b>			
70. Type and purpose of the operation * (use Section 6 if required) <b>Testing the basic functions of the drone</b>			
71. VLOS * <input checked="" type="checkbox"/>	72. BVLOS * <input type="checkbox"/>	73. 400 feet AGL and below * <b>max 33 feet</b>	74. Above 400 feet AGL *
75. Dates, alternate dates and times of the proposed operation * <b>Aug.26 (or Aug.27 to Sept.9), from 8am.to 5pm.</b>		76. Day only * <input checked="" type="checkbox"/>	77. Day and night * <input type="checkbox"/>
78. Location(s) of the operation * (use Section 6 if required)  <b>7200 Sherbrooke St W, Montreal, QC, H4B 1R2 (The football field of Loyola Campus)</b> <b>GPS coordinate: 45.457648, -73.636211</b>			
79. Permission from the land owner(s) * <input checked="" type="radio"/> Yes <input type="radio"/> No		80. Permission from the event organizer(s) * <input type="radio"/> Yes <input type="radio"/> No <input checked="" type="radio"/> N/A	
81. FIR/ATC/FSS/Aerodrome Operator Coordination * (Coordination procedures to be attached with the Site Survey/Plan of Operation) <input type="radio"/> Yes <input type="radio"/> No <input checked="" type="radio"/> N/A		82. Type of Airspace * <input type="checkbox"/> Class A <input type="checkbox"/> Class B <input checked="" type="checkbox"/> Class C <input type="checkbox"/> Class D <input type="checkbox"/> Class E <input type="checkbox"/> Class F <input type="checkbox"/> Class G	

83. **Site Survey/Plan of Operation** (For standing applicants, provide a template of a site survey/plan of operation and any guidance material for staff) \*  
**It is explained in section 3.**

84. **Security Plan** (For standing applicants, provide your procedures for a security plan and any guidance material for staff) \*  
**It is explained in section 4.**

85. **Emergency Plan** (For standing applicants, provide your procedures for an emergency plan and any guidance material for staff) \*  
**It is explained in section 5.**

**SECTION 6 – ADDITIONAL INFORMATION (continue on an additional sheet if necessary)**

86. Additional Information \*

**SECTION 7 – FALSE DECLARATION STATEMENT**

87. Paragraph 7.3(1)(a) of the *Aeronautics Act* states: "No person shall knowingly make any false representation for the purpose of obtaining a Canadian aviation document or any privilege accorded thereby. An individual who is convicted of this type of offence is punishable on summary conviction to a fine not exceeding \$5,000 or to imprisonment for a term not exceeding one year, or to both a fine and imprisonment. A corporation that is convicted of this type of offence is punishable on summary conviction to a fine not exceeding \$25,000. Aviation Enforcement also has the option of assessing a punitive suspension of a Canadian Aviation Document (CAD) rather than proceeding by summary conviction or indictment, and depending on the circumstances and other factors surrounding the offence, may take this course of action.

**SECTION 8 – DECLARATION**

88. I, the applicant, agree that the UAV system will be operated in accordance with the Special Flight Operations Certificate (SFOC) \*

I hereby certify that the Staff Instruction SI 623-001 has been read and understood.

I hereby certify that each pilot has been trained and is qualified to safely operate the UAV system.

I hereby declare that to the best of my knowledge the information entered on this application is accurate.

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date (yyyy-mm-dd)

The applicant is over 18 years of age

89. I have enclosed the following supporting documentation (\*\* **Mandatory Documents to be attached with this application**)

- |  |  |  |   |
|--|--|--|---|
| <input checked="" type="checkbox"/> ** Copy of Liability Insurance Certificate | <input type="checkbox"/> ** Operation Manager's Qualification Details                | <input type="checkbox"/> ** Ground Supervisor's Qualification Details            | <input type="checkbox"/> ** UAV Pilot(s) Qualification Details                      |
| <input type="checkbox"/> ** Visual Observer's Qualification Details            | <input checked="" type="checkbox"/> ** Manufacturer UAV Specifications & Limitations | <input checked="" type="checkbox"/> ** Site Survey/Plan of Operation             | <input checked="" type="checkbox"/> ** Security Plan                                |
| <input checked="" type="checkbox"/> ** Emergency Plan                          | <input type="checkbox"/> ** Proof of Corporation (if applicable)                     | <input checked="" type="checkbox"/> ** UAV Operations and Maintenance Procedures | <input type="checkbox"/> ** Description of UAV System Modifications (if applicable) |
| <input type="checkbox"/> Rest. Operator Certificate Aero. (Radio Licence)      | <input checked="" type="checkbox"/> Property Owner's Permission                      | <input type="checkbox"/> Event Organiser's Permission                            | <input checked="" type="checkbox"/> Operations Manual(s)                            |
| <input checked="" type="checkbox"/> Training Manual(s)                         | <input type="checkbox"/> Standard Operating Procedures                               | <input checked="" type="checkbox"/> Check List(s)                                | <input checked="" type="checkbox"/> Operational Flight Plan                         |
| <input type="checkbox"/> UAV Pilot Flight Log(s)                               | <input type="checkbox"/> UAV Maintenance Log(s)                                      | <input type="checkbox"/> UAV Incident/Accident Safety Reporting Form             | <input type="checkbox"/> Medical Certificate/Declaration                            |

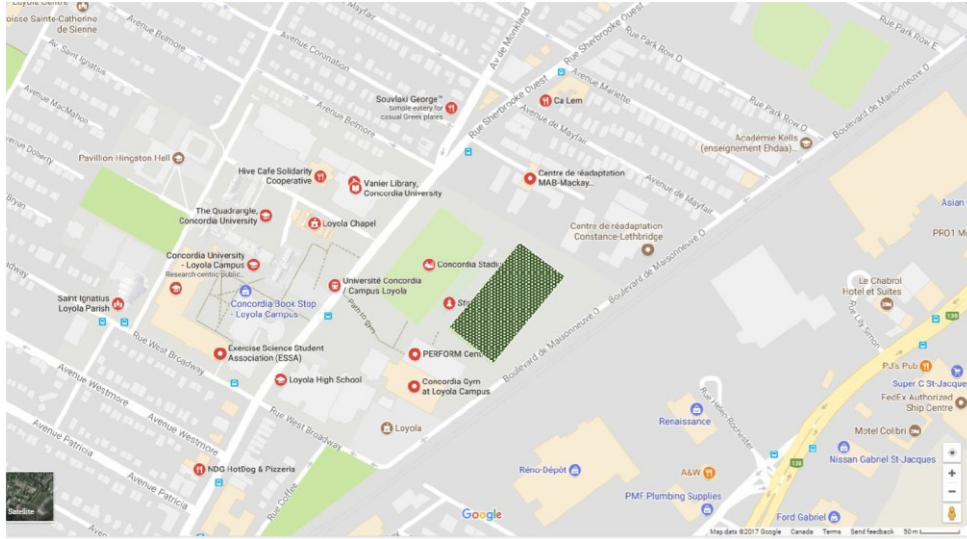
Others: List any other document attached with this application: \_\_\_\_\_

## C.2. Supporting Documents

- **Mandatory Documents**
  - *Copy of Liability Insurance Certificate*: It is Attached
  - *Operation Manager's Qualification Details*: The operation Manager, Ms. Neshat Bolourian, who is the pilot of this operation as well, works on a research related to "Structure Inspection Using UAV". In this operation, she aims to test the UAV. Until now, her experience is only limited to flying a small drone in her lab and she does not have any chance to fly a bigger drone yet. So, there is not any Official Qualification Document available now.
  - *Ground Supervisor's Qualification Details*: The ground supervision, Dr. Amin Hammad, is a professor at Concordia University and he was the supervisor of many projects in last years. "Bridge Inspection Using UAV" is the subject of one of the researches under his supervision right now. At this stage, his student (Ms.Neshat Bolourian) wants to test UAV. No Official Qualification Document is available right now.
  - *UAV Pilot Qualification Details*: The pilot, Ms. Neshat Bolourian, is the same as Operation Manager. As explained before, she has only small experience with a small drone. She does not participate in any classes to take Pilot Certificate till now.
  - *Visual Observer's Qualification Details*: Ms.Negar Salimzadeh, is the visual observer of the operation and one of the students of research group. Although she does not have a chance to participate in an operation yet, she is familiar with her duties and responsibilities.
  - *Manufacturer UAV Specifications & Limitations*: The attached UAV manual includes all specification and limitations.
  - *Site Survey/Plan of Operation*: It explained in Section 3 in detail.
  - *Security Plan*: It explained in Section 4 in detail.
  - *Emergency Plan*: It explained in Section 5 in detail.
  - *Proof of Corporation*: It is not available.
  - *UAV Operation and Maintenance Procedure*: It is attached.
  - *Description of UAV System Modification*: It is not applicable.
- **Optional Documents**
  - **Property Owner's Permission**: The owner of the property is Concordia University and the permission is attached.
  - **Operations Manual**: The manual of UAV (DJI Matrice 100) is attached.
  - **Check List**: It is attached.
  - **Operational Flight Plan**: It is explained in Section 3.

## C.3. Site Survey/Plan of Operation

The following figures shows the location of the operation site. The hatched area is the football field of Concordia University located at 7200 Sherbrooke St. W. , Montreal, QC, H4B 1R2 (45° 27' 28.54",-73° 38' 10.36").



Also, the following figures represent other information related to the operation:



- █ Property Boundaries (Loyola Campus of Concordia University)
- █ Flight Area Boundaries (Football Field)
- █ Flight Path
- Accesses to Sherbrook St. (by car)
- Access to Masisonneuve Blvd. (on foot)
- Pilot
- Supervisor
- Observer
- Guards

#1 : 45° 27' 30.01"N 73° 38' 9.44"W  
 #2 : 45° 27' 26.55"N 73° 38' 13.51"W  
 #3 : 45° 27' 25.05"N 73° 38' 10.89"W  
 #4 : 45° 27' 28.54"N 73° 38' 6.87"W



- Pre-flight Procedure



Before starting the operation, the following steps will be done:

1. Preparing permissions and required documents.
  2. Verifying SFOC approval.
  3. Setting the plan and recheck the path.
  4. Checking the weather whether it is sunny
  5. Checking the area and ensure that the condition and environment is same as before.
  6. Checking the obstacles and clean the area if possible
  7. Verifying flight batteries are fully charged and stable
  8. Placing UAV at the take-off point
  9. Turning on camera and controller
  10. Checking the Bluetooth connection between the remote control and the UAV
  11. Calibrating the UAV
  12. Announcing takeoff to the team
- In-flight Procedure

During the flight, flight path for the UAV and other obstacles should be monitored. In operation plan the flight path is a circle with 10 meter diameter at around 15 m above the ground. Although Matrice 100 has BTH function, the battery level should be monitored. The pilot will announce landing as soon as it starts. If there is not any obstacle or people, UAV will be landed.

For take-off, we have following steps:

1. Placing UAV on the flat ground,
2. Powering on the battery,
3. Launching the camera,
4. Waiting until the Aircraft Status Indicator flashes green,
5. Pushing the throttle stick up slowly to take off or use Auto Takeoff

And for landing:

1. Hovering over a level of surface,
2. Gently pulling down on the throttle stick down for 3 seconds until the motors come to a stop.

During the flight, four members of our team (guards) will stand around the site and protect other people not to enter the site.

- Post-flight Procedure

After finishing the flight, all equipment will be powered down, checked and brought back to the cases. Anything that was put at the place by the team such as notification, sign, etc. should be removed.

#### C.4. Security Plan

The operation is done at the middle of a football field with the standard size (105 by 68 meters). Before starting the procedure, any obstacles such as nets, ball, etc. will be removed. After removal of people from operating area, only the team members will be allowed to enter the field. So, for commencing take-off and landing should be announced to the team in advance.

The permission to access the filed, which is the property of Concordia University, is attached to this form.

Flight limits on height and distance can be set. So, the drone will not fly over limitations. Moreover, the permission from aerodrome authorities is not provided. The operation site is 8.9 km far from the nearest aerodrome authorities, Aéroport Pierre-Elliott-Trudeau, Dorval, QC (less than 9 km).

The operating temperature is between  $-10^{\circ}\text{C}$  to  $40^{\circ}\text{C}$ . The UAV should not be used in adverse weather conditions including raining, snowing, fog, and wind speeds exceeding 10 m/s. In case of low battery and low GPS signal, it should be returned to home.

#### C.5. Emergency Plan

In case of any accident, we should call Concordia University Security (514 848 2424- ext. 3700). Regarding their rules, they are responsible to call 911, provide necessary equipment (first aid kits, fire extinguisher, etc.) or emergency medicals and supports on site. Because of the easy accessibility to the site and availability of emergency equipment at campus, the applicant can deal with many disasters resulting from operation.

#### C.6. UAV Operation and Maintenance Procedure

Operation is subject to the following two conditions: (1) the device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation. This device has warranty for one year which cover any damages related to the Matrice 100. Otherwise, the company will not get the responsibility of the damages. We provided 4 guards for wings to reduce the potential for damage.

#### C.7. Copy of Liability Insurance



## C.8. Summary of the Operation

### (1) Contact information

- Applicant  
Name: Dr.Amin Hammad  
Address: Concordia Institute for Information Systems Engineering (CIISE), Concordia University  
1515 Ste-Catherine Street West, EV7.634, Montreal, Quebec, H3G 2W1  
Tel: (514) 848-2424 ext: 5800, Fax: (514) 848-3171  
E-mail: hammad@ciise.concordia.ca
- Company  
Concordia University
- Operation Manager  
Name: Neshat Bolourian  
Address: Department of Civil, Building and Environmental Engineering, Concordia University, 1455 de Maisonneuve Blvd. West, EV-9.215, Montreal, QC H3G 1M8  
Tel: (514) 848-2424 (ext.7074), Cell: (514) 946-6065  
e-mail: n\_bolour@encs.concordia.ca

### (2) Operation

- Purpose of Operation

The main purpose of this operation is testing the drone in context of the research. This research is done under supervision of Dr. Hammad at Concordia University. So, Concordia University supports it and gives the permission to use on its property. Before running the main case study of the research, a simple test is required. So, this VLOS operation is needed. The flight may only take around 30 min.

The applicant and the operation manager will be present at the site during the operation.

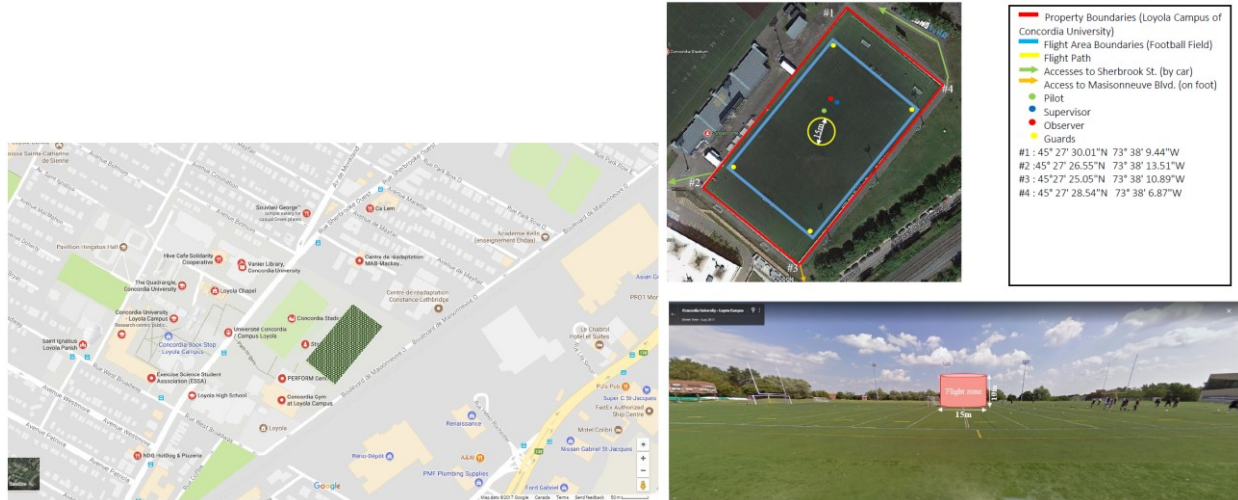
- Date of Operation

The best time for operation is the weekend morning (good light condition with less passengers). The planned date is August 26<sup>th</sup>, 2017. The historical average weather of this day is between 17°C and 25°C. And the alternative date is August 26<sup>th</sup>, 2017. The total required duration is estimated 3 hours which includes 30 minutes for the flight operation.

- Planned date and time : August 26<sup>th</sup>, 2017 , 8 am – 5 pm
- Planned date and time : August 27<sup>th</sup> to September 9<sup>th</sup> 2017 , 8 am – 5 pm

- Operation Location

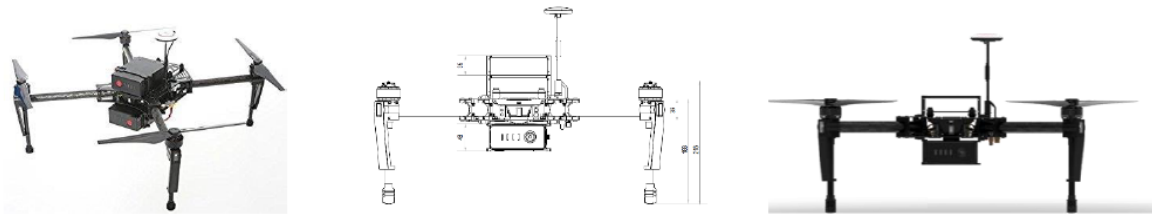
The following figure shows the location of the operation site. The hatched area is the football field of Concordia University located at 7200 Sherbrooke St. W. , Montreal, QC, H4B 1R2 (45° 27' 28.54", -73° 38' 10.36"). Also, the following figures represent other information related to the operation.



- Equipment

- Matrice 100

Matrice 100 is used in this operation which is controlled by a remote control.



The diagonal wheelbase is equal to 650 mm. The specifications are:

- Structure

- Weight (with TB47D battery) 2355 g
- Weight (with TB48D battery) 2431 g
- Max. Takeoff Weight 3600 g
- Expansion Bay Weight 45 g
- Battery Compartment Weight 160 g
- Zenmuse X3 Gimbal with Camera Weight 247 g

- Propulsion system

- Motor Model DJI 3510
- Propeller Model DJI 1345s
- ESC Model DJI E SERIES 620D

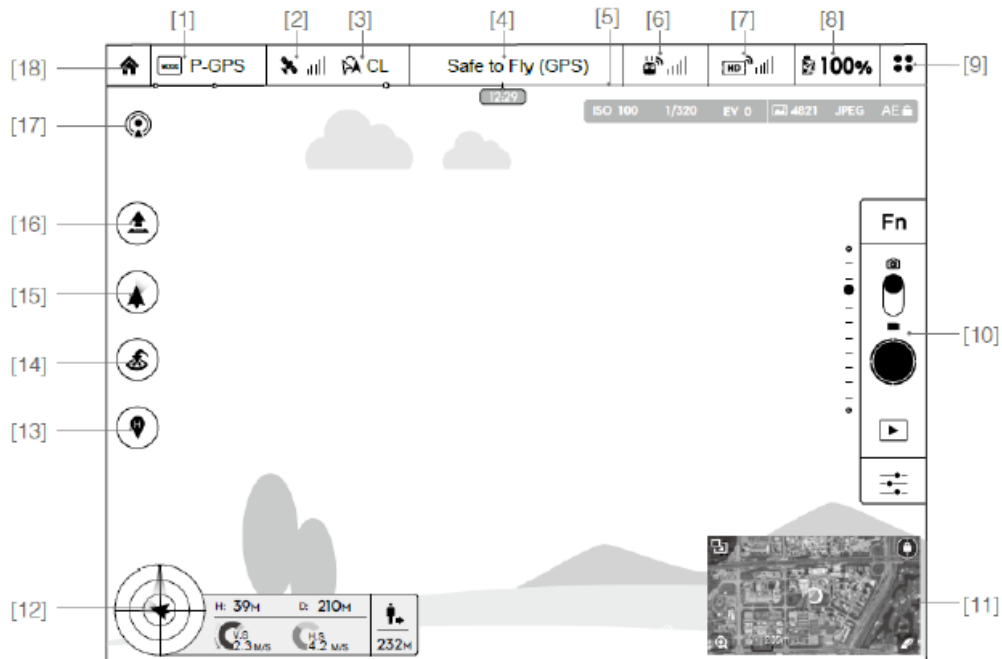
- Other
  - Operating Temperature     -10°C to 40°C
  
- Charger
  - Model                         A14-100P1A
  - Voltage Output             26.3 V
  - Power Rating                100 W
  
- Performance
  - Hovering Accuracy (P-Mode With GPS)     Vertical: 0.5 m, Horizontal: 2.5 m
  - Max. Angular Velocity                     Pitch: 300°/s, Yaw: 150°/s
  - Max. Tilt Angle                             35°
  - Max. Speed of Ascent                       5 m/s
  - Max. Speed of Descent                     4 m/s
  - Max. Wind Resistance                     10 m/s
  - Max. Speed                                 22 m/s (ATTI mode, no payload, no wind)  
   17 m/s (GPS mode, no payload, no wind)

Matrice 100 has “smart return to home”, “low battery return to home” and “fail safe return to home” ability. It will be activated automatically when the signal of the remote control is lost. Also, it can automatically return home if the current battery level becomes low.

The method of takeoff and landing is vertical and the navigation equipment is **A2 PRO PLUS GPS Module**. The operating temperature is between -10°C to 40°C. It should not be used in adverse weather conditions including raining, snowing, fog, and wind speeds exceeding 10 m/s.

➤ Remote Controller

The remote controller is powered by a 2S rechargeable battery with a capacity of 6000mAh. The combined system operates at 2.4 GHz with maximum signal transmission range of 2 km. The screen of the controller (ipad) Pros the following information: (1) flight mode, (2) GPS signal strength, (3) IOC setting, (4) system status, (5) battery level indicator, (6) remote controller signal, (7) HD video link signal strength, (8) battery level, (9) general setting, (10) camera operation bar, (11) mini map, (12) flight telemetry (flight attitude and radar function including speed in both H/V directions, roll and pitch, etc.), (13) Home point settings, (14) RTH, (15) Gimbal operation mode, (16) auto takeoff/landing, (17) livestream and (18) back.



- Personnel
  - Supervisor:

Dr. Amin Hammad has been working as a professor in Concordia Institute for Information Systems Engineering (CIISE) since 2003. There are several research projects which are done under his supervision regarding sustainability and infrastructure management. One of his current projects is about infrastructure inspection using UAV. In his lab, the research group is working on simulating and developing a new path planning method and detecting the defects of structures. Meanwhile, they want to test Matrice 100 in the context of the research.

- Operation Manager (Pilot)

Ms. Neshat Bolourian is a PhD student at Concordia University. She has been doing research about “bridge inspection using UAV” under supervision of Dr. A. Hammad from September 2015. In this step of the project, she wants to do a simple test with the drone which will be used in the further implementation and real case study.

## APPENDIX D. LIST OF PUBLICATIONS

### *Journal Papers*

- **Bolourian, N.**, Nasrollahi, M., Bahreini, F., & Hammad, A. (2022). Point cloud-based concrete surface defect semantic segmentation using modified PointNet++. *Journal of Computing in Civil Engineering* (Submitted).
- **Bolourian, N.**, & Hammad, A. (2020). LiDAR-equipped UAV path planning considering potential locations of defects for bridge inspection. *Automation in Construction*, 117, 103250.

### *Book Chapter*

- **Bolourian, N.**, & Hammad, A. (2019). Path planning of LiDAR-equipped UAV for bridge inspection considering potential locations of defects. In *Advances in Informatics and Computing in Civil and Construction Engineering* (pp. 545-552). Springer, Cham.

### *Conference Papers*

- **Bolourian, N.**, Hammad, A., & Ghelmani, A. (2022). Point cloud-based concrete surface defect semantic segmentation using modified PointNet++. In *29th EG-ICE International Workshop on Intelligent Computing in Engineering*, Denmark.
- Nasrollahi, M., **Bolourian, N.**, & Hammad, A. (2019). Concrete surface defect detection using deep neural network based on lidar scanning. In *Proceedings of the CSCE Annual Conference, Laval, Greater Montreal, QC, Canada* (pp. 12-15).
- Nasrollahi, M., **Bolourian, N.**, Zhu, Z., & Hammad, A. (2018). Designing LiDAR-equipped UAV platform for structural inspection. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction* (Vol. 35, pp. 1-8). IAARC Publications.
- **Bolourian, N.**, Soltani, M. M., Albahria, A. H., & Hammad, A. (2017). High level framework for bridge inspection using LiDAR-equipped UAV. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction* (Vol. 34). IAARC Publications.