3D SEGMENTATION AND DAMAGE ANALYSIS FROM ROBOTIC SCANS OF DISASTER SITES

A Dissertation
Presented to
The Academic Faculty

By

Jingdao Chen

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy in the
Institute for Robotics and Intelligent Machines

Georgia Institute of Technology

May 2021

3D SEGMENTATION AND DAMAGE ANALYSIS FROM ROBOTIC SCANS OF DISASTER SITES

Approved by:

Dr. Yong K. Cho, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Dr. Frank Dellaert
School of Interactive Computing
Georgia Institute of Technology

Dr. Jun Ueda
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Eric Marks
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: December 3, 2020

Life can only be understood backwards; but it must be lived forwards.

Søren Kierkegaard

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Yong Cho for his meticulous guidance and mentorship throughout my PhD studies, and without whom this journey would have been impossible. Even though the PhD grind is full of ups and downs with multiple rejected papers, late nights at work, and tiring field trips, Dr. Cho has been extremely supportive in this process and never ceases to be a source of encouragement for me to become a better student, a better engineer, and a better researcher. Most PhD students would say that PhD life is all work and no play, but I do have to admit that I did have fun during my 6 years at the RICAL lab. In fact, no year at the RICAL lab would be complete without a Korean BBQ feast in which we all have too much food, Thanksgiving dinner at Dr. Cho's place with an impromptu ping-pong tournament, and a random road trip across the state for a research project.

I would like to thank Dr. Zsolt Kira for his help and suggestions for deep learning techniques used in this research. I took the Deep Learning class at Georgia Tech with Dr. Kira in 2016 and have been using deep learning methods in my research ever since. Dr. Kira has also provided help in terms of reviewing papers and offering ideas in my research which eventually contributed to the learnable region growing and incremental segmentation parts of this thesis.

I would like to thank my thesis committee members Dr. Eric Marks, Dr. Frank Dellaert and Dr. Jun Ueda for their suggestions and feedback for this PhD thesis. Their guidance has been invaluable in making sure that this thesis consists of high quality work and is meaningful to a wider, interdisciplinary audience.

I would also like to express my gratitude to my colleagues from the RICAL lab, my accomplices in many different research efforts over the years. I am especially grateful to Yihai Fang and Jay Park, who guided and mentored me during the first few years of my PhD studies; to Pileun Kim, whose work on the GROMI robot is critical to large parts of this

thesis; to Jisoo Park, who helped out significantly in data collection during the Guardian Centers field test; to Shiqin Zeng, John Yi, and Mark Kahoush, for their contribution to point cloud building element retrieval and point cloud scene completion; to Yosuke Yajima; for writing part of the ROS code for incremental segmentation; to Kinam Kim, Carter Price, Inbae Jeong, Minki Kim, and Jitae Kim, with whom I have collaborated on several construction-related projects that are not directly used in this thesis but are still a valuable part of my research.

I would like to thank my collaborators from Hanyang University ERICA for giving me a warm welcome in Korea and helping ensure that the international disaster relief research project could be a success. Dr. Changsoo Han and Dongik Sun have been instrumental in realizing the robotic scanning experiments in Korea whereas Dr. Yonghan Ahn and Byeol Kim have kindly provided data from structural damage experiments that is used in this research.

Last but actually first, I would like to thank my spiritual family at Georgia Tech, Westminster Christian Fellowship and North Avenue Presbyterian Church, who have showed me that light can shine out of darkness, and the darkness has not overcome it. It is this light that has carried me through the toughest of times. Members of the church, especially Neale, Carol, Jason, and Jasmine, have served as role models in their tireless work in welcoming international students at Georgia Tech. Special thanks to Edwin and Shan, my fellow PhD classmates, who have so graciously hosted me at their place as I finish up this thesis document. Many have started their PhD quest in the search of truth, yet I have come to realize that what better way to find it, than in $\lambda\acute{o}\gamma o\varsigma$, the beginning of wisdom.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviations

RANSAC	Random Sample Consensus
LiDAR	Light Detection and Ranging
S3DIS	Stanford 3D Indoor Spaces dataset
RGB	Red-Green-Blue
CAD	Computer-Aided Design
BIM	Building Information Modeling
UAV	Unmanned Aerial Vehicle
CNN	Convolutional Neural Network
FPFH	Fast Point Feature Histogram

Symbols

P	Point cloud scene
$P*$	Point cloud segment created by region growing
L	Instance label vector
Q	Input point cloud for a region growing subproblem

p_{seed}	Seed point for region growing

SUMMARY

Disaster relief and response plays an important role in saving lives and reducing economic loss after earthquakes, windstorm events and man-made explosions. Mobile robots represent an effective solution to assist in post-disaster reconnaissance in areas that are dangerous to human agents. These robots need an accurate 3D semantic map of the site in order to carry out disaster relief work such as search and rescue and damage assessment. Thus, there exists a research need to automatically identify building elements and detect structural damage from laser-scanned points clouds acquired by mobile robots. Current methods for point cloud semantic segmentation mostly perform direct class prediction at the point level without considering object-level semantics and generalizability across datasets. Moreover, current segmentation methods are unsuitable for real-time operation because they are designed to work as a post-processing step and do not process points from new scans in an online manner. This research proposes a learnable region growing method to perform class-agnostic point cloud segmentation in a data-driven and generalizable manner. In addition, an anomaly-based crack segmentation method is proposed where a deep feature embedding is used as a basis for separation between inlier and outlier points. Finally, an incremental segmentation scheme is used to process point cloud data in an online fashion and combine semantic information across multiple scans.

CHAPTER 1
INTRODUCTION

1.1 Background

The hazardous nature of post-disaster scenarios such as earthquakes and hurricanes means that mobile robots have to be used to effectively carry out disaster relief operations. Mobile robots can use sensing devices such as cameras and LiDAR devices to acquire scan data of damaged infrastructure and assess the hazard level of different regions in the disaster site. Studies [1][2] have shown that the key reconnaissance tasks that need to be performed are identifying structural weaknesses, locating key assets, and surveying leakages. The conventional process to perform these tasks is labor-intensive since the raw sensor data has to be manually examined by human agents and lacks interpretability. In addition, analyzing the point cloud data is challenging due to the unstructured nature of disaster sites as well as confounding factors such as occlusion and clutter.

Multiple studies [3][4] have proposed the use of 3D point cloud data as a feature-rich and geometrically-correct way to represent the physical environment. Obtaining semantic information from point clouds is important for robots to carry out place recognition, obstacle avoidance, and object retrieval. Especially, the segmentation process can subdivide a point cloud scene into constituent objects which can help the management and decision-making process in disaster relief. Although not as mature as 2D object recognition, 3D object recognition has gained significant research interest in recent years. Some examples of well-known datasets in the 3D domain are: (i) Stanford Large-Scale Indoor Spaces (S3DIS) [5] for recognition of building elements (ii) KITTI [6] dataset for autonomous driving, and (iii) ModelNet [7] and ShapeNet [8] for recognition of Computer-Aided Design (CAD) models.

However, there are some limitations in existing methods for 3D object recognition. Methods that perform classification at the point level (i.e., semantic segmentation) do not output the object boundary which is important to delineate separate objects in a scene. On the other hand, methods that represent objects by 3D bounding boxes are inaccurate when used in situations with non-rectangular objects or when bounding boxes overlap, since bounding boxes only constitute a rough approximation of the object boundary [9]. Moreover, methods for point cloud semantic segmentation such as PointNet [10] and SGPN [11] are not suitable for real-time scanning because they are fundamentally offline methods and do not process data incrementally.

This research proposes an improved methodology for 3D segmentation of robotics scans as follows: (i) learnable region growing is used to improve generalizability of instance segmentation, (ii) anomaly-based damage detection from point cloud is used to implement crack segmentation without explicit training, (iii) a multi-view context pooling (MCP) method is used to implement incremental segmentation and combine information from multiple viewpoints.

1.2 Literature Review

1.2.1 Point Cloud Object Recognition

The simplest method to recognize objects from point cloud data is to assume that the objects fall into simple geometric categories such as planes and cylinders [12] [13]. The corresponding geometric parameters can then be estimated using Random Sample Consensus (RANSAC) or Hough transform [14] [15]. Another simple segmentation method is region-growing, which initializes a seed point and gradually expands the region in a local neighborhood to form objects [16] [17] [18]. In cases where 3D Computer-Aided Design (CAD) models of the objects of interest are available, registration and matching can be used to find the objects in a point cloud scene [4] [19] [20]. The limitation of this approach is that it is not practical in situations where the design models are not available, such as on

historical building sites.

Later research focused on the problem of recognizing complex objects by using the idea of feature descriptors to numerically encode and summarize the characteristics of 3D objects. 3D feature descriptors fall into 2 different categories, which are local descriptors and global descriptors. Local descriptors, such as Fast Point Feature Histogram [21], Viewpoint Feature Histogram [22], and Spin Images [23], work by binning features in a local neighborhood (i.e. normals, curvature, and density) into histograms. An object can then be recognized by finding correspondences in terms of these local descriptors. In contrast, global descriptors such as Global Radius-based Surface Descriptor [24], Principal Axes Descriptor [25], and Ensemble of Shape Functions [26], work by computing overall properties of an object such as area, length, and angles and encoding those values into a feature vector. Then, classical machine learning classifiers such as support vector machines or K-nearest neighbors may be use to predict the class label based on this feature vector. Overall, the use of feature descriptors is advantageous over simpler rule-based methods because it allows the complex geometry of point cloud objects to be described numerically.

More recent research have been focused on the use of neural networks for point cloud object recognition due to the prevalence of deep learning in the greater research community. Deep learning methods are attractive because they can directly model the relationship between a 3D point cloud object and its corresponding class label without the need to manually engineer feature descriptors. Deep learning methods have also achieved impressive results in modern computer vision benchmarks [27]. Initial approaches to perform object recognition in the point cloud domain are based on 3D convolutions which are an extension to 2D convolutions commonly used in the image domain. In this case, the input point cloud is first converted into a voxel representation or occupancy grid before being passed into a 3D convolutional neural network. Some examples of these networks are 3D Network-in-Network [28], Voxception-ResNet [29], VoxNet [30], ShapeNets [7]. The limitation of these methods are that 3D convolutions are computationally expensive, and voxel grids are

unable to capture fine geometric detail if the voxel size is too large. In addition, these methods are limited to object-level classification and do not perform point-level classification.

An alternative category of deep learning methods for 3D object recognition uses bounding boxes as the choice of object representation [31][32][33]. These methods are focused on using regression methods to estimate the location and extent of specific 3D objects in a point cloud scene. These methods are popular in the autonomous driving domain which is focused on a narrow set of object classes such as cars and pedestrians and do not usually require fine point-level classification. Some examples of works in this category are MV3D [32], which generates object bounding boxes by fusing information from images and birds-eye view LiDAR, and Frustum PointNets [34], which predicts object bounding boxes by projecting object proposals from the 2D images to 3D space. The main limitation of these methods is that they need to either train separate networks for each class [35] or use class-specific anchor boxes [32]. Moreover, representing objects in the form of bounding boxes often result in ambiguity in cases where there are non-rectangular objects or when multiple objects overlap.

Another line of work in the domain of deep learning methods for 3D object recognition uses unstructured point clouds directly as input to the neural network. This line of work is driven by works such as PointNet [10] and PointNet++ [36] which proposed the idea of using 1x1 convolutions in combination with max-pooling to combine features across an unstructured point cloud. Later works further expand on this idea by introducing structures such as Recurrent Consolidation Units [37], grouping matrix [11] or learnable embeddings [38] [39] to improve the semantic segmentation performance and also jointly perform instance segmentation. One problem with these methods is that they have no elegant way to handle objects that do not fall neatly into the set of pre-defined object class, often just designating them as "clutter" objects. In addition, these methods are not continuous in the sense that they require the point cloud scene to be subdivided into 1m x 1m blocks before the prediction stage, and a *BlockMerging* [11] step is used to merge the predictions across

different blocks in the post-processing stage. This could lead to inaccurate prediction along the boundaries of the point cloud blocks.

In the robotics literature, point cloud segmentation is commonly performed through region growing [40][41][22][42] due to its speed and simplicity. Methods that incorporate neural networks can usually only process data on a scan-by-scan basis due to the computational demands. There have been some attempts to make use of information from multiple viewpoints such as joint viewpoint prediction and categorization [43], grouped pooling [44], or global view pooling [45]. However, the final information merging step is still performed offline as a post-processing step since it is computationally demanding. In summary, there is still a research need for 3D object recognition methods that are robust, generalizable, and work in the real-time robotic setting.

1.2.2 Damage detection using mechanical sensors

A common method for structural health monitoring of civil structures is the use of vibration sensors [46][47]. Vibration sensors can be used to monitor for frequency shifts and mode shape changes that are linked to changes in structural properties. However, this method depends on pre-installation of a large number of sensors which is expensive and only practical for critical structures such as bridges. Acoustic emission is another commonly used technique for damage evaluation of concrete structures and steel structures [48][49]. Acoustic emission testing relies on sensors that can listen for acoustic events and measure the component strength and risk of failure. However, acoustic emission testing usually requires physical contact with the damaged structure which is impractical in high-risk or inaccessible regions of a disaster site. Other methods such as ultrasonic testing [50] and guided wave testing [51] make use of actively emitted pulses instead of passively measuring acoustic waves. Ultrasonic testing is commonly used to detect internal defects but only works on an area in close proximity to the transducer. On the other hand, guided wave testing can be used to inspect large areas of a structure at once but cannot localize the damaged region

accurately.

1.2.3 Damage detection using 2D images

The main benefit of using images for damage detection instead of mechanical sensors is that they can be acquired from longer ranges and are lightweight such that they can be equipped on Unmanned Aerial Vehicles (UAV). There are several works that use deep learning techniques for crack detection from images. Zou et al. [52] used a deep convolutional neural network (CNN) for automatic crack detection by learning high-level features for crack representation. The idea of using CNNs can be extended to use guided filtering and Conditional Random Fields (CRFs) to refine the final prediction results [53]. Another possibility is to pre-process the images using homomorphic filtering and the Otsu thresholding method before passing them to the network [54]. Benz et al. [55] studied the problem of crack segmentation from images acquired by unmanned aircraft systems (UAS). The study uses transfer learning from an auxiliary dataset to initialize the weights of a neural network for crack segmentation. However, the network still requires fine-tuning in a supervised manner from an actual labeled dataset of cracks. In the industrial domain, Liu et al. [56] used an encoder-decoder network in conjunction with background suppression modules to detect cracks from images of plastic materials. Whereas, Tabernik et al. [57] studied surface crack detection from images of industrial products. Since the work is targeted for quality control applications, it focused more on the decision problem of identifying images with cracks rather than the detection problem of accurately segmenting the cracks. One common limitation is that these methods require strong supervision in the form of a database of images with labelled cracks. In addition, these methods are only applicable in the 2D domain and do not predict the 3D geometric information that is necessary to determine the crack dimensions.

A subset of works in the literature focus on the highly related problem of crack detection from asphalt pavements. Early studies use the intensity as an indicator feature for

cracks [58][59]. The problem with these methods is that the performance is negatively by shadows, lighting effects, non-uniform crack widths, and poor intensity contrast [59]. CrackNet [60] was one of the first large-scale studies into pavement crack detection using deep learning techniques. The study uses a convolutional neural network (CNN) to detect cracks in a dataset of 2000 pavement depth images generated by a laser scanning system. When compared to non-deep learning methods such as 3D shadow modeling and Support Vector Machines (SVM), CrackNet achieved much more accurate crack detection. CrackNet II [61] was introduced as an improvement over the original CrackNet with a deeper network architecture that has fully learnable parameters. This resulted in higher precision and recall as well as processing speed. CrackNet-V [62] further improved the network design with deeper convolutional layers and a more specific activation function and the performance was also validated using a larger dataset. To overcome the data deficiency problem for crack images, Li et al. [63] proposed semi-supervised learning using a combination of labeled and unlabeled pavement images. In addition, an adversarial learning method was used to improve the segmentation predictions. Several works [64][65] explore crack detection using fully unsupervised algorithms such as Otsu's thresholding and Random Sample Consensus (RANSAC). However, these algorithms require manual tuning of the threshold hyperparameters to achieve good performance.

1.2.4 Damage detection using 3D point clouds

A simple method to perform damage detection is to measure the deviation between the post-damage point cloud and the pre-damage point cloud that can be generated from previous scans or from design models [66][67]. Compared to the image domain, this method is more practical in the point cloud domain since point clouds can capture the 3D geometry in absolute scale. The problem with this method is that it requires the pre-damage point cloud which may not be available for all buildings on the disaster site. A geometry-based method to detect cracks and spalling in point cloud data is to estimate the surface normal

and measure the deviation by comparing to a nominal value [68] or by finding outliers after combining the surface normal and color information [69]. Other alternative methods for damage detection include measuring the perpendicular distance from a pre-defined reference plane [70]. However, the limitation of these geometry-based methods is that they require significant parameter tuning and carefully constructed scoring algorithms.

Similar to the image domain, some researchers turned to machine learning techniques for damage detection in the point cloud domain. The problem is that large databases with annotated damages exist in the image domain (e.g. CrackTree260 [52], DeepCrack [53], and UAV 75 [55]) but are difficult to find in the point cloud domain. One possible solution is to synthetically generate deformations in the point cloud data derived from a database of building element Computer Aided Design (CAD) models [71]. However, this method relies on having the building elements cleanly pre-segmented from a point cloud scene which is difficult to achieve on real disaster sites due to a high amount of occlusion and clutter. Another related method to synthesize deformations such as spalling directly on the point cloud is to apply cubic surface equations together with added noise [72]. Then, geometry-based classifiers can be trained to segment out damaged regions without reliance on color or intensity information. However, this method still results in a large number of false positives due to surface nonuniformities and sharp features such as edges. In the case of larger-scale damage such as that caused by hurricanes, an airborne LiDAR scanning system can be used to gather point clouds of the disaster site [73]. Building damage indicators can be derived from geometric features such as including roof area and volume, roof orientation, and roof shape. This method works well for macro-level building damage such as destroyed roofs or collapsed structures but does not work for micro-level damage indicators such as cracks.

Damage information from civil infrastructure can also be summarized and stored in the form of as-damaged Building Information Models [74][75]. These models contain detailed and semantically-rich information about the location and shape of infrastructure damage present in a building that can be useful to civil engineers and reconstruction and recovery

personnel. In order to automatically generate these models, it is important to reconstruct the full point cloud scene to provide context for the damaged regions and not just describe infrastructure damage in a local context.

1.3 Objectives

The main objective of this research is to *identify an automated, accurate, and efficient method to organize and interpret point cloud data collected from laser-scanning robots on disaster sites*. The proposed method should be capable of instance segmentation of point cloud data acquired from building structures in a post-disaster reconnaissance scenario to label different types of building structures and identify damaged structures. The specific research objectives are as follows:

Objective 1: Investigate a learnable region growing method for point cloud segmentation

Class-specific point cloud segmentation face issues with generalizability across datasets due to mismatched classes. Using a learnable region growing method for class-agnostic point cloud segmentation can allow for more generalizable segmentation results while maintaining the data-driven and continuous properties.

Objective 2: Identify damaged areas such as cracks from point clouds of building components based on anomaly-detection

Strictly supervised methods for damage detection from point clouds are impractical due to a lack of labelled data for post-disaster environments. This research proposes a crack segmentation method where anomalies are detected based on deep point-level features vectors. The method is potentially advantageous because it allows the damaged region to be extracted without requiring explicit training.

Objective 3: Investigate incremental segmentation of laser scans acquired from mobile robots

Current point cloud semantic segmentation methods are poorly suited to robotic real-time scanning because they are meant to operate on single point clouds and do not have a mechanism to incorporate information from new scans incrementally. To address these issues, this study proposes a multi-view incremental segmentation method for online segmentation of point clouds.

CHAPTER 2

POINT CLOUD INSTANCE SEGMENTATION

Robots commonly make use of 3D perception to understand and interact with their surroundings [40][41]. With advances in technology such as LiDAR, depth cameras, and photogrammetry, the use of 3D point clouds to represent the geometry of a 3D scene has rapidly increased in popularity [5][36][76]. While modern sensor technology allows 3D point clouds to be collected on a large-scale basis, interpreting the raw point cloud data to infer structure and semantics remains a fundamental challenge.

Point cloud segmentation is a relevant component of robotic sensing due to its use in localization [41], obstacle detection [66], object recognition [77]. Point cloud segmentation essentially refers to the task of subdividing an input point cloud scene its constituent objects. Point cloud segmentation can take the form of semantic segmentation, where each point is assigned a class label (e.g. chair, table, wall) or instance segmentation, where each point is assigned an instance label (e.g. object 1, object 2).

One category of approaches to point cloud segmentation is to compute local features such as color, normals, or curvature and apply a threshold on these features to connect points together in a region growing process [78][22][79][41]. These approaches are popular in the robotics domain because they are simple, continuous, and are computationally efficient. The main issue with these approaches is that they are sensitive to the threshold hyperparameter and frequently result in undersegmentation or oversegmentation.

Another category of approaches to point cloud segmentation is to use deep neural networks to predict the segmentation labels [11][38][31]. These approaches have gained significant research attention in the past few years due to achieving state-of-the art performance in computer vision benchmarks. The main benefit of these learning-based approaches is that they are data-driven and require less manual tuning of hyperparameters or

hand-engineering of feature descriptors. However, current deep learning methods for point cloud segmentation have the limitation that they are not continuous. In particular, a block division step is require to subdivide a point cloud scene into smaller blocks. In addition, most of the deep learning approaches are class-specific, i.e. they are designed to work with a pre-defined set of classes and not for general segmentation of objects.

The key idea of this research is to formulate the region growing segmentation process as a deep learning problem and use a deep neural network, LRGNet, to predict the output of each region growing subproblem. This way, the method is able to have the advantages of each type of approach, i.e. it is continuous and class-agnostic, but is also data-driven and generalizable. The following sections will present, in order, the (i) problem definition, (ii) LRGNet network architecture, (iii) region growing simulation, (iv) inference process, and (v) experimental results.

2.1 Problem Definition

The problem of point cloud segmentation is defined as taking an input point cloud with N points and predicting an integer instance label for each of the N points. Suppose that the input, **P**, is represented as an $N \times F$ matrix, where F is the number of features, and the output is represented as an N-dimensional integer vector, **L**.

The idea of region growing segmentation is to consider a point cloud region that starts from a seed point p_{seed}, and gradually morph that region into a set of points that represents a complete object. Denoting the set of all points that belong to the same object instance as p_{seed} as P^*, the goal of region growing is to incrementally add points to the current region until the optimal set P^* is achieved. Once the region growing process terminates for a single object, the points in the region are assigned the same unique instance label and the region growing process starts over again with a new seed point. This process continues until all N points in P have been assigned instance labels.

The region growing problem can be broken down into smaller subproblems at each

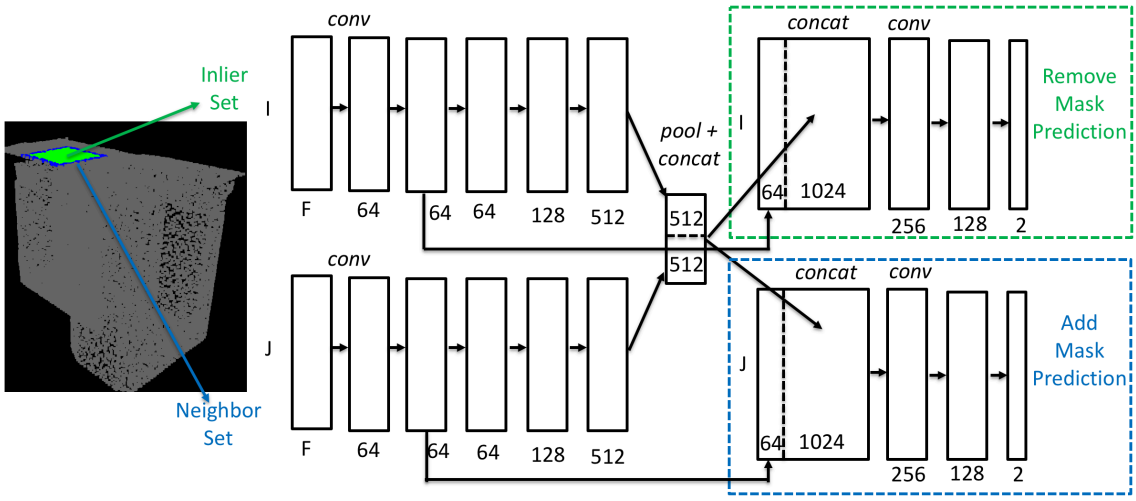


Figure 2.1: LRGNet network architecture

decision point of region growing. Supposing that the subproblem involves morphing the current region at step k, Q_k, into the region at step $k+1$, Q_{k+1}, we can formulate the subproblem into solving for the function f such that $f(Q_k) = Q_{k+1}$. Thus, starting at the seed point, we have $Q_0 = \{p_{seed}\}$. Then, we can iteratively apply the function f so that $Q_1 = f(Q_0)$, $Q_2 = f(Q_1)$, etc. until Q_k approaches the optimal region P^*. In this study, the function f is derived through a data-driven process, allowing us to arrive at a "learnable" region growing method.

2.2 Learnable Region Growing

2.2.1 Network architecture

The function f is computed using a deep neural network, LRGNet, as shown in Figure 2.1. The network takes in a set of inlier points and a set of neighbor points and predicts how to add and remove points from the point sets to create a new point set. The upper input branch of the network comes from the inlier set, which is defined as a set of I points that lie in the current region Q_k. Whereas, the lower input branch of the network comes from the neighbor set, which is defined as a set of J points that are unlabeled and are within δ-distance of any point in Q_k. The I and J points are obtained by random sampling

with replacement when there are fewer or more than I/J points. Both the inlier set and neighbor set consist of points with F features each. In this study, the input set of features are the local XYZ coordinates, room-normalized XYZ coordinates, RGB color, normals, and curvature. The local XYZ coordinates are determined with respect to the center of the current region whereas the room-normalized XYZ coordinates are determined with respect to the entire room. The normal and curvature values at each point are computed using Principal Component Analysis (PCA). Thus, the total number of features is $F = 13$. Other than the room-normalized XYZ corodinates, all the feature values are normalized by subtracting the column-wise median over all the input points.

The input matrices are then processed by a series of feature-wise convolutional layers, similar to that used in PointNet [36]. The features from each point are combined into a 1024-dimensional global feature vector by a max-pooling operation across all points followed by a concatenate operation between the upper and lower branches. This global feature vector is then concatenated back to each of the feature sets of the I and J points. These concatenated feature vectors are then passed through a few more convolutional layers before resulting in output layers in two branches. The upper output branch predicts the remove mask, which determines which of the I points from the current region should be removed when generating the next region. Whereas, the lower output branch predicts the add mask, which determines which of the J points from the neighbor set should be added to the next region. This mechanism of having both added points and removed points allow the region growing process to correct for errors. That is, if a point from the neighbor set is incorrectly added to the region at the current step, there is still a possibility that that point can be removed at a future step. Given the predicted add mask and remove mask, the current region Q_k is updated by adding points and removing points to create the next region Q_{k+1}.

The hyperparameters for neighbor threshold distance, δ, size of the inlier set, I, and size of the neighbor set, J are set to 0.1m, 512, and 512 respectively. These values are

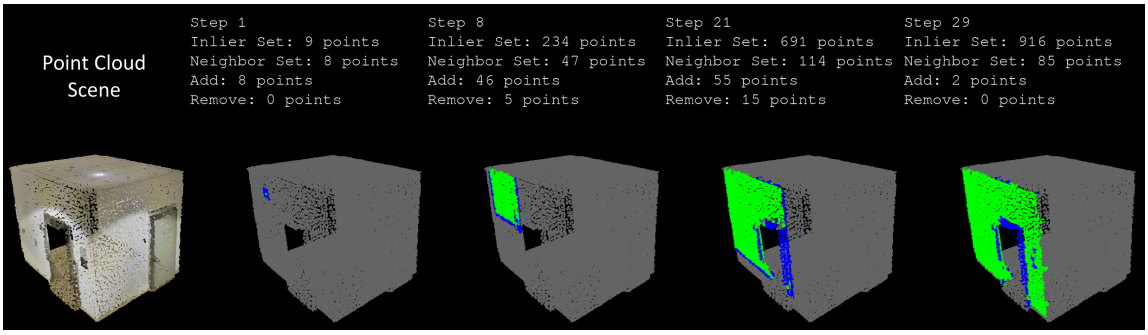


Figure 2.2: Visualization of the progression of region growing (green points are inlier set points and blue points are neighbor set points).

obtained by tuning on the S3DIS validation dataset. Further analysis of the effect of these hyperparameters will be given in the Experimental Results section.

2.2.2 Region growing simulation and training

Training data for LRGNet is obtained through region growing simulations on the S3DIS dataset [5], which comes with ground truth instance labels. The S3DIS dataset consists of 6 different building areas, with a total of 272 rooms and an average of 38 object instances in each room. The simulated region growing process is carried out by taking the labelled point cloud of a room and performing region growing in a carefully controlled manner from randomly selected seed points. At each intermediate stage of region growing, the inlier set and neighbor set can be computed and the ground truth *add* and *remove* masks can also be determined since which points have the same instance labels as the seed point are known in advance. These sets of inputs and corresponding outputs are stored in batches and used to train LRGNet in an offline process.

To help LRGNet generalize better to unseen data, artificial errors are added to the simulated region growing process. This is achieved by randomly adding and removing points incorrectly from the current region based on the mistake probability α. To ensure that the region growing process is able to converge to the current set of points, the mistake probability is gradually decreased by 0.01 after each step. In addition, data augmentation is also carried out to increase the amount of training data available. Data augmentation is achieved

by: (i) randomly flipping the *x* and *y* axes, (ii) randomly rotating the scene in increments of 90°, and (iii) randomly adjusting the initial mistake probability, α to a value between 0.2 and 0.4 (this is to simulate different levels of noise during the region growing process). In total, the training dataset contains 1406516 points clouds and 63047 object instances after data augmentation.

The proposed LRGNet, as described in the previous section, is implemented in Tensorflow. The network is trained with the ADAM optimizer for 40 epochs with a learning rate of 0.001 and a batch size of 100. The loss function is computed as the sum of the binary cross-entropy loss for the *add* mask output and the binary cross-entropy loss for the *remove* mask output. Note that the loss term of the *remove* mask output is normalized by frequency to account for the fact that removed points occur much less frequently than added points.

2.2.3 Region growing at inference time

Figure 2.2 shows a visualization of the progression of region growing for a wall object at inference time, demonstrating the number of points that are added and removed at different stages of the process. In addition, Algorithm 1 summarizes the procedure of applying the proposed region growing method to output instance labels L given an input point cloud P. The region growing process consists of starting from a seed point and then iteratively adding relevant points from the neighbor set and removing incorrect points from the inlier set until the termination condition is reached. The *add mask* and *remove mask* are obtained from the output of a trained LRGNet, described in the previous section. In this study, the termination conditions are as follows:

- The current region does not expand for two consecutive steps
- The set of points to be added is predicted to be empty
- There are no unassigned neighbor points remaining

These termination conditions are designed to halt the region growing process when there

are no longer useful actions to be taken or when there are oscillatory situations where points are added, removed at the next step, and then immediately added again. When the termination condition is reached, all points in the current region Q_k are assigned the same object ID. Then, the region growing process starts over with a new seed point with a new object ID.

Several heuristic methods are applied to improve the segmentation results. An intelligent seed selection process [42] is used where instead of randomly selecting the initial seed point, the seed point is selected as the remaining unlabelled point with minimum curvature value. In addition, a cutoff threshold [36][80] is applied to point cloud segments that are too small (i.e. less than 10 points) to avoid oversegmentation. In cases where the region growing output results in a segment that is too small, points in that segment will be relabelled based on the nearest neighbor point among points that are already labelled.

Algorithm 1: Region Growing

Input: PointCloud, P
Output: InstanceLabels, L
$\phi \equiv$ empty label ;
$L = \{\phi, \forall p \in P\}$;
$obj\_id = 1$;
while $\phi \notin L$ **do**
  p_{seed} = unlabeled point with min curvature;
  $k = 0$;
  $Q_k = \{p_{seed}\}$;
  while *true* **do**
    $inliers = sample(Q_k, I)$;
    $neighbors = sample(\{p : |p - q| < \delta, p \in P, q \in Q_k, L[p] = \phi\}, J)$;
    $add, remove = net(inliers, neighbors)$;
    if *termination condition* **then**
      $L[Q_k] = obj\_id$;
      $obj\_id = obj\_id + 1$;
      break;
    else
      $Q_{k+1} = Q_k - inliers[remove] + neighbors[add]$;
      $k = k + 1$;

2.2.4 Local search to optimize region growing

To further correct for errors in the region growing process, this research employs different local search techniques to iterate through different possibilities of region growing from a given seed point. This is relevant since the proposed region growing method is probabilistic in nature: points in the inlier set and neighbor set are randomly sampled at each step and the learnable function f also returns a probability distribution over the points to be added or removed. The local search techniques that are considered in this research are (i) random restart from the same seed point while probabilistically sampling points to be added and removed and (ii) beam search [81], where multiple states for the current region are simultaneously stored and expanded, limited to a fixed number of states after each expansion step. On the other hand, the optimality criterion that are considered in this research are (i) Maximum Likelihood (ML): sum of the log-likelihoods from each step and (ii) number of points (NP): number of points in the final point cloud region, based on the intuition that regions with more points are usually more complete. The region growing outcomes are separately stored and at the end of the local search process, the region growing outcome corresponding to the highest value for the optimality criterion will be used as the final result. In the Experimental Results section, these local search techniques will be compared against the greedy inference technique of directly applying the *add* and *remove* masks with a cutoff threshold of 0.5.

2.3 Experimental Results

2.3.1 Generalization of segmentation performance across datasets

Performance validation of the proposed LRGNet method is carried out using the metrics of normalized mutual information (NMI), adjusted mutual information (AMI), and adjusted rand index (ARI), as defined in [84], as well as precision (PRC), recall (RCL), and mean intersection-over-union (mIOU). The baseline methods used for comparison are (i) sim-

Table 2.1: Segmentation performance comparison on the ScanNet and S3DIS datasets

Method	NMI	AMI	ARI	PRC	RCL	mIOU
ScanNet (train) → S3DIS (test)						
Region growing	0.71	0.70	0.59	0.19	0.34	0.38
Rabbani et al. [82]	0.72	0.71	0.62	0.17	0.31	0.36
FPFH [83]	0.62	0.60	0.39	0.14	0.25	0.32
PointNet [36]	0.58	0.48	0.38	0.18	0.17	0.25
PointNet++ [76]	0.62	0.56	0.40	0.15	0.22	0.31
JSIS3D [38]	0.74	0.73	0.63	0.28	0.29	0.36
3D-BONET [31]	0.75	0.72	**0.68**	0.20	0.29	0.35
LRGNet	0.75	0.74	0.67	0.25	0.41	0.43
LRGNet + local search	**0.76**	**0.75**	**0.68**	**0.34**	**0.44**	**0.45**
S3DIS (train) → ScanNet (test)						
Region growing	0.62	0.60	0.44	0.17	0.23	0.30
Rabbani et al. [82]	0.64	0.62	0.49	0.13	0.24	0.32
FPFH [83]	0.53	0.51	0.28	0.10	0.14	0.26
PointNet [36]	0.57	0.51	0.40	0.08	0.13	0.26
PointNet++ [76]	0.63	0.57	0.47	0.15	0.21	0.32
JSIS3D [38]	0.57	0.56	0.31	0.15	0.13	0.22
3D-BONET [31]	0.59	0.54	0.34	0.10	0.13	0.24
LRGNet	0.69	0.67	0.54	0.25	**0.33**	**0.39**
LRGNet + local search	**0.69**	**0.68**	**0.56**	**0.31**	**0.33**	0.38

ple region growing based on thresholding, (ii) smoothness constraints from Rabbani et al. [82], (iii) Fast Point Feature Histogram (FPFH) descriptor [83], (iv) PointNet [36] (v) PointNet++ [76] (vi) JSIS3D [38] and (vii) 3D-BONET [31]. To be compatible with other methods, (iv) and (v) compute the instance labels by first predicting the class labels and then connecting neighboring points that have the same class labels.

Validation is carried out using two different datasets with different class definitions in order to fully examine the generalization performance of different point cloud segmentation methods. To be precise, the S3DIS dataset [5] and the ScanNet dataset [85] are used as training and test data alternatingly, as shown in Table 2.1. In Table 2.1, the performance metrics are obtained by averaging across all rooms/scenes of the dataset. Precision and recall are computed based on an IOU threshold of 50%.

Results in Table 2.1 show that the propsoed LRGNet method shows the best generalization performance compared to other segmentation methods, whether tested on the ScanNet dataset or on the S3DIS dataset. In addition, after applying a local search method (random restart with number of points as the optimality criterion) to optimize the region growing process, the performance of LRGNet further improves by 1% - 9%. The main reason that LRGNet is able to achieve good generalization performance is that it is class-agnostic, i.e. it does not make any assumptions about the classes of objects that are present in the scene. This factor comes into play when the training dataset and test dataset differ in terms of the type of scene and types of objects that are usually present in the scene. In this research, S3DIS consists of mostly office and hallway scenes whereas ScanNet consists of mostly living room and bedroom scenes. As a result, methods that are tuned to work with objects in a particular scene might not work with objects in different scenes.

Figures 2.3 and 2.4 show visualizations of the segmentation results on 3 different scenes of the S3DIS dataset and ScanNet dataset respectively. Results show that the proposed method is able to generate object segments that are closer matches to the ground truth whereas the other methods generate either too few or too many segments.

2.3.2 Ablation study

An ablation study was carried out to examine the difference in segmentation performance as different configuration changes and architecture changes are made to the proposed LRGNet. This study was conducted on the S3DIS dataset with Areas 1,2,3,4,6 used as training data and Area 5 used as validation data. Results in Table 2.2 show that having the remove mask mechanism, using intelligent seed selection, and performing feature normalization all play important roles in improving the segmentation performance. In addition, using the full suite of input features (i.e. XYZ + RGB + normal + curvature) as well as larger values of the I and J parameters tend to improve performance.

The effect of applying different local search methods to optimize the region growing

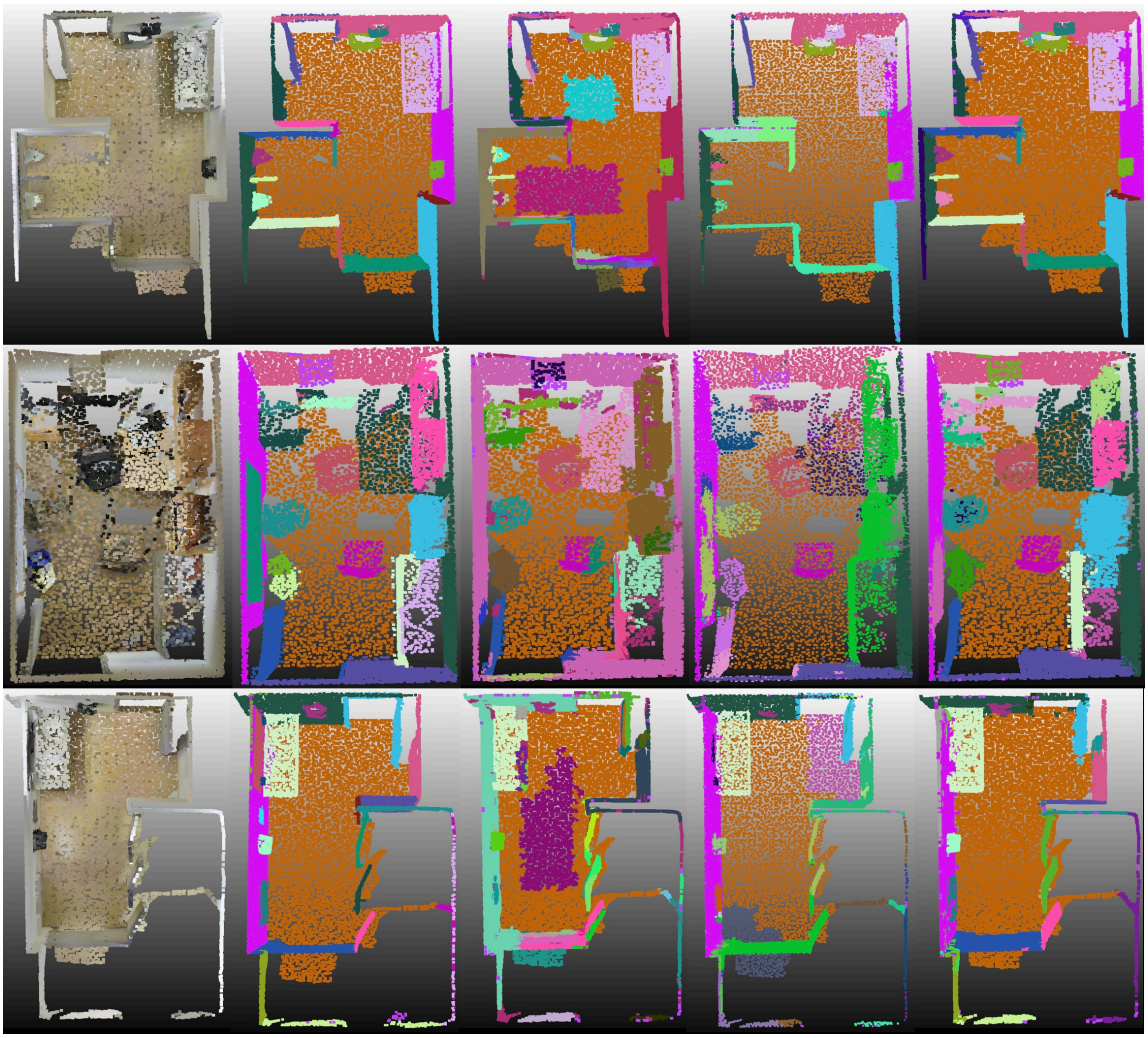


Figure 2.3: Segmentation results on 3 different rooms of the S3DIS dataset: (from left to right) (i) original RGB point cloud (ii) ground truth (iii) PointNet++ (iv) 3D-BoNet (v) proposed method

process of LRGNet is analyzed in Table 2.3. The results are obtained on the S3DIS dataset using the metrics of NMI, AMI, ARS, and average number of inference steps needed to segment one object instance. In order to ensure that each method has roughly the same number of computation steps, random restart is applied with 10 total restarts whereas beam search is applied with a branching factor of 3 and expansion factor of 3. Results in Table 2.3 show that random restart with the number of points criteria achieved the best performance overall. Note that applying the local search methods cause a significant increase in the number of computational steps required, even higher in proportion to the number of restarts

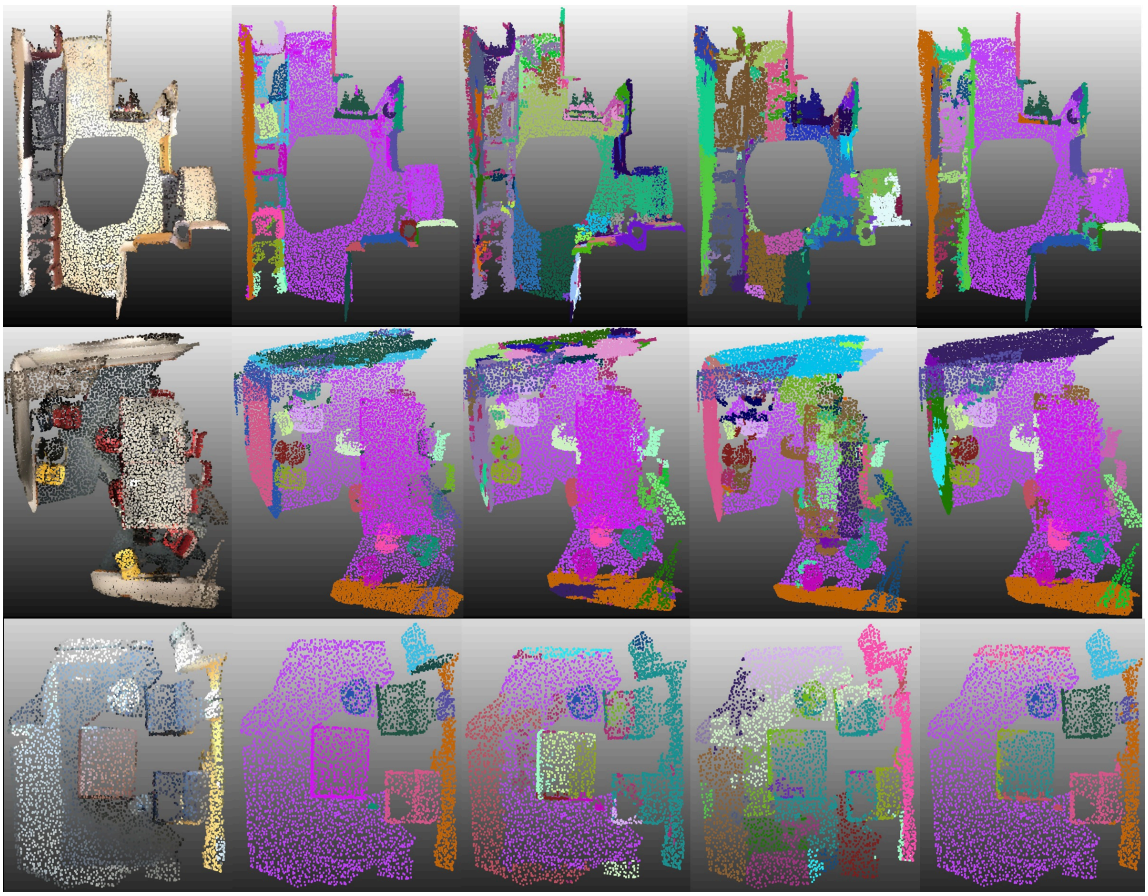


Figure 2.4: Segmentation results on 3 different scenes of the ScanNet dataset: (from left to right) (i) original RGB point cloud (ii) ground truth (iii) PointNet++ (iv) 3D-BoNet (v) proposed method

or branching carried. This is because local search introduces more variance to the length of an average region growing subroutine.

2.3.3 Discussion

The main advantages of the proposed method are that is can be used to segment objects of arbitrary shape, size, or class, and that it is more accurate and generalizable compared to existing methods. In addition, the process for generating training data is based on simulation, so large-scale data augmentation can be easily performed from a single annotated dataset.

One limitation of the proposed learnable region growing method is the high computa-

Table 2.2: Performance comparison of different configurations of LRGNet

Method	NMI	AMI	ARI	PRC	RCL	mIOU
no remove mask	0.78	0.75	0.73	0.51	0.41	0.42
no seed selection	0.81	0.77	0.76	0.40	0.55	0.53
no feature normalization	0.81	0.76	0.75	0.39	0.53	0.52
only XYZ	0.76	0.75	0.61	0.24	0.42	0.44
only XYZ+RGB	0.78	0.78	0.70	0.33	0.53	0.51
$I = 128$, $J = 128$	0.79	0.79	0.69	0.32	0.57	0.55
$I = 256$, $J = 256$	0.80	0.80	0.74	0.35	0.56	0.54
complete method	0.81	0.78	0.77	0.43	0.56	0.54

Table 2.3: Performance comparison of different local search methods

Method	NMI(%)	AMI(%)	ARS(%)	Avg steps
greedy	82±4	73±7	74±12	13.38
random restart - ML	82±5	78±7	77±12	190.06
random restart - NP	82±4	79±6	77±10	188.70
beam search - ML	82±5	78±7	77±12	159.79
beam search - NP	82±4	78±6	77±10	175.50

tional cost of having to run the deep network multiple times to segment each object. To enable real-time implementation, it may be necessary to reduce the network size or use techniques such as distillation to speed up network inference. Another possibility is to use LRGNet as a post-processing method for auto-annotating point cloud data. This will enable another faster network to be trained for the segmentation task based on the auto-annotated labels.

Another limitation is the lack of temporal consistency or temporal reasoning when performing inference with LRGNet. Currently, the input to LRGNet is conditioned on only the current state of the segmented object and not on previous states. Ideally, the region growing process should be temporally consistent from each segmentation state to the next. This could involve modifying the network architecture to incorporate Recurrent Neural Network (RNN) or Long-Short Term Memory (LSTM) layers.

CHAPTER 3

DAMAGE ASSESSMENT FROM POINT CLOUDS

Conventional methods for building damage assessment rely on manual inspection of damage indicators such as liquefaction, tilting or cracks [86][87]. The problem with manual inspection is that it is time-consuming, labor-intensive, subjective, and risky. Sensor-based damage assessment methods such as vibration testing [46], acoustic emission testing [48], and guided wave testing [51] can usually provide more objective measurements of the structural damage. However, the limitation of these sensor-based methods is that they require pre-installation of sensors and physical contact with the damaged structure which is impractical in high-risk or inaccessible regions of a disaster site. On the other hand, damage assessment can also be performed by observing the surface of damaged structures using a remote sensing approach by acquiring image [53] and point cloud data [69]. Image and point cloud data can efficiently capture the surface condition without physical contact, but the raw data is difficult to parse due to the high dimensionality. Although various learning-based architectures and datasets have been studied for this purpose, classifying damaged and deformed structures is still challenging due to the low availability and possible non-existence of ground truth datasets.

This research focuses on the problem of damage assessment in the point cloud domain since point clouds can directly represent 3D geometry and can also represent additional quantities such as color and reflectivity. Point clouds are easy to visualize and can be used to estimate multiple damage indicators such as tilt, drift, cracks in physical units (i.e. meters). To tackle the problem of automated damage assessment from disaster site point clouds, this research proposes a method for crack segmentation using unsupervised machine learning techniques. In particular, a deep neural network is used to compute a discriminative point feature embedding under a transfer learning framework. Moreover, an unsupervised learn-

ing technique based on anomaly detection is used to overcome the data deficiency problem for crack detection due to a lack of annotated data for cracks in point clouds. The proposed method is evaluated based on laser-scanned point clouds from (i) the 2015 Nepal earthquake and (ii) the Guardian Centers disaster preparedness and tactical training center. The performance of the proposed point feature embedding is compared against that of other feature representations such as color, intensity, normal vector, curvature, Fast Point Feature Histogram (FPFH), and PointNet++. In addition, the performance of the anomaly detection algorithm is compared against that of other algorithms for segmenting the damaged regions such as K-means clustering and mean-shift clustering.

3.1 Structure Level Damage Assessment

The acquired point cloud data will be used to perform structure-level damage analysis. A common method to perform point cloud-based damage analysis is to detect the deviation between the as-damaged point cloud and the as-designed model. However, this requires a detailed as-designed model that closely matches the damaged building which may not be available for all buildings. Thus, for cases where the design model is not available or where the damage can only be detected through drift measurements, a computational geometry approach is proposed to reconstruct the post-damage building status directly from 3D point cloud data. In particular, inter-story drift estimation and tilt estimation are performed since they represent critical damage indicators for structural health monitoring. In this section, the experiments are carried out on point cloud data of a damaged concrete structure at the Hanyang University ERICA Structures Lab.

3.1.1 Inter-story Drift Ratio Estimation

The first step is a floor segmentation procedure that organizes the point cloud into multiple subsections corresponding to each building story. The raw point cloud is partitioned into floor/ceiling points and non-floor/ceiling points for each building story. The point cloud

is sorted according to ascending z-coordinate, and the points is assigned to one of discrete z-coordinate bins. For each z-coordinate bin, the point density is calculated and plotted on a graph. We will classify the points into floor/ceiling points and non-floor/ceiling points based on the property that the point cloud is dense with respect to the z-coordinate in floor or ceiling areas and sparse for the area in between. The classification process is based on the local point density distribution for each z-coordinate bin. For example, if the point density is greater by 50% compared to the median of the surrounding 10-bin window, then all the points in that bin are classified as floor/ceiling points, else the points are classified as non-floor/ceiling points. This procedure allows the point cloud regions that separate out each building story to be extracted from the scene.

The next step is to parameterize each building story as a plane model using Random Sample Consensus (RANSAC) algorithm for plane detection. The RANSAC algorithm is chosen because it is robust with respect to noisy data and can work even when the data has multiple underlying modes. After a reference plane is selected, the drift for each story is measured as the deviation from the nominal position. Figure 3.2 shows an example of the inter-story drift estimation from the point cloud of a damaged concrete structure (Figure 3.1). In this example, the RANSAC algorithm will detect the top, middle, and bottom planes. Then, the drifts of the middle and top planes will be measured by using the bottom plane as the reference plane. For each story, the drift will be quantified in mm in both the x-direction, y-direction, as well as the total horizontal drift.

3.1.2 Tilt Estimation

In addition, the tilt of each building story will be measured in a similar manner. The normal vector of each detected plane will be compared using the bottom plane as the reference plane and the tilt angle will be measured with respect to the difference in normal vector direction. For example, in the concrete structure in Figure 3.3, the tilt of the middle plane is estimated to be 1.21 degrees (1.09 degree in the x-axis and 0.51 degrees in the y-axis)

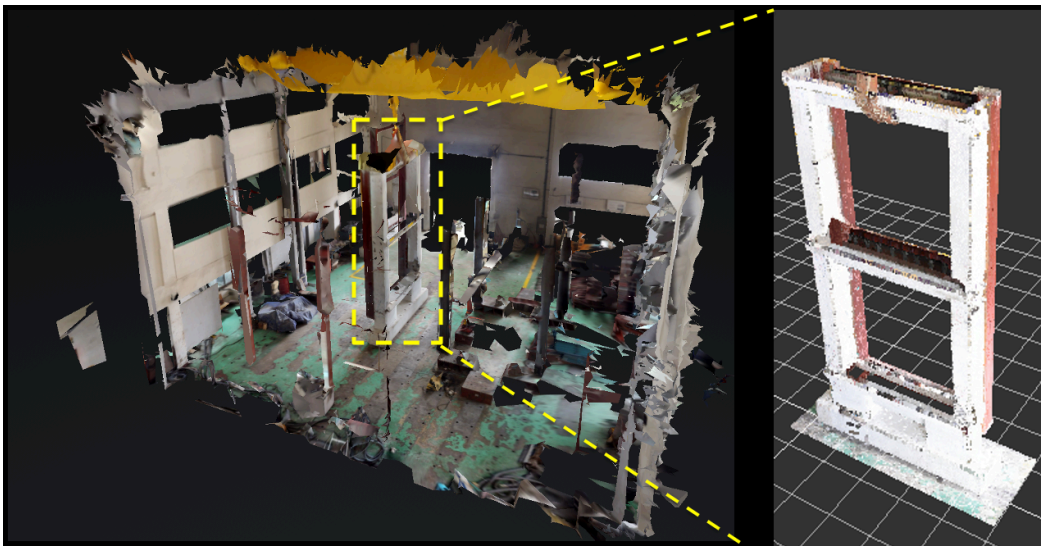


Figure 3.1: Post-damage model of the structure captured by a 3D scanner after an external force is applied

and the tilt of the top plane is 1.39 degrees (1.21 degrees in the x-axis and 0.69 degrees in the y-axis).

3.2 Surface Level Damage Assessment

3.2.1 Feasibility of Damage Assessment based on Crack Segmentation

Based on existing literature, there are multiple sensing methods available to assess structural damage such as cracks after disaster events. However, several of these sensing methods rely on pre-installation of sensors or physical contact with the damaged structure which is impractical in a disaster relief scenario. Thus, in this research, a remote sensing approach based on point cloud data is proposed that can be deployed on mobile robots for non-intrusive damage assessment.

The Federal Emergency Management Agency (FEMA) guidelines for evaluation of earthquake damaged concrete show that observed damage such as cracks are important indicators for building component behavior in post-earthquake scenarios [88]. For example, even a one-eighth inch crack in a wall panel could indicate that the structure is on the verge of brittle shear failure [88]. In general, seriousness of structural damage is tied to its

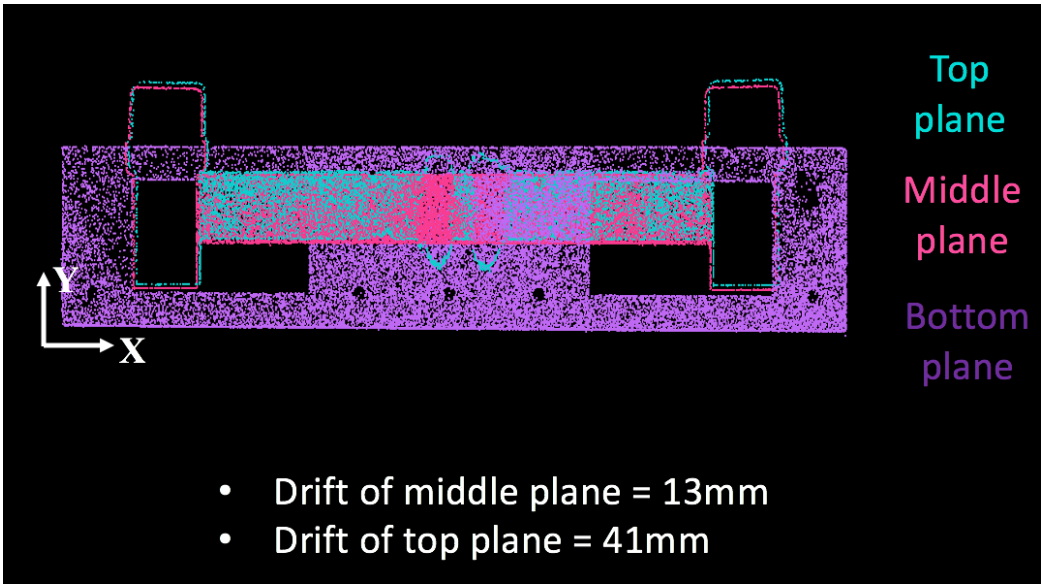


Figure 3.2: Inter-story drift estimation of a damaged concrete structure based on RANSAC plane detection

observability. More specifically, the maximum crack width is indicative of the maximum reinforcement strain and is a good indicator for damage severity. Thus, the proposed point cloud-based approach is advantageous since the physical crack geometry can be directly estimated in 3D from point cloud data.

One concern with this approach is that inspecting external structural damage (e.g. surface cracks) may not reflect the internal structural damage that cannot be observed. However, current post-earthquake inspection guidelines show that visual inspection of external cracks is an established practice for post-earthquake structural safety evaluation [89][88]. Generally, earthquake damage to concrete and masonry walls is visible on the exposed surface[88]. Besides, in concrete structures, internal damage is usually caused by thermal attacks such freezing or fire events, whereas earthquakes usually cause stress forces that result in external damage [90]. In certain situations, earthquakes may cause hidden damage such as a buckled rebar. Even in these situations, it is still cost-prohibitive and even dangerous to intrusively test every structure in a building for hidden damage [88]. Thus, the proposed point cloud-based approach is a reasonable solution for non-intrusive damage assessment.

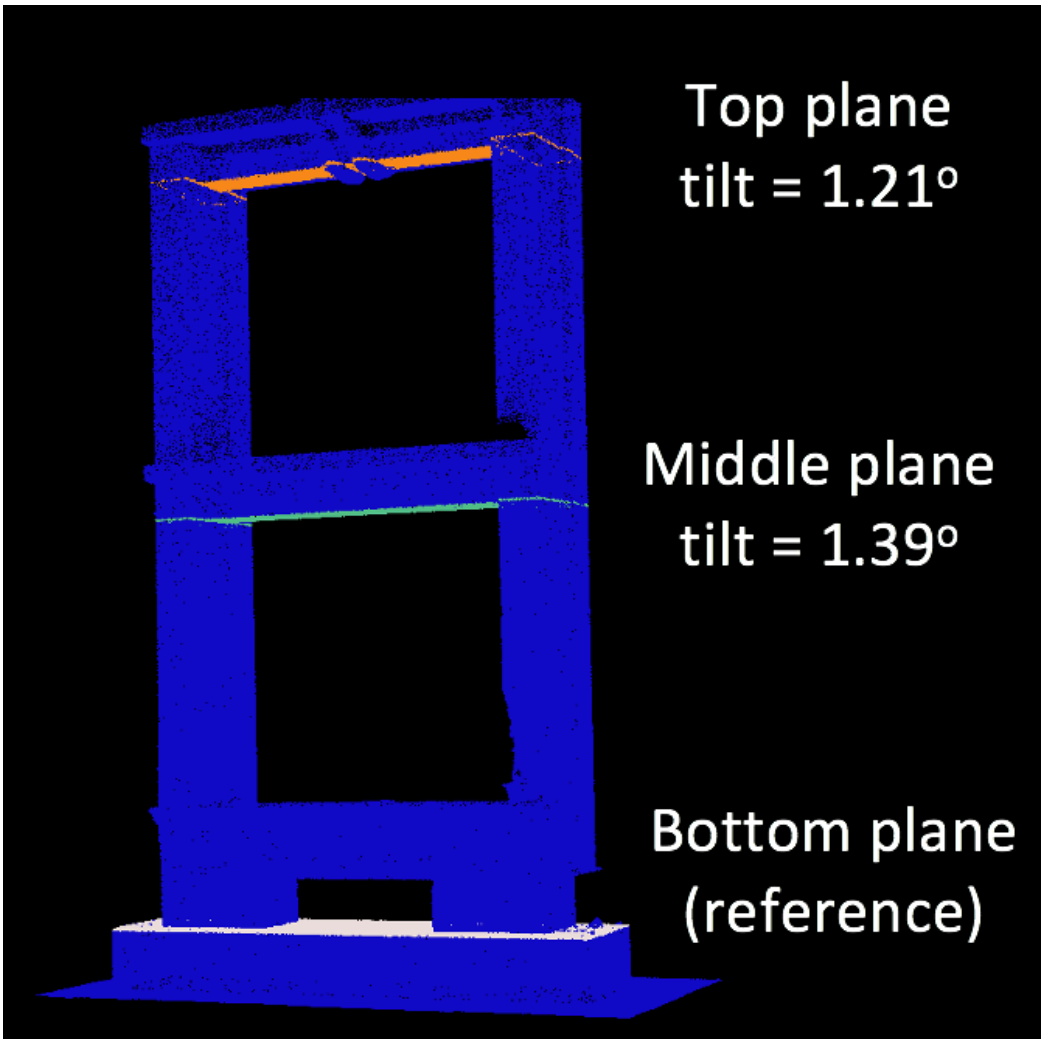


Figure 3.3: Inter-story drift estimation of a damaged concrete structure based on RANSAC plane detection

3.2.2 Building Component Extraction

The first step in detecting damaged building elements is to isolate specific building components of interest. This study uses a semi-automated framework for building component extraction from a raw point scene (shown in Figure 3.4). First, a region growing segmentation algorithm is applied to group neighboring points together into clusters such that each cluster contains points from a unique building component. The region growing algorithm works by considering the normal vector and color features from each point and joining them if they have similar features and are within a neighborhood of 0.1m. Figure 3.5 shows a vi-

sualization of the region growing segmentation results for two different scenes where each point cloud cluster is shown in a different color. Next, the building components of interests are extracted by manually indicating the dimensions for a specific type of building component. For example, the concrete columns from the scene in Figure 3.5a can be extracted by filtering for point cloud segments that match dimensions of 0.4m x 0.4m x 2.6m.

Figure 3.4: Visualization of overall framework for building component extraction (e.g. columns)

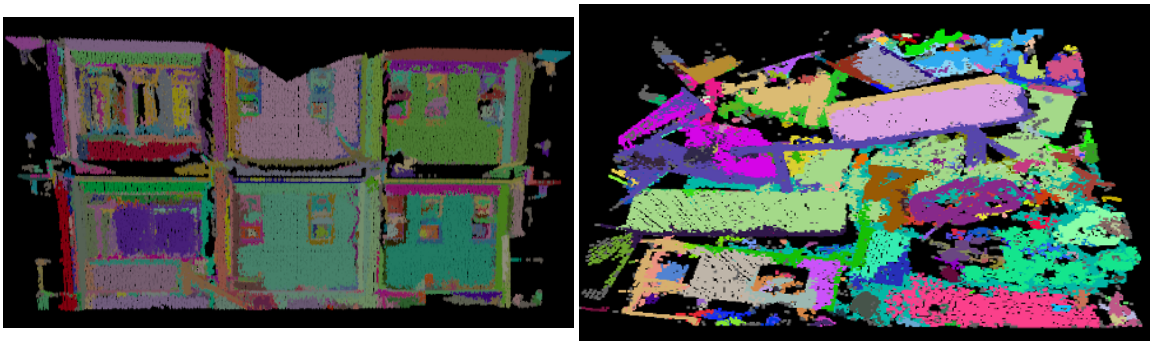


Figure 3.5: Region growing segmentation results for (a) concrete column scene and (b) concrete slab scene

3.2.3 Point Feature Embedding Computation

Once the building components of interest are extracted, point features are computed for each point in that building component. In any machine learning framework, features are

important to describe the characteristics of a data sample and are used as a basis for differentiation between samples from different classes. In this study, point features are used to differentiate between cracked regions and undamaged regions in point cloud data. Simple examples for point features are measurements that can be obtained directly from the data acquisition device such as color, intensity and XYZ coordinates. Other examples for point features are features derived from the local geometry such as the normal vector, which describes the orientation of the local surface, and the curvature, which describes whether the local surface is flat or curved.

On the other hand, this study uses the concept of a feature embedding to represent discriminative information about a point. A feature embedding is a transformation of the original space of raw features into a new feature space that is more semantically meaningful. The benefit of having a point feature embedding compared to the raw features is that the embedding is able to capture high-dimensional information in the local geometry and summarize that information succinctly in a feature vector. There exist feature descriptors such as Viewpoint Feature Histogram (VFH) [91], Fast Point Feature Histogram (FPFH) [92], and Spin Images [93] that can compute geometrical features for each point. However, these feature descriptors are manually designed and not data driven. There are also deep neural nets for point clouds such as PointNet [94], PointNet++ [95], and Recurrent Consolidation Units [96] that can predict feature embeddings but these embeddings are tuned more for classification instead of segmentation.

This research proposes a point-based deep neural network that is used to extract discriminative point features using the geometry of the local point neighborhood. Since annotated data from disaster sites are difficult to obtain, this research uses a network that is trained on regular building scenes and applied on disaster site scenes under a transfer learning scheme. The network is trained to distinguish different types of points without being conditioned to specific types of objects so that it can generalize to new scenes. In this study, the network is trained using the Stanford 3D Indoor Spaces (S3DIS) dataset [97]

which contains common building elements such as walls, floors, columns, and beams.

Figure 3.6 shows the neural network architecture used to compute CrackEmbed, point feature embeddings in this study. The input to the network is a point cloud with N points represented by a N x 6 matrix, where each row is a point with 6 features which are the XYZ coordinates and RGB color. The network uses a context pooling module to extract geometrical information from neighboring points. In particular, for each of the N input points, a set of M neighboring points are sampled at a radius of 0.01m to be used as context information. In this study, N and M are set to 256 and 50 respectively after hyperparameter tuning. The input features of the neighboring points are processed using a series of convolutional layers and finally pooled into a 200-dimensional feature vector. This contextual feature vector is concatenated back to the 6-dimensional vector of raw features for each input point. Next, the concatenated features are processed with two more convolutional layers to form a 50-dimensional feature embedding, which means that each input point is represented with a set of 50 features. This feature embedding is trained using the triplet loss function [98], which aims to optimize the weights of the network such that points that are semantically similar are close together in feature space whereas points that are points that are semantically dissimilar are further apart in feature space. During the training process, triplets of points are sampled from the training dataset with each triplet having two points with the same instance label and one point with a different instance label. The optimizer then updates the network weights so that the feature embeddings of the similarly labeled point pair are closer together compared to the feature embeddings of the differently labeled point pair. This process works under the assumption that points that originate from the same object instance have similar geometry and should have similar feature embeddings whereas points that originate from different object instances should have dissimilar feature embeddings. Since this triplet loss embedding is trained to separate out points with different types of geometry, it can be transferred to the crack segmentation task because cracked points are likely to have different geometry compared to the surrounding points.

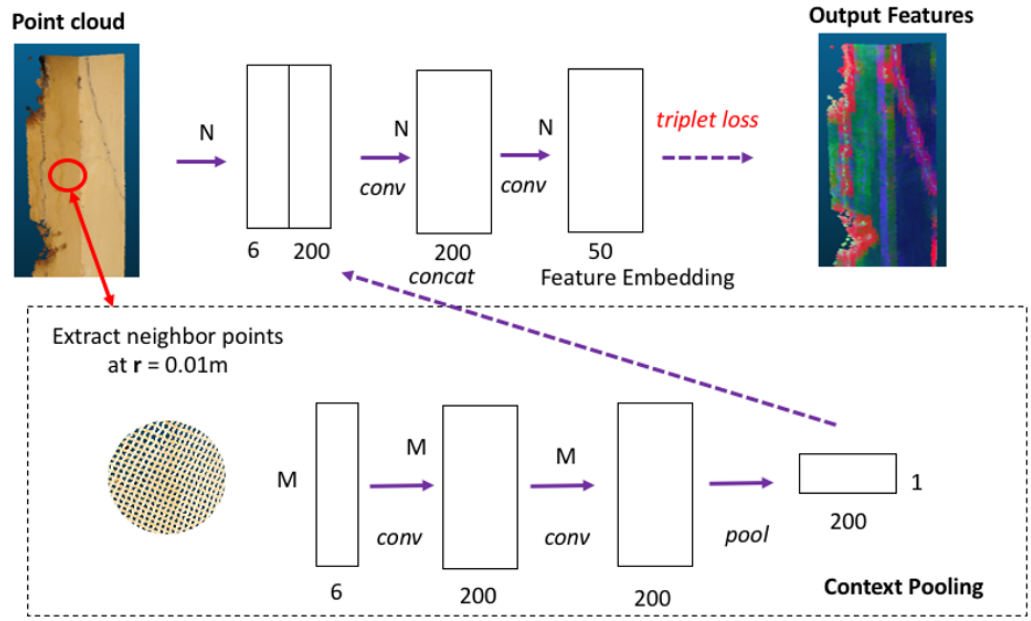


Figure 3.6: Network architecture to compute CrackEmbed point feature embeddings based on the triplet loss function

3.2.4 Anomaly Detection

After the point features are computed, an unsupervised learning algorithm is applied to separate out damaged points from non-damaged points based on their distribution in feature space. This process works based on the intuition that non-damaged points have features that appear at a high frequency whereas damaged points that features that deviate from normal and appear at a lower frequency. Thus, an unsupervised learning algorithm that can analyze the feature space and identify anomalies will be able to function as a damage detection algorithm. Using unsupervised learning algorithms is advantageous compared to supervised learning algorithms in the context of disaster site data because they do not require training data which is difficult to acquire and annotate. Unsupervised learning is also a more robust solution compared to applying a fixed threshold on the features values since it can adapt to the feature distribution.

This study uses several clustering methods as baseline algorithms for crack segmentation since clustering is a commonly used unsupervised learning algorithm. One such

algorithm is K-means clustering [99], which divides the data into K clusters by iteratively computing the cluster centers and assigning data points to the closest cluster. Figure 4 shows a visualization of K-means clustering used to segment out the cracked region. To show the distribution of points in feature space, Principal Component Analysis (PCA) is used to reduce the feature dimensionality to two dimensions so that it can be plotted. As shown in Figure 3.7, the points are color-coded into 5 different colors with corresponds to the K=5 clusters determined by K-means clustering. From these clustering results, the cluster with the smallest number of points is assumed to be the outlier and the points in that cluster are returned as the cracked region. Another clustering method is Gaussian Mixture Models [100], which considers both the mean and covariance of each cluster and also uses soft assignment of data points to clusters. A third clustering method considered in this study is mean-shift clustering [101], which aims to discover a set of centroids and updates them so that they represent the mean of their respective regions.

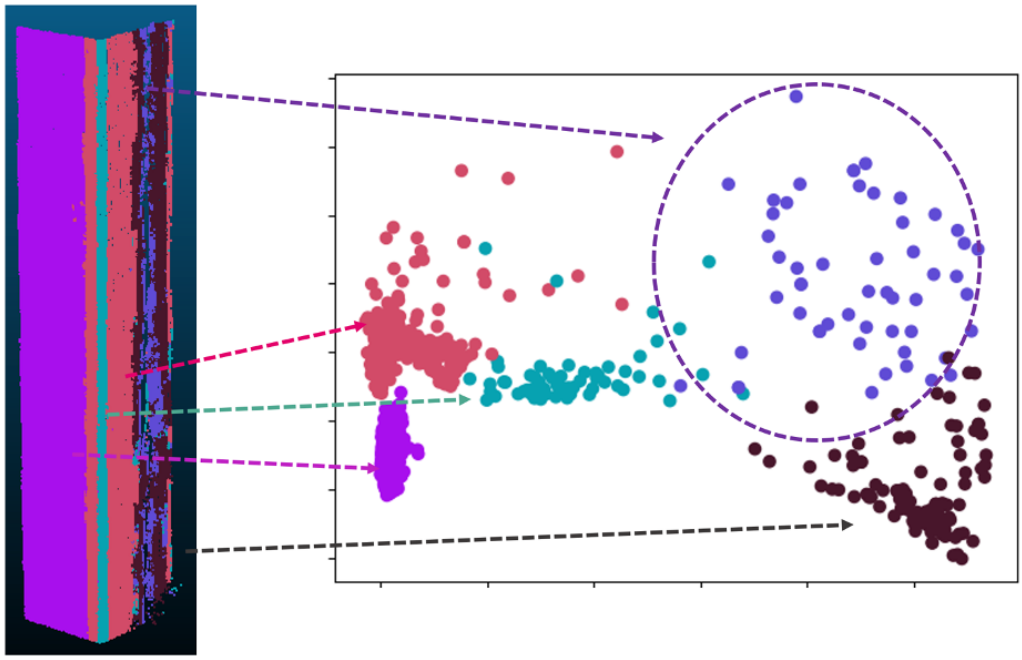


Figure 3.7: Visualization of K-means clustering used to segment the points belonging to the cracked region in the feature space

In order to specifically segment out anomaly points, this study uses anomaly detection

methods which are also known as novelty detection or outlier detection methods. This study considers 4 different anomaly detection algorithms which are (i) isolation forest [102], (ii) one-class SVM [103], (iii) robust covariance [104], and (iv) local outlier factor [105]. Isolation forest constructs a decision tree structure over the data points and uses the number of splits required to isolate a point as a measure of normality. On the other hand, one-class SVM works by constructing a frontier over the regular data points and defines anomalies as points that lie outside the frontier. Robust covariance assumes that the regular data points are Gaussian distributed and solves for the covariance of these data points so that the normality can be measured based on the distance to the central mode. Finally, local outlier factor measures the normality of each data point by comparing the local density to the average density of its nearest neighbors, Figure 3.8 shows a visualization of the isolation forest algorithm used to segment out the crack region. Each point is color-coded by its normality score where red indicates points that are likely to be common points whereas blue indicates points that are likely to be anomaly points. As shown in the figure, the points that have low normality score (i.e. anomaly points) are correlated with points in the cracked region of the original point cloud.

3.3 Experimental Results

To thoroughly evaluate the effectiveness of the proposed crack segmentation method, this research makes use of two laser-scanned point cloud datasets: (i) a maternity hospital that was damaged during the 2015 Nepal earthquake, and (ii) a damaged concrete structure at the Guardian Centers disaster training facility. The Nepal dataset is taken from a publicly-available repository [106] whereas the Guardian Centers dataset is collected by the authors. The Nepal dataset (Figure 3.9a) contains 7 cracked concrete columns each measuring about 0.4m x 0.4m x 2.6m whereas the Guardian Centers dataset (Figure 3.9b) contains 2 cracked concrete slabs each measuring about 5m x 3m x 0.5m. The Guardian Centers dataset has several damaged slabs but only 2 were selected for evaluation because the others either do

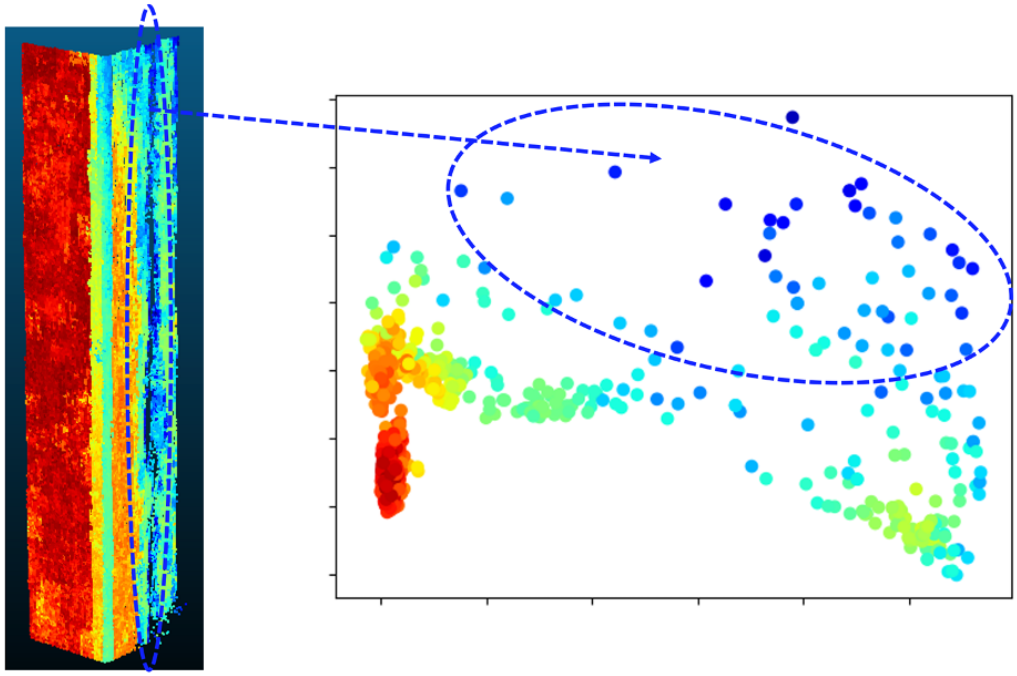
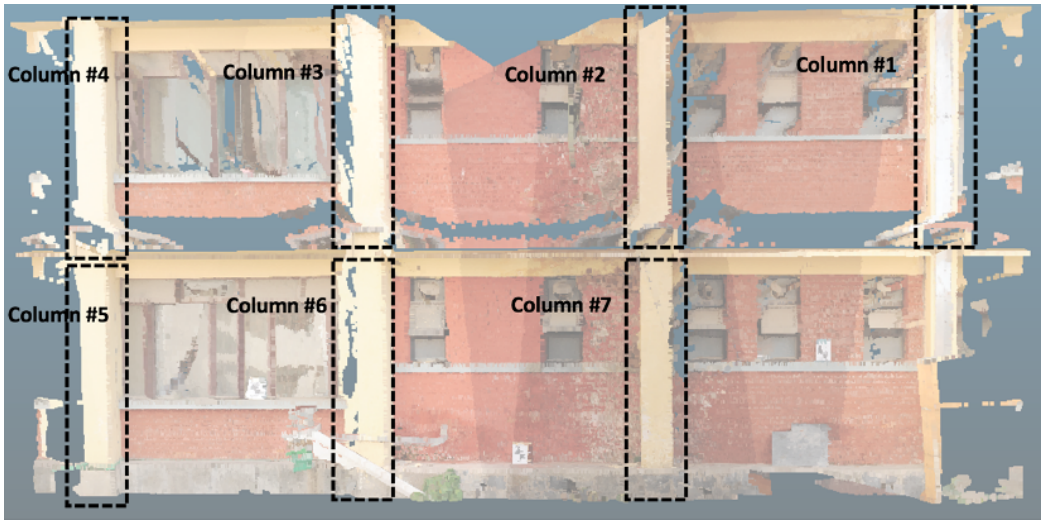


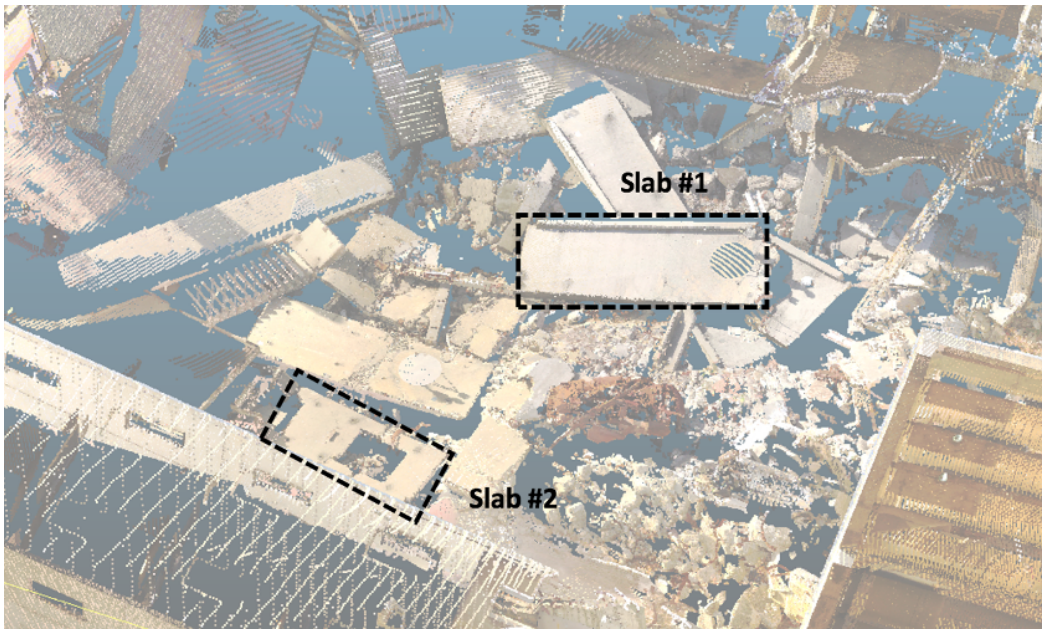
Figure 3.8: Visualization of Isolation Forest used to segment the points belonging to the cracked region in the feature space

not have visible cracks or are overly deformed. In addition, the slabs had some shadow artifacts and outlier points due to registration errors which were manually removed as a pre-processing step. To create ground truth annotations of the cracked regions for each building structure, the authors manually labelled the point cloud data with segmentation masks using the CloudCompare software [107].

The crack segmentation accuracy is measured using the point-wise precision and recall metrics, which are determined by how well the predicted segmentation mask matches the ground truth segmentation mask. In addition, the F1-score is used to represent the combined information from both precision and recall metrics (since the F1-score is the harmonic mean of precision and recall). Finally, the length and width of the crack is estimated by taking the extent of all the points within the predicted segmentation mask. Then, the length error and width error are used to measure the absolute error in the predicted dimensions of the overlapped segmentation mask. Note that the length and width error are measured at the longest and widest part of the crack respectively, so the length and width



(a) Nepal dataset containing damaged columns



(b) Guardian Centers dataset containing damaged slabs

Figure 3.9: Different datasets for damaged structural components

error could be zero even though the predicted segmentation mask is not perfect.

Tables 3.1 and 3.2 show the performance comparison of different feature representations on the Nepal dataset and the Guardian Centers dataset respectively. Crack detection was performed on damaged columns in the Nepal dataset and on slabs in the Guardian Centers dataset. The proposed CrackEmbed was compared with several baselines including raw features such as (i) RGB color and (ii) intensity, geometric features such as (iii) normal and (iv) curvature, and computed features such as (v) Fast Point Feature Histogram (FPFH) [92] and (vi) PointNet++ [95]. In this performance comparison, the isolation forest algorithm is used for anomaly detection. Results show that the CrackEmbed achieved the highest precision, recall, and F1-score as well as the lowest length error and width error. Figures 3.10 and 3.11 show the visualization of the crack segmentation results using different feature representations where the predicted crack region is highlighted in yellow. The figures show that there are still many false positive points that are segmented across all the methods but the CrackEmbed method resulted in the overall closest match to the ground truth.

Table 3.1: Performance comparison of different feature representations on the Nepal dataset

Feature	Precision	Recall	F1-score	Length error (m)	Width error (m)
RGB color	0.32	0.44	0.36	0.001	0.011
Intensity	0.17	0.33	0.21	0.003	0.004
Normal	0.26	0.37	0.30	0.001	0.012
Curvature	0.29	0.42	0.34	0.001	0.015
FPFH	0.21	0.10	0.13	0.020	0.018
PointNet++	0.15	0.10	0.12	0.002	0.045
CrackEmbed	0.41	0.70	0.50	0.000	0.004

Table 3 shows a performance comparison of different unsupervised machine learning algorithms used to separate out the cracked points. These include anomaly detection methods such as (i) isolation forest [102], (ii) one-class SVM [103], (iii) robust covariance [104], and (iv) local outlier factor [105]. Clustering methods such as (i) K-means clustering [99],

Table 3.2: Performance comparison of different feature representations on the Guardian Centers dataset

Feature	Precision	Recall	F1-score	Length error (m)	Width error (m)
RGB color	0.72	0.57	0.63	0.027	0.021
Intensity	0.60	0.61	0.60	0.006	0.008
Normal	0.52	0.40	0.45	0.146	0.148
Curvature	0.47	0.44	0.44	0.049	0.040
FPFH	0.61	0.08	0.14	0.083	0.086
PointNet++	0.60	0.34	0.41	0.052	0.068
CrackEmbed	0.69	0.75	0.71	0.004	0.004

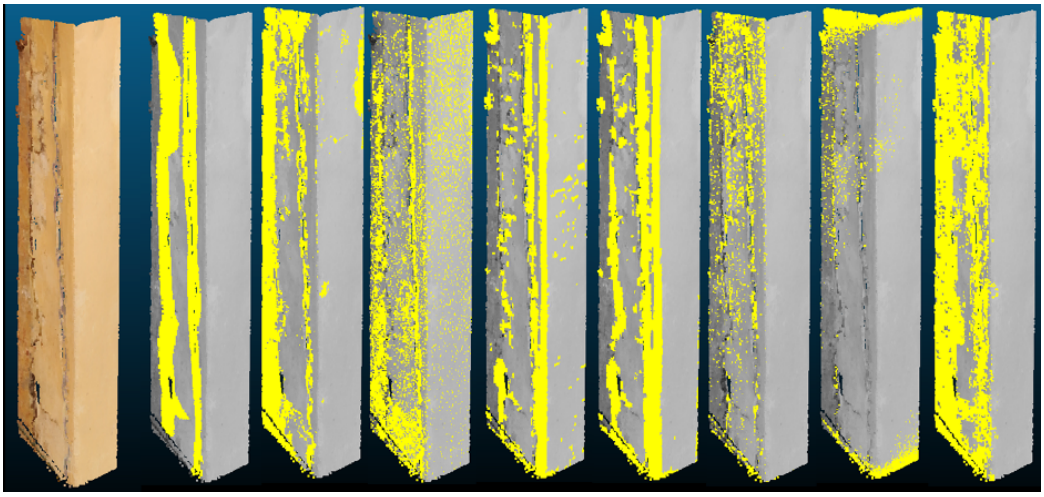


Figure 3.10: Visualization of column crack segmentation results with different feature representations: (a) input point cloud (b) ground truth (c) RGB color (d) intensity (e) normal (f) curvature (g) FPFH (h) PointNet++ (i) CrackEmbed

(ii) Gaussian Mixture Model [100], and (iii) mean-shift clustering [101] are also included as baseline methods for comparison. In this study, the anomaly detection and clustering algorithms are implemented using the *scikit-learn* [108] library. Results in Table 3.3 show that the anomaly detection methods, with the exception of local outlier factor, outperform the clustering methods in general. Among the anomaly detection methods, the isolation forest method achieved the highest recall and the lowest length error and width error. Whereas, the robust covariance method achieved the highest precision and F1-score.

Figure 3.12 shows the end result of damage detection where the segmented cracks from each column can be mapped back to the original point cloud scene. In this way, the affected

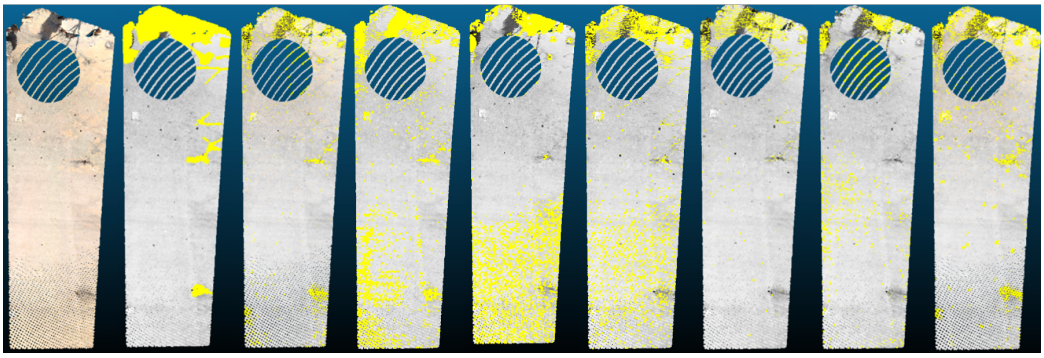


Figure 3.11: Visualization of slab crack segmentation results with different feature representations: (a) input point cloud (b) ground truth (c) RGB color (d) intensity (e) normal (f) curvature (g) FPFH (h) PointNet++ (i) CrackEmbed

Table 3.3: Performance comparison of different unsupervised algorithms on the Nepal dataset

Method	Precision	Recall	F1-score	Length error (m)	Width error (m)
Anomaly detection					
Isolation Forest	0.39	0.76	0.50	0.000	0.004
One-class SVM	0.38	0.62	0.41	0.027	0.010
Robust covariance	0.59	0.50	0.52	0.040	0.010
Local Outlier Factor	0.13	0.12	0.12	0.018	0.008
Clustering					
K-means	0.58	0.36	0.43	0.287	0.010
Gaussian Mixture Model	0.27	0.15	0.19	0.386	0.020
Mean-shift	0.34	0.24	0.27	0.032	0.012

building can be described in terms of the precise location of each crack as well as the length and width of each crack. This information can be used for further processing by the disaster relief team for structural analysis and risk analysis.

3.4 Performance Analysis

3.4.1 Analysis of different feature representations

The effectiveness of different point feature representations when used for the task of crack segmentation can be analyzed based on the results in Tables 3.1 and 3.2. Overall, the proposed CrackEmbed achieved the best performance across both the Nepal dataset and

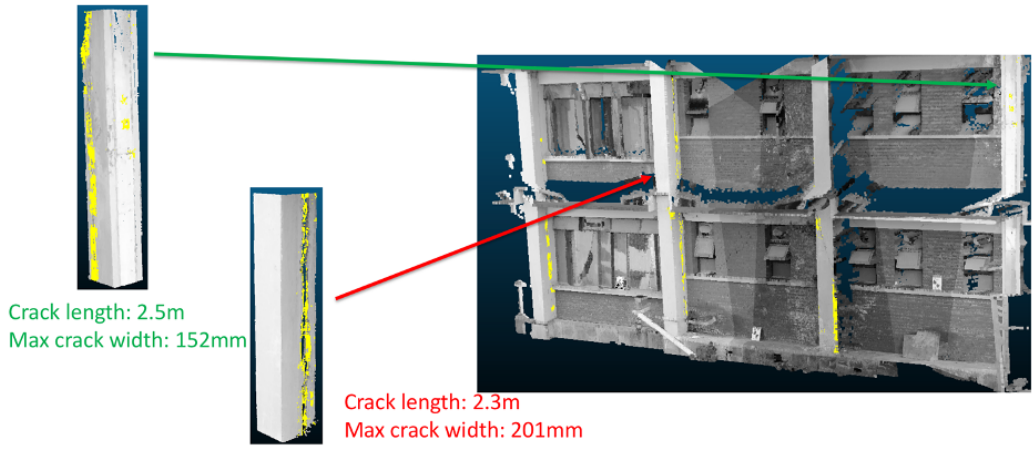


Figure 3.12: Visualization of crack segmentation results for concrete columns mapped back to the original scene

the Guardian Centers dataset. This demonstrates that the proposed embedding is able to successfully capture the relevant information in point cloud data and represent them as feature vectors that are useful for segmentation. The RGB color, normal, and curvature features also achieved reasonable performance whereas the FPFH and PointNet++ features resulted in poor performance. This is because FPFH is designed for point cloud registration whereas PointNet++ is designed for point cloud classification and the computed features may not be suitable for segmentation.

The segmentation performance can also be compared in terms of the qualitative results in Figures 3.10 and 3.11. One caveat of this comparison is that the ground truth outline of a crack is difficult to define in certain situations where the outline of the crack is not visually prominent. However, the performance can still be compared by examining the frequency of false positive points in regions that clearly do not have cracks. The RGB color, intensity, and CrackEmbed features have false positives that are mostly caused by shadow effects in the point cloud. Whereas, the normal, curvature, FPFH, and PointNet++ features have false positives that occur around the edges of the building component. Other confounding factors include uneven surface and uneven resolution of the point cloud caused by incomplete laser scanning. Especially in the Nepal dataset, the performance of the intensity feature is

negatively affected by false positives, possibly since the intensity measurement could vary depending on the laser incident angle as well as the distance between the scanned structure and the laser scanner.

3.4.2 Analysis of different unsupervised algorithms

Table 3.3 analyzes the segmentation performance comparison of different unsupervised algorithms on the Nepal dataset, including anomaly detection algorithms and clustering algorithms. Among the anomaly detection algorithms, the isolation forest and robust covariance algorithms performed the best overall when considering all the metrics. One-class SVM achieved a reasonable performance whereas the local outlier factor algorithm performed poorly, especially in terms of the F1 score. This could be due to the way the cracked points are distributed in feature space, where they are isolated from other regions, but not sparse enough such that the local outlier factor is significant. On the other hand, among the clustering algorithms, K-means clustering achieved the best performance whereas the other two algorithms did not perform as well. One downside of using a clustering algorithm is that the hyperparameters such as the number of clusters (for the case of K-means clustering and Gaussian Mixture Model) and the bandwidth (for the case of mean-shift clustering) have to be carefully selected to achieve good performance. A clustering algorithm usually creates multiple clusters, one of which is assigned to the anomaly region, whereas an anomaly detection algorithm directly separates the anomaly region and the non-anomaly region. Based on this as well as the results in Table 3.3, it is better to use anomaly detection algorithms such as isolation forest and robust covariance for the task of crack segmentation.

3.4.3 Analysis of the effect of crack width

Crack width is an important factor to consider when analyzing the accuracy of crack segmentation since it is related to the visibility of the crack and the difficulty in extracting the crack points from surrounding points. Figure 3.13 shows a graph analyzing the trend of

the F1 score with varying crack widths when performing crack segmentation on the Nepal dataset. The graph shows that there is a general upward trend in F1 score as the crack width increases and this trend is observed across different feature representations. Larger cracks usually occupy a larger region in the point cloud and have more prominently different color and intensity values, so it is reasonable that larger cracks can be more accurately detected by the crack segmentation algorithm. There are a few cases where the F1 score decreases with crack width. This is because there are other factors that affect the difficulty of segmenting a particular crack such as the noise level in the point cloud and the prominence of the crack. Overall, the impact on performance of crack width is acceptable because the lower accuracy on smaller cracks is less important whereas the higher accuracy on larger cracks is more important for structural analysis of the building component.

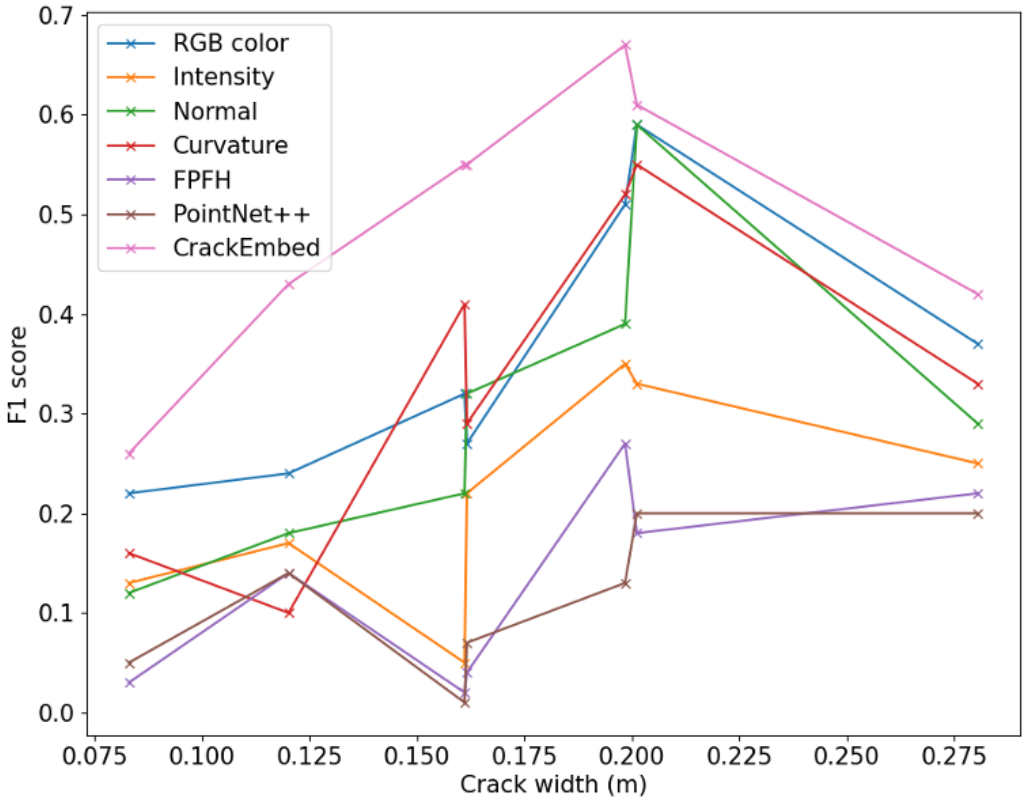


Figure 3.13: Graph of F1 score vs crack width for different feature representations

3.4.4 Analysis of the effect of outlier ratio

Another important factor to consider when analyzing the crack segmentation accuracy is the ratio between the number of points on the crack and the total number of points on a building component. This is because the assumption of the anomaly detection step is that the points on the crack makes up a minority of all the points and can be separated as outliers. The effect of this outlier is analyzed in Figure 3.14, which shows the relationship between the F1 score and the outlier ratio for different feature representations on the Nepal dataset. Based on the results, there is no clear increasing or decreasing trend in the accuracy with different outlier ratios. This is likely because the outlier ratio is mostly low in the Nepal dataset (i.e. points on cracks only make up around 5% - 15% of the point cloud) so it does not significantly alter the feature distribution when performing anomaly detection. There could potentially be situations where the outlier ratio is high such as partially collapsed structures or structures with a deformed surface. However, those situations are more suitable to be processed by deformation detection or debris detection algorithms and not crack detection which is the focus of this study.

3.4.5 Analysis of the effect of point cloud resolution

The crack segmentation accuracy can be further analyzed by examining the effect of point cloud resolution. Depending on the laser scanning hardware as well as the distance between the scanned structure and the laser scanner, the point cloud resolution (i.e. the spacing between individual points in the point cloud) could be different. This effect was simulated on the Nepal dataset by applying voxel-grid downsampling to the original laser-scanned point cloud at varying resolutions of 0.001m, 0.002m, 0.005m, 0.007m, 0.01m, 0.02m, 0.05m, 0.07m, and 0.1m. Figure 3.15 shows the effect of point cloud resolution on the F1 score whereas Figure 3.16 shows the effect of point cloud resolution on the crack width estimation error, averaged over all the columns in the dataset. Note that in both graphs, the X and Y axes are plotted on a logarithmic scale for easier visualization. Results show that

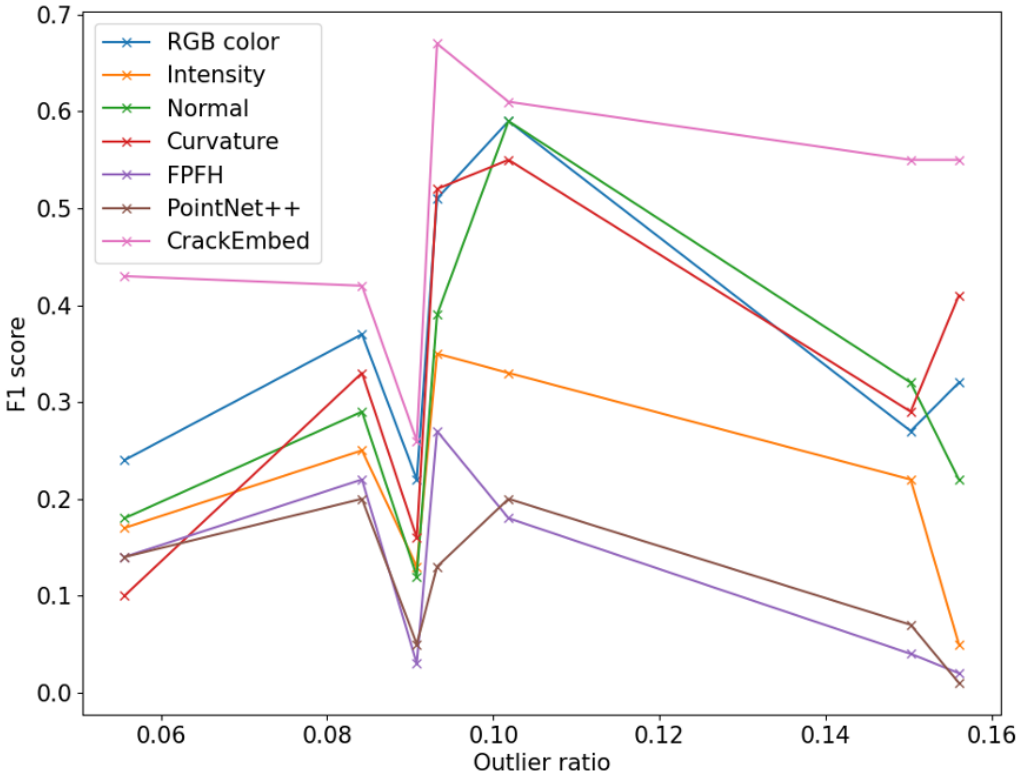


Figure 3.14: Graph of F1 score vs outlier ratio for different feature representations

the segmentation performance remains mostly unchanged in the 0.001m to 0.01m range. Whereas, the F1 score decreases significantly and the width error increases significantly when the resolution approaches the 0.1m range. This trend makes sense because when the point spacing is large, there is less information in the point cloud for the segmentation algorithm to clearly identify the outline of the crack. On a practical level, the results also suggest that when acquiring the point cloud data, having a resolution (point spacing) in the 1 mm to 1 cm range is important to maintain accurate crack segmentation and crack width estimation.

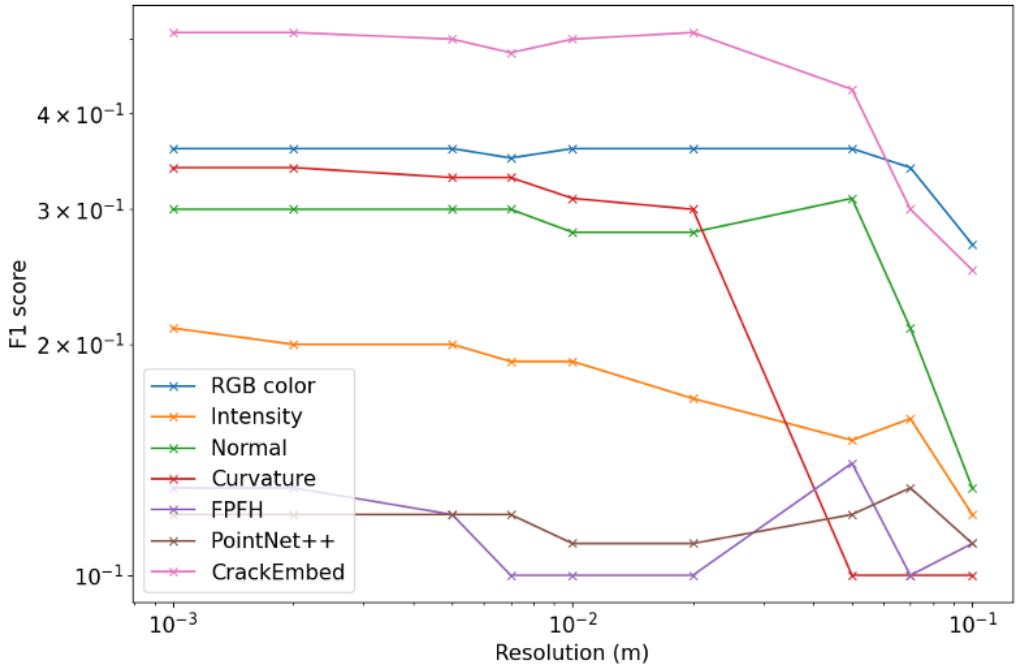


Figure 3.15: Graph of F1 score vs point cloud resolution for different feature representations

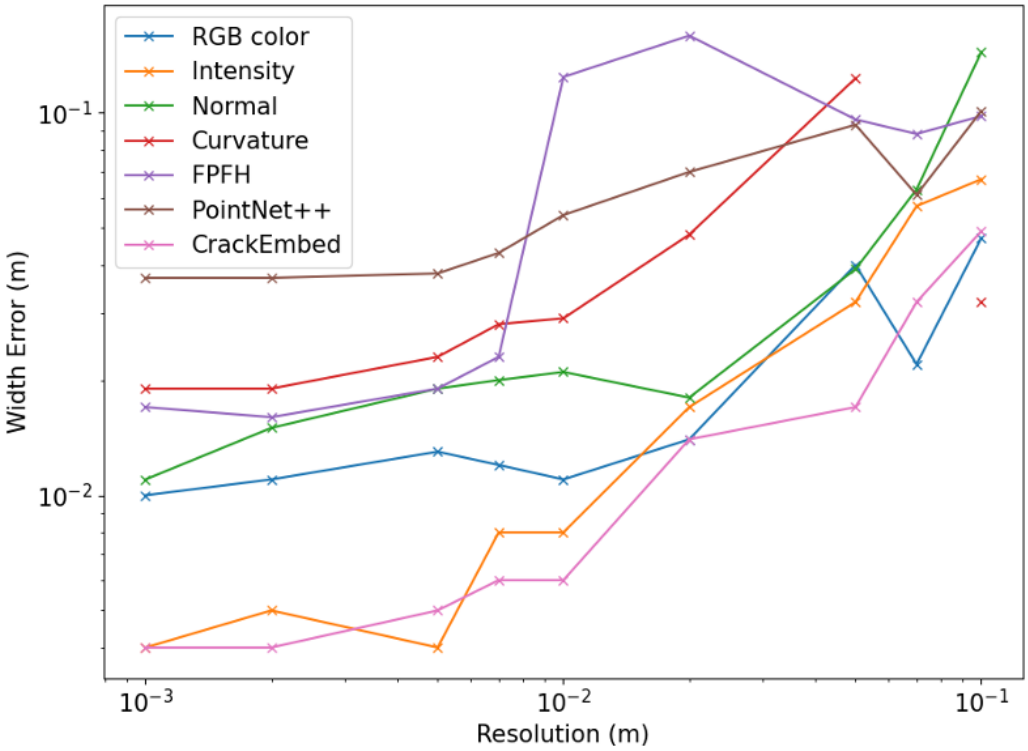


Figure 3.16: Graph of width error vs point cloud resolution for different feature representations

CHAPTER 4
INCREMENTAL SEGMENTATION

When surveying an unknown environment such as a disaster site, mobile robots need to be able to sense and understand the surrounding environment. Whether using cameras [109], depth sensors [110], or laser scanners [111], the surrounding 3D scene can be represented and processed as point cloud data. The use of point cloud data has seen increased popularity as the representation of choice for 3D scene understanding [3][4]. This is because point clouds can be stored as a simple array of points and are able to represent precise 3D geometry of objects.

When considering the problem of processing point cloud data for robotic applications, current methods for point cloud semantic segmentation such as PointNet [10] and SGPN [11] are not suitable for real-time scanning because they are fundamentally offline methods and do not process data incrementally. It is desirable to have a point cloud segmentation method that can process new scans in an online manner and still incorporate information from previous scans.

This study proposes a multi-view incremental segmentation method to address the problem of online instance segmentation of 3D point clouds. The segmentation process is designed to assign a semantic label and an instance point to each scanned point. A deep neural network, MCPNet, is proposed, which uses a multi-view context pooling (MCP) module to incorporate information from previous scans to improve the segmentation of the current scan. The MCP module works by selectively extracting points from the global pool of points that are relevant to the current region scanned by the robot. This mechanism allows the segmentation method to process new scans within tenths of a second compared to half a minute if an offline method were used to perform segmentation of the entire point cloud. The following sections will present, in order, the (i) architecture design, (ii) experimen-

tal results on simulation studies, (iii) field test at Guardian Centers, and (iv) experimental results on Guardian Centers data.

4.1 Architecture Design

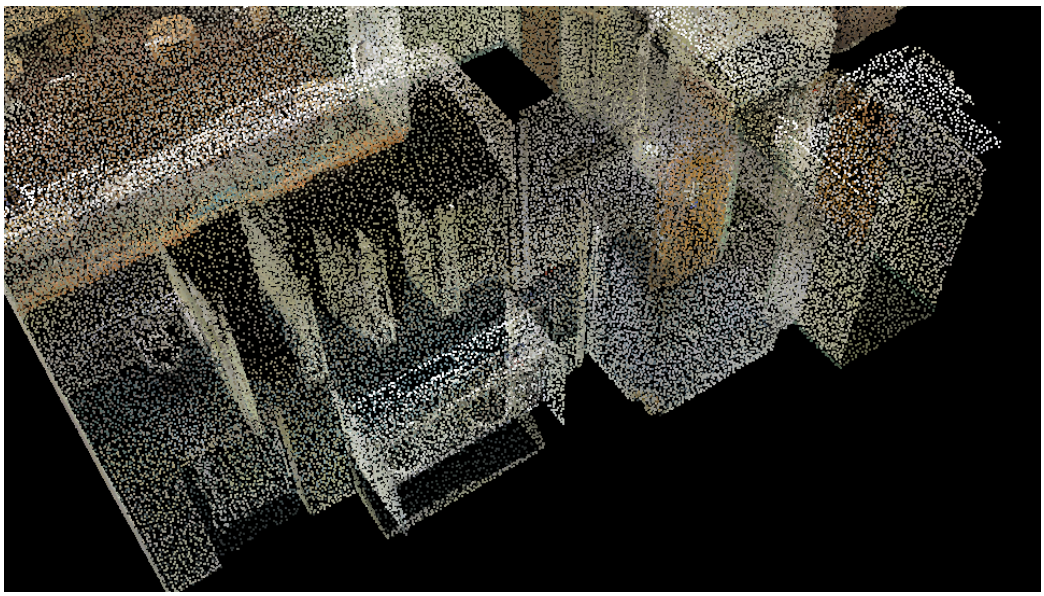
4.1.1 Data generation with ray tracing

To be able to generate a sufficient amount of training data for incremental segmentation, a simulated scanning environment is created where a virtual robot is able to acquire virtual scans by ray tracing. The simulated environment is generated based on the Stanford 3D Indoor Spaces (S3DIS) dataset [5], which contains ground truth semantic labels and instance labels for each point. The S3DIS dataset consists of 6 building areas in a university environment. The dataset has 13 labelled classes which are board, bookcase, beam, chair, column, door, sofa, table, window, ceiling, floor, wall, and clutter.

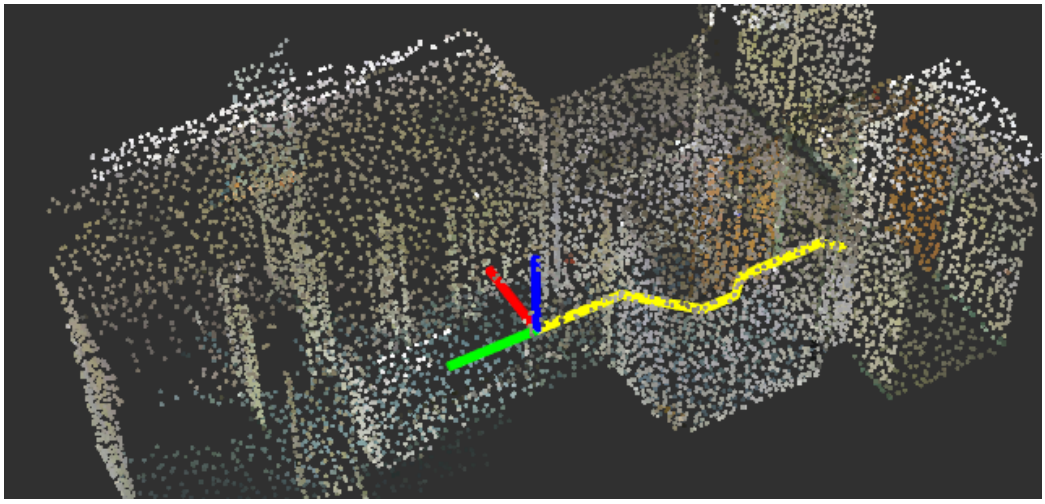
First, the original point cloud data is converted into a 3D occupancy grid representation where each occupied voxel in the simulated environment maps to a point in the real environment. Next, a collision-free trajectory is manually generated for the virtual robot to travel through the building and fully scan the surrounding environment. The virtual robot then executes this trajectory while collecting scan data every 0.2m. The virtual robot is assumed to have accurate localization so that the scanned coordinates are correct. The robot utilizes a virtual laser scanner which has a horizontal field of view of 360°, vertical field of view of 180°, horizontal resolution of 1°, and vertical resolution of 1°. The point cloud generated by this virtual laser scanner can be computed by ray-tracing: whenever a ray intersects an occupied voxel in the simulated environment, the corresponding point will be added to the scanned point cloud. The acquired point cloud scan consists of XYZ + RGB color points (assuming that the virtual robot has a camera to map color information onto the raw point cloud). The ground truth instance label and semantic label for each point can also be obtained from annotations of the original point cloud.

Figure 4.1 shows a comparison of the original point cloud and the synthetic point cloud

generated with ray-tracing. The figure shows that the original point cloud is more complete whereas the synthetic point cloud contains scanning artifacts such as occlusion and uneven resolution which are more representative of robot-scanned data in the real world. This ray-tracing process is advantageous because it can be used to automatically generate training data on a large-scale basis while reproducing real-world scanning issues such as occlusion and clutter. The ray tracing code has been publicly released at [112].



(a) Original point cloud



(b) Synthetic point cloud generated with ray-tracing

Figure 4.1: Ray-tracing example to generate a synthetic point cloud

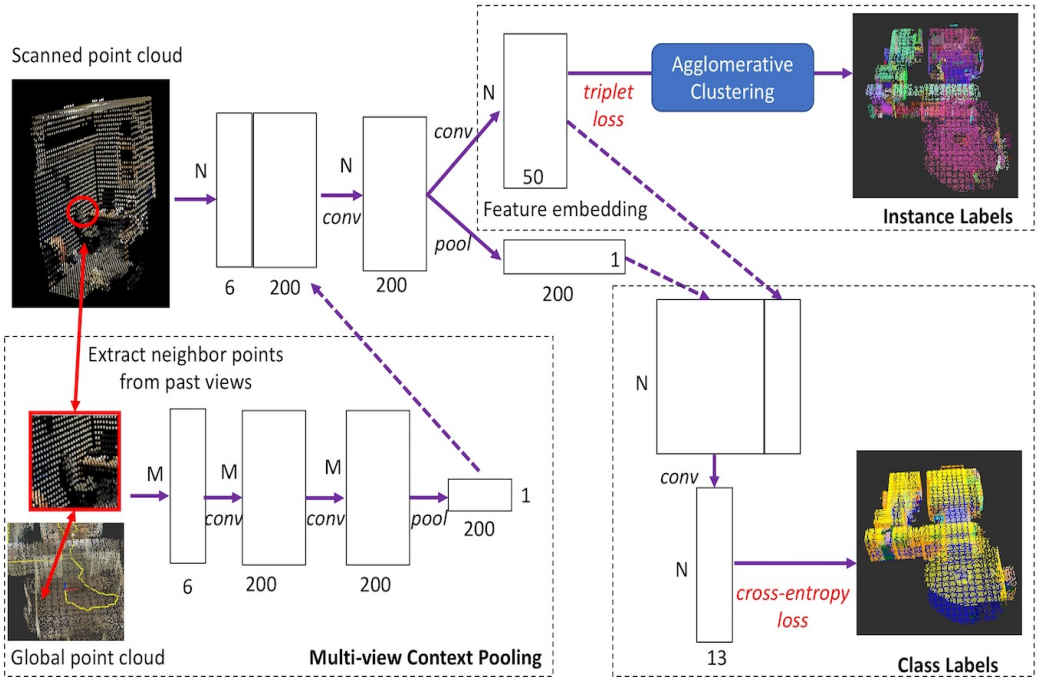


Figure 4.2: Network architecture of MCPNet

4.1.2 Incremental segmentation

The incremental segmentation framework is designed to be able to work in an online manner: at each timestamp where the robot acquires a new point cloud scan, the point cloud will be processed with a segmentation algorithm to obtain semantic labels and instance labels. More, specifically each point will be assigned a semantic label indicating what type of object it belongs to and an instance label indicating which object instance in the scene it belongs to.

A global lookup table is used to keep track of the scanned point clouds as well as the corresponding segmentation results. In this study, the lookup table is represented as a voxel grid with a voxel size of 0.1m, where the voxel coordinate is used as a unique key to retrieve a row from the global lookup table storing the raw point coordinates and segmentation results. This mechanism allows the incremental segmentation framework to be computationally efficient, since only one point is stored per voxel, and neighbor/context points for an arbitrary voxel can be easily retrieved from the global lookup table with

constant time complexity.

The input scan goes through a few pre-processing steps. First, the scan is filtered to keep only points that are within 2m from the scan origin. This is to overcome the sparsity issue for regions that are too far from the robot, with the idea that only points that are close to the robot should be processed whereas points that are far away can be processed as the robot moves closer to them. Next, the input coordinates are normalized by subtracting the scan origin the X-Y coordinates as well as subtracting the minimum Z value from the Z coordinates. Finally, random sampling with replacement is used to divide the scan into batches of N points.

Next, the input scan is passed to a deep neural network, MCPNet (shown in Figure 4.2). The input to MCPNet is a Nx6 matrix consisting of N points with 6 features each (XYZ and RGB color). The input feature matrix is passed through a feature convolution layer before splitting into two branches. The upper branch predicts a 50-dimensional feature embedding for each point and is responsible for computing instance labels. Whereas, the lower branch predicts a probability distribution that is used to determine the class labels. The lower branch initially uses a max pooling layer to obtain a global feature vector representing all points in the current scan. This global feature vector is then concatenated back into point-wise feature vectors and these combined features are passed to another convolution layers to predict the output of class probabilities.

The main innovation in MCPNet lies in the Multi-view Context Pooling (MCP) module which is used to pool features from past views and use them as contextual information to aid inference for the current scan. For each of the N original points in the input matrix, MCP extracts M neighbor points from past views and their corresponding features to form a Mx6 matrix. Neighbor points are defined as points that are at most three voxels away from the original point in the global lookup table. Random sampling with replacement over the neighbor points is used to ensure that the dimensions of the neighbor point input matrix is fixed at Mx6. This neighbor point input matrix is processed with 2 feature convolution

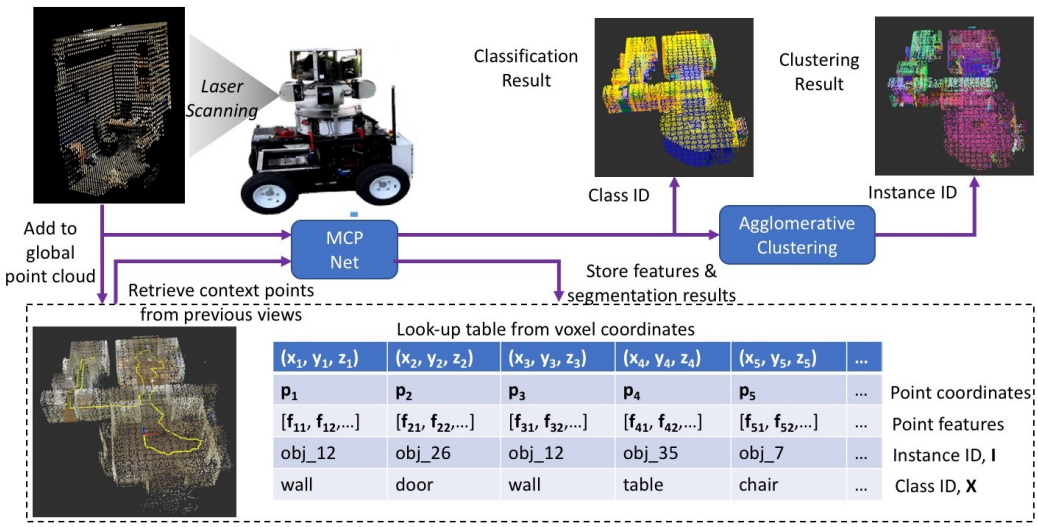


Figure 4.3: Framework for incremental point cloud segmentation using MCPNet

layers and 1 max pooling layer to arrive at a 200-dimensional feature vector, which represents the context information available for the original point. This process is repeated for each of the N original points to obtain a Nx200 context matrix which is concatenated to the original Nx6 input matrix.

MCPNet is trained from scratch using data generated from ray-tracing simulations. The loss function used for training is the sum of the triplet semihard loss for the upper branch and the softmax cross-entropy loss for the lower branch. The network is trained using the ADAM optimizer for a total of 100 epochs. The learning rate used is 0.001 whereas the batch size used is $N = 256$ points.

4.1.3 Point cloud clustering

As described in the previous section, the upper branch of MCPNet is used to predict a 50-dimensional feature embedding which can be further used for instance label prediction. The feature embedding is designed such that points that belong to the same object instance are close together in feature space whereas points that belong to different object instances are further apart in feature space. MCPNet is trained to learn this feature embedding using the triplet loss function [113], which computes a loss term based on a sample of three

points consisting of one similar pair and one dissimilar pair. The triplet loss is designed to enforce the constraint $||f(p_1) - f(p_2)||^2 + \alpha < ||f(p_1) - f(p_3)||^2$, where f is the projection function to feature space, p_1 and p_2 are points from the same object, p_1 and p_3 are points from different objects, and α is the margin of separation.

Once the feature embedding is obtained, the instance labels can be determined by performing agglomerative clustering. Agglomerative clustering works by forming clusters of connected points. Connected points are defined to be points that are within one voxel distance away in the global lookup table and have similar feature embeddings. Similarity in the feature embedding is determined based on the cosine similarity, $\frac{f(p_i) \cdot f(p_j)}{||f(p_i)||\,||f(p_j)||}$, and this value must be greater than β, where is a preset threshold hyperparameter. For a given point in the input scan, the instance label is assigned according to the following rules:

- If no connections exist, the point is initialized as the seed for a new cluster with a new instance ID.
- If connections exist to a single existing cluster, the point is added to that cluster and takes the corresponding instance ID.
- If connections exist to multiple existing clusters, all connected clusters are merged and their instance IDs are updated accordingly.

4.1.4 Incremental segmentation framework

The overall framework for incremental segmentaiton is shown in Figure 4.3. The framework utilizes a global lookup table based on voxel coordinates that stores the original point coordinates, point features, instance labels, and class labels. When the robot acquires laser scan data, new data is added to the global lookup table for points that lie in previously unseen voxels. The current scan points, together with neighbor points obtained by Multi-view Context Pooling (MCP), are passed to MCPNet to predict a class label and instance label for each point. Finally, these labels are used to update the global lookup table.

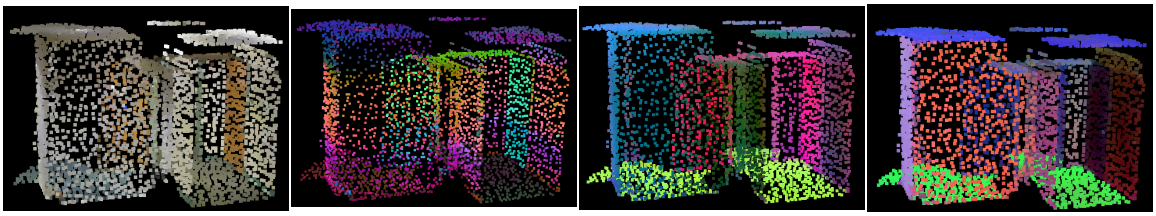


Figure 4.4: Visualization of different feature embeddings: (i) original point cloud (ii) SGPN embedding (iii) proposed embedding (without MCP) (iv) proposed embedding (with MCP)

4.2 Experimental Results on Simulation Studies

4.2.1 Classification performance

Table 4.1 shows a comparison of the classification performance of different segmentation methods, evaluated on the S3DIS dataset [5]. The evaluation metrics used are the Intersection-over-Union (IOU), IOU averaged over all classes, point-wise accuracy, and point-wise accuracy averaged over all classes. Cross validation is used, where 5 building areas are used as training data and 1 building area is used as validation data. The baseline methods for comparison are PointNet, PointNet++, SGPN, and VoxNet. In addition, the offline versions of PointNet and PointNet++ are also shown for comparison.

Results in Table 4.1 show that the proposed MCPNet has the best classification performance among online methods in terms of all four evaluation metrics. The use of the MCP module also resulted in a significant improvement in performance compared to the baseline method without using the MCP module. MCPNet did not perform as well as the offline versions of PointNet and PointNet++, but only for the case where the point clouds are pre-segmented into rooms. This shows that one of the critical reasons why the original offline versions of PointNet and PointNet++ are able to perform well is the availability of room segmentation, which is not the case for robotic real-time scanning.

Figure 4.5 shows the graph of average accuracy and inference rate against the number of context points used for MCPNet. The graph shows that as the number of context points increases, the average accuracy increases but the inference rate decreases. The final im-

Table 4.1: Classification performance on S3DIS dataset

Method	Mean IOU(%)	Overall IOU(%)	Mean Acc.(%)	Overall Acc.(%)
Offline methods				
PointNet [10]	39.6	58.3	55.1	73.6
PointNet++ [36]	49.3	65.3	67.0	78.9
PointNet [10] (no room segmentation)	36.2	54.0	51.1	70.0
PointNet++ [36] (no room segmentation)	39.0	55.3	57.1	71.1
Online methods				
PointNet [10]	25.0	42.1	41.9	59.1
PointNet++ [36]	23.8	43.1	36.5	60.2
SGPN [11]	24.9	43.3	39.5	60.3
VoxNet [30]	18.2	29.5	33.4	45.5
Proposed	25.4	41.1	40.8	57.7
Proposed + MCP	39.2	59.5	57.8	74.9

plementation uses 50 context points to maximize accuracy while maintaining a reasonable inference rate.

4.2.2 Clustering performance

Table 4.2 shows a comparison of the clustering performance (i.e. performance on the instance label prediction task) on the S3DIS dataset. The performance metrics that are used are Normalized Mutual Information (NMI), Adjusted Mutual Information (AMI), and Adjusted Rand Index (ARI), defined as in [84]. The baseline methods considered are normal and color-based region growing, PointNet, PointNet++, SGPN, and VoxNet. PointNet, PointNet++, and VoxNet perform clustering by assigning neighboring points that have the same class label to be in the same cluster. Whereas, SGPN and MCPNet perform clustering by predicting a feature embedding and using agglomerative clustering. Results in Table 4.2 show that MCPNet achieves the best clustering performance across all three metrics.

Figure 4.4 shows a visualization of the feature embeddings generated by different deep neural networks. The visualization is generated by taking the final layer of the network

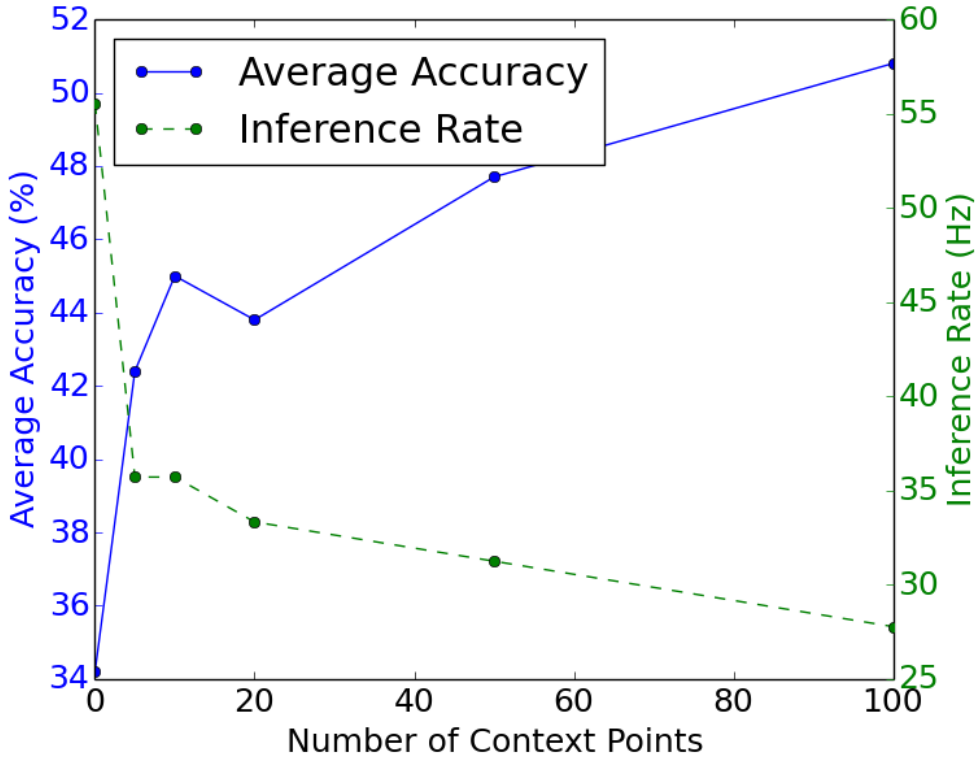


Figure 4.5: Graph of average accuracy and inference rate against the number of context points

branch used to perform clustering, projecting the feature vector to 3 dimensions using Principal Component Analysis, and then assigning the RGB color of each point based on its projected 3-dimensional feature vector. Figure 4.4 shows that the proposed method with MCP is able to generate feature embeddings that are most suitable for clustering (i.e. points from the same object are colored similarly).

Figure 4.6 shows a visualization of different stages of incremental segmentation on the S3DIS dataset. The figure shows that as the simulated robot moves around the building and acquires more point cloud data, the instance labels and semantic labels are incrementally updated. On the other hand, Figure 4.7 shows a visualization of the classification results in different areas of the building and from different angles.Note that the ceiling is removed in Figure 4.7 for clarity.

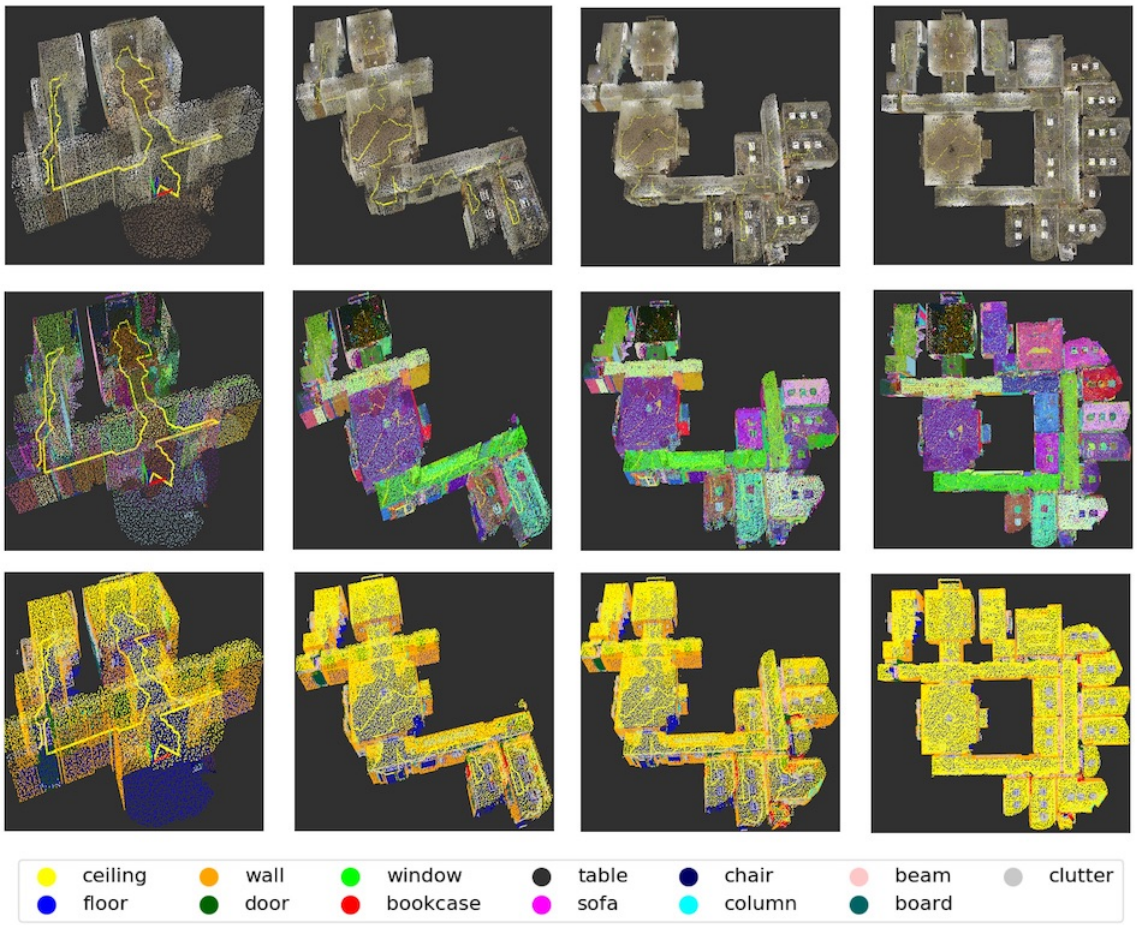


Figure 4.6: Visualization of incremental segmentation results on the S3DIS dataset. Top row shows the input point cloud, middle row shows the clustering results, and the bottom row shows the classification results.

4.2.3 Efficiency evaluation

The performance of different online segmentation algorithms can also be compared with respect to their time efficiency and memory efficiency. The efficiency evaluation is performed on an Intel Xeon E3-1200 CPU paired with a NVIDIA GTX1080 GPU. Table 4.3 shows that the proposed MCPNet has a higher processing time than other methods, but still has a short enough processing time to be run in real-time for a laser scanner operating at 10 Hz. The CPU usage is on par with other methods, whereas the network size is smaller than all the other methods except VoxNet. Figure 4.5 shows that the inference rate is dependant on the parameter for number of context points. Thus, the inference rate can be increased

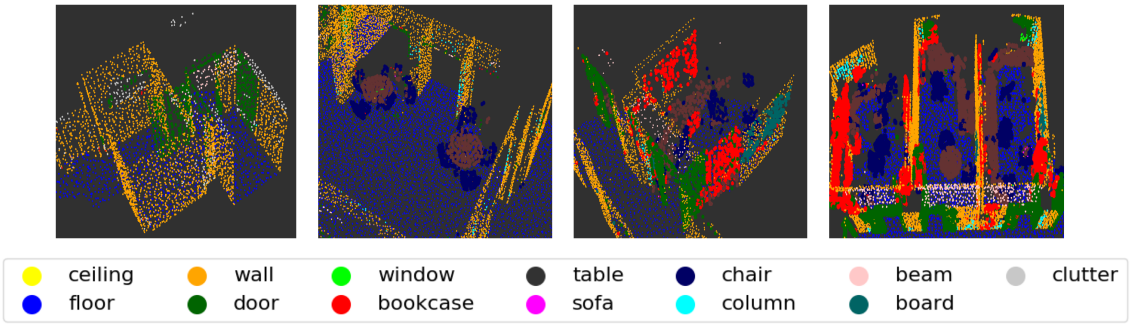


Figure 4.7: Visualization of classification results from different angles: (i) hallway with doors (ii) hallway with tables and chairs (iii) office room with board, bookcases, tables and chairs (iv) three adjacent offices

Table 4.2: Clustering performance on S3DIS dataset

Method	NMI(%)	AMI(%)	ARI(%)
Normal + color	78.5	63.4	25.0
PointNet [10]	72.1	59.2	12.1
PointNet++ [36]	76.1	66.4	17.1
SGPN [11]	78.5	60.5	26.1
VoxNet [30]	70.5	58.0	11.3
Proposed	75.9	64.8	18.3
Proposed + MCP	85.6	74.2	39.7

by decreasing the number of context points while sacrificing the accuracy.

4.2.4 Comparison with other online variants

This study also compares the performance between the proposed MCPNet with other variants of PointNet that can work online. Figure 4.8 shows a visualization of these different variants. The simplest version of online PointNet, which is the result shown in Table 4.1. This variant works by processing point cloud data on a scan-by-scan basis, i.e. the point cloud from each scan is passed in single batches to PointNet to predict the class labels. Another version of PointNet is with temporal expansion, where previous scans within 0.1s of the current scan are combined together to form contextual information that can be used as input to PointNet when predicting the class labels. The output class labels are then

Table 4.3: Efficiency comparison for instance segmentation on S3DIS

Method	Processing Time per Scan (s)	Network Size (MB)	CPU Memory Usage (GB)
PointNet [10]	0.017	14.0	0.94
PointNet++ [36]	0.027	11.5	0.97
SGPN [11]	0.019	14.9	2.49
VoxNet [30]	0.017	0.5	1.22
Proposed	0.019	0.9	1.26
Proposed + MCP	0.034	1.8	1.26

masked so that only the points in the current scan are updated. A third version of PointNet is with spatial expansion, where previous scans within 0.5m of the current scan are combined together to form contextual information. Note that these 3 variants use the original PointNet architecture and are trained using the original method. In contrast, MCPNet uses a novel Multi-view Context Pooling module to incorporate context information from previous scans in a more structured manner and also conducts training in scanning simulations that provide realistic sequences of point cloud data.

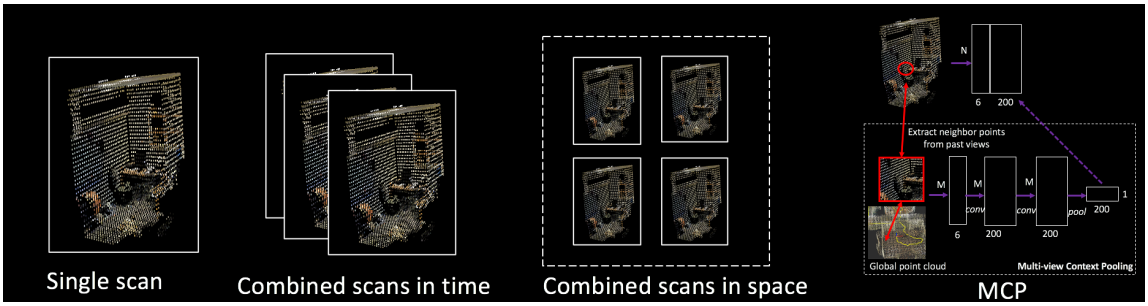


Figure 4.8: Comparison of MCPNet framework with other online variants of PointNet

Table 4.4 shows a quantitative comparison between the segmentation performance of these online variants of PointNet with the proposed MCPNet on the S3DIS dataset. Results show that both variants of online PointNet that use multiple scans outperform the single scan version on semantic segmentation accuracy metrics. This result is expected since using multiple scans in the prediction stage can provide more contextual information about objects in the scene. The results also show that MCPNet still has the best segmentation

performance overall. This suggests that the custom MCP module and training framework are effective in helping the network learn to incorporate information from previous scans. In terms of the computation time, the temporal expansion PointNet variant did not lead to significant increase in computation time compared to the simple online PointNet since the scans are already sorted and indexed in the order of acquisition time so the overhead of looking up context points is not too big. On the other hand, the spatial expansion variant as well as MCPNet have larger increases in computation time. This is because they require a spatial lookup for context points from previous scans which are not stored contiguously in memory.

Table 4.4: Online segmentation performance on S3DIS dataset

Method	Mean IOU	Overall IOU	Mean Acc.	Overall Acc.	Time(s)
PointNet (single scan)	25.0	42.1	41.9	59.1	0.017
PointNet (multiple scans in 0.1s)	27.8	48.0	43.4	64.8	0.017
PointNet (multiple scans in 0.5m)	29.4	49.7	45.4	66.4	0.051
Proposed MCPNet	39.2	59.5	57.8	74.9	0.034

4.3 Field test at Guardian Centers

To verify the proposed method for robotic scanning at disaster sites, the Guardian Centers disaster training facility was selected as a test environment. The facility has multiple large-scale structures with simulated damages from earthquakes, hurricanes, or terrorist attacks. This environment allows further investigation of the ability for mobile robots to complete scanning missions and perform incremental segmentation of the acquired scans.

This research makes use of the Ground Robot for Mapping Infrastructure (GROMI) to carry out scanning of damaged structures. The mobile robot is equipped with a Velodyne VLP-16 laser scanner and is built to travel over rough terrain. Figure 4.9 shows the robotic scanning process carried out around a damaged concrete structure at Guardian Centers. The

scene spans a region of 65m x 55m x 20m and contains a mix of regular building structures such as beams, columns, walls, and slabs as well as damaged structures and debris. As the robot travels around the site, laser scan data is collected dynamically and registered into a combined point cloud using Simultaneous Localization and Mapping (SLAM). Overall, the scans collected amounted to around 4 hours of data totaling 60GB.

Figure 4.10 shows the overall experimental setup for annotation and organization of the acquired point cloud data. First, a static ground-based laser scanner is used to obtain high resolution point clouds of the test site. The resulting point cloud is manually annotated to get ground truth instance labels and class labels for each point in the scene. The instance labels delineate the object or building element instances whereas the class labels indicate the type of objects. There are two main advantages of annotating ground truth from the static laser scans: (i) the static laser scans are acquired at a higher resolution and have a lower level of noise, (ii) the static scan of a structure only has to be annotated once and the annotations can be reused for multiple robotic scans of the same structure. On the other hand, the mobile robot data is divided into individual scanning sequences each spanning a duration of around 3 minutes. Each of the scanning sequences are automatically registered by using the Lightweight and Ground-Optimized Lidar Odometry and Mapping (LeGO-LOAM) [114] method. The scanning sequences are then registered to the annotated static point cloud so that the ground truth annotations can be matched to each individual scans. Note that the ground truth annotations are matched on a point-to-point basis so that any point not contained in the robotic scans, such as those from the interior areas of the damaged structure, is not considered for evaluation. The scanning sequences are subdivided into training scenes and test scenes for the application of a deep learning framework.

Figure 4.11 summarizes the main differences between the indoor simulation environment used in Section 4.3 and outdoor scanning with a real robot. First, the simulation environment has ideal localization and mapping whereas the real environment has noisy localization and mapping due to robot motion and vibration as well as registration errors.

Next, the simulated environment has mostly smaller scenes and objects at the room-level whereas the real environment spans a larger region and has large building structures. In addition, it is possible to acquire a dense point cloud in the simulated environment whereas the real environment has mostly sparse point clouds since the VLP-16 scanner has only 16 vertical scan lines. Finally, the simulated environment has regular building structures whereas the real environment has damaged building structures that could be significantly deformed from the original shapes. These differences make it much more challenging to implement an effective segmentation method for the real environment compared to the simulated environment.

To overcome these differences, the incremental segmentation framework was slightly modified for the outdoor scanning environment. First, the local scanning range is extended to 10m relative to the robot position compared to 2m in the indoor environment. This is because the Velodyne VLP-16 device has a minimum scanning range of around 1m so not many objects can be scanned at close range. At the same time, scan points that are further away than 10m are filtered out because those regions are too sparse to be accurately segmented. Another modification is in terms of staging the training data. In the simulated indoor environment, the robot motion is constant so scan data is acquired at a set interval of 0.2m. Whereas in the outdoor environment, the robot motion is variable so scan data is acquired at the native scanning rate of 10Hz and the point clouds are randomly downsampled to remain memory efficient. Finally, the hyperparameter β for incremental clustering is increased from 0.9 to 0.92. This is because the outdoor point clouds are much noisier and more cluttered so increasing the clustering threshold helps maintain the separation between point cloud segments of different objects.

4.4 Experimental Results on Guardian Centers data

The segmentation performance was evaluated on the Guardian Centers dataset as described in the previous section. The dataset consists of 6 scanning sequences, of which 5 are used

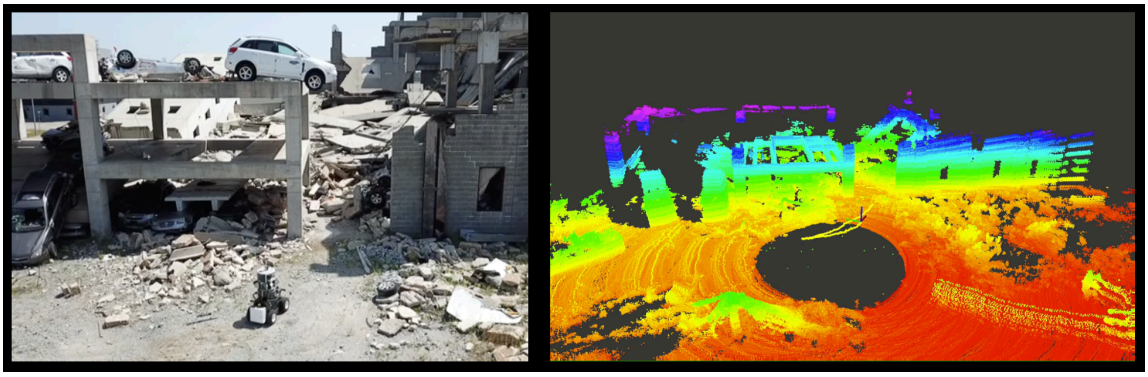


Figure 4.9: Robotic laser scanning at a damaged concrete structure at Guardian Centers: (left) photo of the GROMI robot captured by a drone (right) registered point clouds after applying SLAM

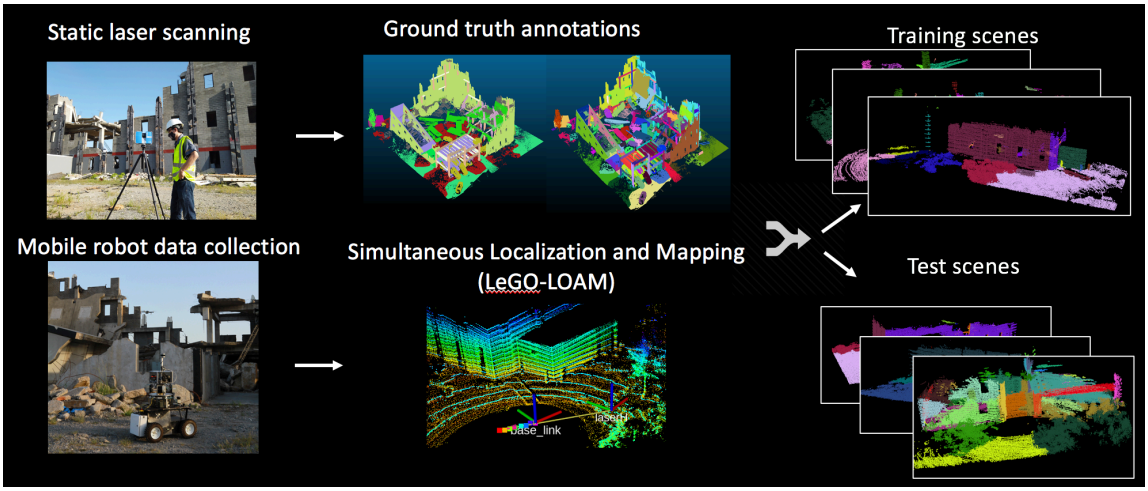


Figure 4.10: Experimental setup for data collection at Guardian Centers

for training and 1 is used for testing. Each scanning sequence generates a point cloud scene of around 100000 points after downsampling. The training sequences contain 5475 individual Velodyne scans whereas the test sequence contains 900 individual Velodyne scans. The semantic segmentation performance is measured using the classification accuracy and classification IOU metrics whereas the instance segmentation performance is measured using the NMI and AMI metrics. Table 4.5 shows a comparison of the semantic segmentation and instance segmentation performance between MCPNet and several baselines including (i) PointNet, (ii) PointNet++, (iii) SGPN and (iv) VoxNet. All networks are trained on the Guardian Centers training dataset. Results show that MCPNet obtained the best perfor-

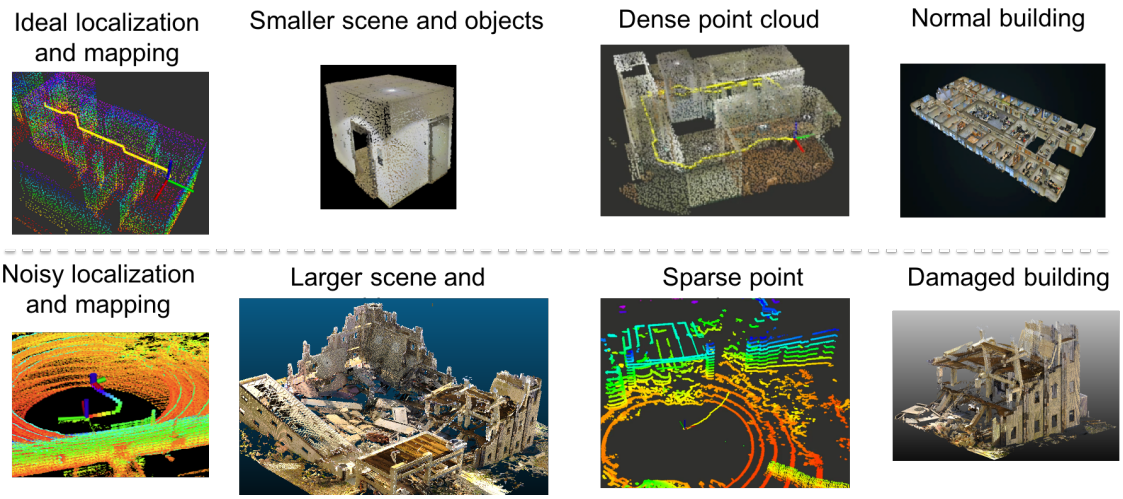


Figure 4.11: Comparison between indoor simulation with S3DIS and outdoor scanning at Guardian Centers

mance in 3 out of 4 metrics (classification accuracy, classification IOU, and NMI).

Figures 4.12 and 4.13 show the instance label prediction results and semantic label prediction results respectively. Compared to the simulated environment, the instance label prediction resulted in significantly more oversegmentation due to the sparser point cloud. In addition, it is more difficult to accurately predict the segmentation boundaries due to noise in the point cloud. The class label prediction results are also less accurate compared to the simulated environment. This is because there are several classes that are easily confused for each other such as clutter and debris as well as wall and damaged wall.

Table 4.6 shows the computation time and memory usage of each segmentation method on the Guardian Centers dataset, measured on an Intel Xeon E3-1200 CPU with a NVIDIA GTX1080 GPU. Results show that similarly to the case with the S3DIS dataset, MCPNet had a higher processing time per scan, but still at a reasonable rate such that it can be run at the 10 Hz capture rate of the Velodyne LiDAR scanner.

Table 4.5: Segmentation performance on Guardian Centers dataset

Method	Class Acc.(%)	Class IOU(%)	NMI(%)	AMI(%)
PointNet	28.8	16.8	46.7	42.8
PointNet++	32.1	19.1	33.1	24.3
SGPN	29.9	17.6	54.8	24.2
VoxNet	19.6	10.9	37.1	28.8
Proposed MCPNet	42.8	27.2	56.4	33.7

Table 4.6: Efficiency comparison for instance segmentation on Guardian Centers

Method	Processing Time per Scan (s)	Network Size (MB)	CPU Memory Usage (GB)
PointNet	0.036	14.0	1.07
PointNet++	0.050	11.5	1.72
SGPN	0.039	14.9	1.61
VoxNet	0.036	0.5	1.55
Proposed MCPNet	0.054	1.8	1.85

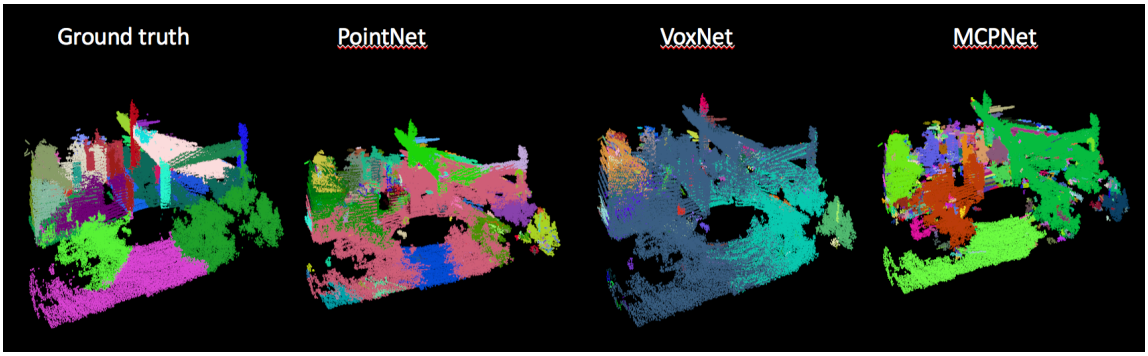


Figure 4.12: Visualization of instance label prediction results

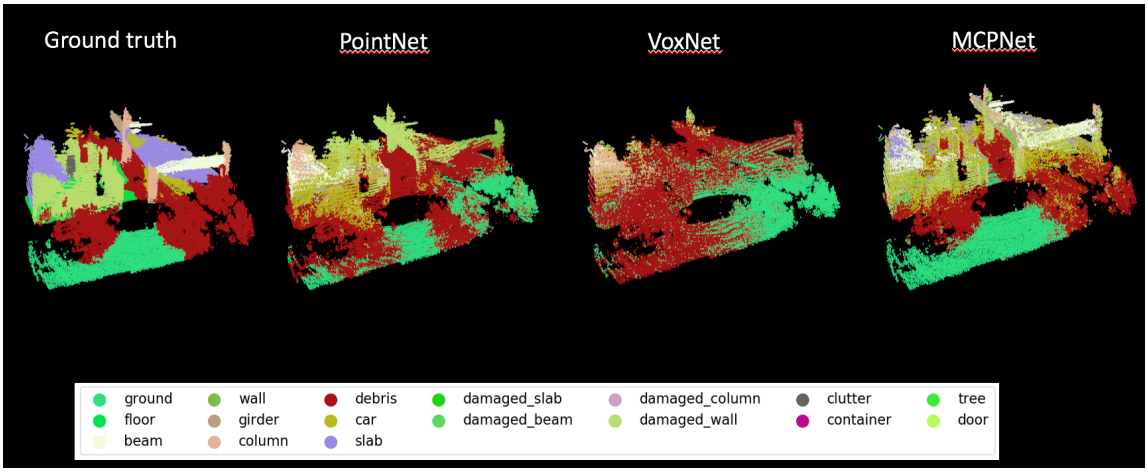


Figure 4.13: Visualization of class label prediction results

CHAPTER 5
CONCLUSION AND FUTURE DIRECTIONS

In conclusion, this thesis presents a framework for 3D segmentation and damage analysis from robotic scans of disaster sites. First, to address the issue of generalizability in point cloud segmentation, a learnable region growing method, LRGNet is introduced. LRGNet is class-agnostic in the sense that it does not rely on class information during training and can theoretically segment object classes that it has not seen during training. LRGNet works by decomposing the region growing problem into smaller subproblems and uses a deep neural network to predict how to add an remove points from a region to morph it into a more complete point cloud object. Experimental results on the S3DIS and ScanNet datasets show that LRGNet is able to better generalize across datasets that have different classes of objects. On the other hand, to tackle the problem of automated damage assessment from disaster site point clouds, a method for crack segmentation using unsupervised machine learning techniques is proposed. A deep neural network is used to compute a discriminative point feature embedding which is then processed by anomaly detection algorithms to detect cracks in point clouds. The performance of the proposed approach was evaluated based on experimental studies on laser-scanned point clouds from two different sites. Findings show that isolation forest and robust covariance were the best performing anomaly detection algorithms whereas the triplet loss embedding was the most effective feature representation for crack segmentation. An ablation study into the effects of different parameters showed that the crack segmentation performance increases with crack size whereas the effect of outlier ratio is less significant. In addition, having a resolution (point spacing) in the 1 mm to 1 cm range is important to maintain accurate crack segmentation and crack width estimation. Finally, the MCPNet method is proposed for performing multi-view incremental segmentation to achieve real-time segmentation of laser-scanned

point clouds for robotic applications. MCPNet is initially trained by utilizing simulated data that is generated through ray-tracing from the S3DIS dataset. MCPNet is able to outperform other online methods in terms of both classification metrics and clustering metrics on this simulated dataset. The incremental segmentation framework was further validated with real robot experiments at the Guardian Centers disaster training site. Even though the disaster environment was much more challenging than the indoor environment due to noise and clutter, the proposed approach still performed the best by making use of context information from multiple views.

Overall, the research findings from this thesis offer promising directions for future research in the area of robotic point cloud data collection for disaster relief. Having a learnable region growing method for class-agnostic point cloud segmentation is an important first step since it can be used to segment objects of arbitrary shape, size, or class. This is especially relevant in the context of disaster relief operations since building elements are likely to be damaged or deformed and are more difficult to be be matched with a specific class of objects. In view of its high computational cost, future work should be focused on enabling real-time implementation by reducing the network size or using techniques such as distillation to speed up network inference. In addition, modifying the network architecture to use Recurrent Neural Network (RNN) or Long-Short Term Memory (LSTM) layers could improve performance by enforcing temporal consistency in region growing. On the other hand, the anomaly detection based framework for crack segmentation provides an avenue for structural damage detection under low supervision conditions. Future work in this area should involve extending the method to segmentation of a wider variety of surface damage, including concrete spalling and water damage. In addition, a larger scale experimental study should be carried out to validate the effectiveness of point cloud based damage detection in comparison with more traditional methods of damage detection such as vibration sensors or acoustic sensors. Finally, this research offers a new direction for incremental segmentation of robotic scans using the idea of multi-view context pooling. The method

performed well in the simulated indoor environment but did take a performance hit in the outdoor disaster site environment. Future research should be focused on improving the robustness in cluttered environments commonly found on disaster sites. Moreover, further efforts need to be carried out to integrate the learnable region growing and crack segmentation methods into the incremental segmentation framework. Nonetheless, the experimental studies in this thesis have demonstrated the importance of generalizability, transferability, and efficiency when it comes to processing data from challenging environments such as disaster sites. It is hoped that the ideas and results presented in this thesis can contribute to a better understanding of how robots can effectively acquire and process spatial data from disaster sites, enabling rapid assessment of structural damage after disaster events and ensuring that the post-disaster rehabilitation process can be carried out safely and smoothly.

REFERENCES

[1] J. Casper and R. R. Murphy, "Human Robot Interactions During the Robot-Assisted Urban Search and Rescue Response at the World Trade Center", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 33, no. 3, pp. 367–385, 2003.

[2] T. Sakaue, S. Yoshino, K. Nishizawa, and K. Takeda, "Survey in Fukushima Daiichi NPS by Combination of Human and Remotely-Controlled Robot", in *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2017, ISBN: 9781538639238.

[3] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans", in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018.

[4] J. Chen, Y. K. Cho, and J. Ueda, "Sampled-point network for classification of deformed building element point clouds", in *Proceedings of the 2018 IEEE Conference on Robotics and Automation (ICRA)*, 2018.

[5] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces", in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite", in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes", in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, IEEE Computer Society, 2015, pp. 1912–1920.

[8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository", arXiv:1512.03012 [cs.GR], 2015.

[9] T.-y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollar, "Microsoft COCO : Common Objects in Context", *https://arxiv.org/pdf/1405.0312.pdf*, 2015. arXiv: arXiv:1405.0312v3.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation", *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[11] W. Wang, R. Yu, Q. Huang, and U. Neumann, "Sgpn: Similarity group proposal network for 3d point cloud instance segmentation", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] J. Lee, H. Son, C. Kim, and C. Kim, "Skeleton-based 3D reconstruction of as-built pipelines from laser-scan data", *Automation in Construction*, vol. 35, pp. 199–207, 2013.

[13] H. Son, C. Kim, and C. Kim, "Automatic 3D Reconstruction of As-Built Pipeline Based on Curvature Computations from Laser-Scanned Data Hyojoo", pp. 925–934, 2014.

[14] A. Adan and D. Huber, "3D Reconstruction of Interior Wall Surfaces Under Occlusion and Clutter", in *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2011, ISBN: 9780769543697.

[15] J Jung, S Hong, S Yoon, J Kim, and J Heo, "Automated 3D Wireframe Modeling of Indoor Structures from Point Clouds Using Constrained Least-Squares Adjustment for As-Built BIM", *Journal of Computing in Civil Engineering*, vol. 30, no. 1, p. 4 015 074, 2015.

[16] A. Dimitrov and M. Golparvar-Fard, "Segmentation of building point cloud models including detailed architectural/structural features and MEP systems", *Automation in Construction*, vol. 51, no. C, pp. 32–45, 2015.

[17] C. Wang and Y. K. Cho, "Smart scanning and near real-time 3D surface modeling of dynamic construction equipment from a point cloud", *Automation in Construction*, vol. 49, pp. 239–249, 2015.

[18] X. Xiong, A. Adan, B. Akinci, and D. Huber, "Automatic creation of semantically rich 3D building models from laser scanner data", *Automation in Construction*, vol. 31, pp. 325–337, 2013.

[19] Y. K. Cho and M. Gai, "Projection-Recognition-Projection Method for Automatic Object Recognition and Registration for Dynamic Heavy Equipment Operations", *Journal of Computing in Civil Engineering*, vol. 28, no. 5, A4014002, 2014.

[20] Y. Turkan, F. Bosché, C. T. Haas, and R. Haas, "Tracking of secondary and temporary objects in structural concrete work - ProQuest", *Construction Innovation*, vol. 14, no. 2, pp. 145–167, 2014.

[21] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration", *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[22] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram", in *Intelligent Robots and Systems (IROS)*, 2010, pp. 2155–2162.

[23] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[24] Z.-c. Marton, D. Pangercic, R. B. Rusu, A. Holzbach, and M. Beetz, "Hierarchical Object Geometric Categorization and Appearance Classification for Mobile Manipulation", in *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*, 2010, ISBN: 1234256789.

[25] J. Chen, Y. Fang, Y. K. Cho, and C. Kim, "Principal Axes Descriptor for Automated Construction-Equipment Classification from Point Clouds", *Journal of Computing in Civil Engineering*, pp. 1–12, 2016.

[26] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification", *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2987–2992, 2011.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. arXiv: 1512.03385.

[28] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and Multi-View CNNs for Object Classification on 3D Data", in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2016, ISBN: 978-1-4673-8851-1. arXiv: 1604.03265.

[29] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and Discriminative Voxel Modeling with Convolutional Neural Networks", 2016. arXiv: 1608.04236.

[30] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition", in *IROS*, 2015.

[31] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni, "Learning object bounding boxes for 3d instance segmentation on point clouds", in *Advances in Neural Information Processing Systems 32*, 2019, pp. 6737–6746.

[32] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving", in *IEEE CVPR*, 2017.

[33] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[34] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, *Frustum PointNets for 3D Object Detection from RGB-D Data*, 2018. arXiv: 1711.08488.

[35] F. Liu, S. Li, L. Zhang, C. Zhou, R. Ye, Y. Wang, and J. Lu, "3dcnn-dqn-rnn: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds", in *The IEEE International Conference on Computer Vision (ICCV)*, 2017.

[36] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space", *arXiv preprint arXiv:1706.02413*, 2017.

[37] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds", in *IEEE International Conference on Computer Vision, 3DRMS Workshop, ICCV*, 2017.

[38] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung, "Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] L. Yi, W. Zhao, H. Wang, M. Sung, and L. J. Guibas, "Gspn: Generative shape proposal network for 3d instance segmentation in point cloud", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[40] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Efficient incremental map segmentation in dense rgb-d maps", in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 5488–5494.

[41] R. Dubé, M. G. Gollub, H. Sommer, I. Gilitschenski, R. Siegwart, C. Cadena, and J. Nieto, "Incremental-segment-based localization in 3-d point clouds", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1832–1839, 2018.

[42] A. Dimitrov and M. Golparvar-Fard, "Segmentation of building point cloud models including detailed architectural/structural features and mep systems", *Automation in Construction*, vol. 51, pp. 32 –45, 2015.

[43] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[44] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "Gvcnn: Group-view convolutional neural networks for 3d shape recognition", in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[45] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition", in *Proc. ICCV*, 2015.

[46] H Noh, R Rajagopal, and A. S. Kiremidjian, "Sequential structural damage diagnosis algorithm using a change point detection method", *Journal of Sound and Vibration*, vol. 332, no. 24, pp. 6419–6433, 2013.

[47] S. W. Doebling, C. R. Farrar, M. B. Prime, and D. W. Shevitz, "Damage identification and health monitoring of structural and mechanical systems from changes in their vibration characteristics: A literature review", *U.S. Office of Scientific and Technical Information*, 1996.

[48] T. M. Roberts and M Talebzadeh, "Acoustic emission monitoring of fatigue crack propagation", *Journal of Constructional Steel Research*, vol. 59, no. 6, pp. 695–712, 2003.

[49] T. Suzuki and M. Ohtsu, "Damage estimation of concrete canal due to earthquake effects by acoustic emission method", *Construction and Building Materials*, vol. 67, pp. 186–191, 2014, 1. Special Issue of KIFA-6 2. Utilization of Crumb Rubber in Asphalt Mixtures.

[50] S. Popovics, J. L. Rose, and J. S. Popovics, "The behaviour of ultrasonic pulses in concrete", *Cement and Concrete Research*, vol. 20, no. 2, pp. 259–270, 1990.

[51] J. B. Harley, "Predictive guided wave models through sparse modal representations", *Proceedings of the IEEE*, vol. 104, no. 8, pp. 1604–1619, 2016.

[52] Q Zou, Z Zhang, Q Li, X Qi, Q Wang, and S Wang, "Deepcrack: Learning hierarchical convolutional features for crack detection", *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1498–1512, 2019.

[53] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, "Deepcrack: A deep hierarchical feature learning architecture for crack segmentation", *Neurocomputing*, vol. 338, pp. 139–153, 2019.

[54] R. Fu, H. Xu, Z. Wang, L. Shen, M. Cao, T. Liu, and D. Novák, "Enhanced intelligent identification of concrete cracks using multi-layered image preprocessing-aided convolutional neural networks", *Sensors*, vol. 20, no. 7, p. 2021, Apr. 2020.

[55] C Benz, P Debus, H. K. Ha, and V Rodehorst, "Crack segmentation on uas-based imagery using transfer learning", Dec. 2019, pp. 1–6.

[56] G. Liu, N. Yang, L. Guo, S. Guo, and Z. Chen, "A one-stage approach for surface anomaly detection with background suppression strategies", *Sensors*, vol. 20, no. 7, p. 1829, Mar. 25, 2020.

[57] D. Tabernik, S. ela, J. Skvar, and D. Skoaj, "Segmentation-based deep-learning approach for surface-defect detection", *Journal of Intelligent Manufacturing*, vol. 31, no. 3, pp. 759–776, 2020.

[58] Y.-C. J. Tsai and F. Li, "Critical assessment of detecting asphalt pavement cracks under different lighting and low intensity contrast conditions using emerging 3d laser technology", *Journal of Transportation Engineering*, vol. 138, no. 5, pp. 649–656, 2012.

[59] V. Kaul, Y. J. Tsai, and R. M. Mersereau, "Quantitative performance evaluation algorithms for pavement distress segmentation", *Transportation Research Record*, vol. 2153, no. 1, pp. 106–113, 2010.

[60] A. Zhang, K. C. P. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen, "Automated pixel-level pavement crack detection on 3d asphalt surfaces using a deep-learning network", *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 10, pp. 805–819, 2017.

[61] A. Zhang, K. C. P. Wang, Y. Fei, Y. Liu, S. Tao, C. Chen, J. Q. Li, and B. Li, "Deep learning based fully automated pavement crack detection on 3d asphalt surfaces with an improved cracknet", *Journal of Computing in Civil Engineering*, vol. 32, no. 5, p. 4 018 041, 2018.

[62] Y Fei, K. C. P. Wang, A Zhang, C Chen, J. Q. Li, Y Liu, G Yang, and B Li, "Pixel-level cracking detection on 3d asphalt pavement images through deep-learning-based cracknet-v", *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 273–284, 2020.

[63] G Li, J Wan, S He, Q Liu, and B Ma, "Semi-supervised semantic segmentation using adversarial learning for pavement crack detection", *IEEE Access*, vol. 8, pp. 51 446–51 459, 2020.

[64] M. R. Jahanshahi, F. Jazizadeh, S. F. Masri, and B. Becerik-Gerber, "Unsupervised approach for autonomous pavement-defect detection and quantification using an inexpensive depth sensor", *Journal of Computing in Civil Engineering*, vol. 27, no. 6, pp. 743–754, 2013.

[65] A Akagic, E Buza, S Omanovic, and A Karabegovic, "Pavement crack detection using otsu thresholding for image segmentation", 2018, pp. 1092–1097.

[66] J. Chen, P. Kim, Y. K. Cho, and J. Ueda, "Object-sensitive potential fields for mobile robot navigation and mapping in indoor environments", in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 328–333.

[67] J. Chen and Y. K. Cho, "Point-to-point comparison method for automated scan-vs-bim deviation detection", 2018.

[68] S. Bose, A. Nozari, M. E. Mohammadi, A. Stavridis, M. Babak, R. Wood, D. Gillins, and A. Barbosa, "Structural assessment of a school building in sankhu, nepal damaged due to torsional response during the 2015 gorkha earthquake", S. Pakzad and C. Juan, Eds., Cham: Springer International Publishing, 2016, pp. 31–41, ISBN: 978-3-319-29751-4.

[69] B. G. Erkal and J. F. Hajjar, "Laser-based surface damage detection and quantification using predicted surface properties", *Automation in Construction*, vol. 83, pp. 285–302, 2017.

[70] W Liu, S Chen, and E Hauser, "Lidar-based bridge structure defect detection", *Experimental Techniques*, pp. 27–34, 2011.

[71] J. Chen, Y. K. Cho, and J. Ueda, "Sampled-point network for classification of deformed building element point clouds", 2018.

[72] M. E. Mohammadi, R. L. Wood, and C. E. Wittich, "Non-temporal point cloud analysis for surface damage in civil structures", *ISPRS International Journal of Geo-Information*, vol. 8, no. 12, 2019.

[73] Z. Zhou, J. Gong, and X. Hu, "Community-scale multi-level post-hurricane damage assessment of residential buildings using multi-temporal airborne lidar data", *Automation in Construction*, vol. 98, no. July 2017, pp. 30–45, 2019.

[74] L. Ma, R. Sacks, and R. Zeibak-shini, "Advanced Engineering Informatics Information modeling of earthquake-damaged reinforced concrete structures", *Advanced Engineering Informatics*, vol. 29, no. 3, pp. 396–407, 2015.

[75] R. Sacks, R. Zeibak-shini, A. Aryal, and S. Filin, "Preparation of Synthetic As-Damaged Models for Post-Earthquake BIM Reconstruction Research", *Journal of Computing in Civil Engineering*, vol. 30, no. 3, pp. 1–12, 2016.

[76] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space", *arXiv preprint arXiv:1706.02413*, 2017.

[77] P. Kim, J. Chen, and Y. K. Cho, "Robotic sensing and object recognition from thermal-mapped point clouds", *International Journal of Intelligent Robotics and Applications*, vol. 1, no. 3, pp. 243–254, 2017.

[78] A. C. C. Law, N. Southon, N. Senin, P. Stavroulakis, R. Leach, R. D. Goodridge, and Z. Kong, "Curvature-based segmentation of powder bed point clouds for inprocess monitoring", in *Proceedings of the 29th Annual International Solid Freeform Fabrication Symposium*, 2018.

[79] X. Wang, L. Zou, X. Shen, Y. Ren, and Y. Qin, "A region-growing approach for automatic outcrop fracture extraction from a three-dimensional point cloud", *Computers and Geosciences*, vol. 99, pp. 100 –106, 2017.

[80] J. Chen, Z. Kira, and Y. K. Cho, "Deep learning approach to point cloud scene understanding for automated scan to 3d reconstruction", *Journal of Computing in Civil Engineering*, vol. 33, no. 4, p. 04 019 027, 2019.

[81] R. Haeb-Umbach and H. Ney, "Improvements in beam search for 10000-word continuous-speech recognition", *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 353–356, 1994.

[82] T. Rabbani, F. van den Heuvel, and G. Vosselman, "Segmentation of point clouds using smoothness constraints", in *ISPRS 2006 : Proceedings of the ISPRS commission V symposium Vol. 35, part 6 : image engineering and vision metrology, Dresden, Germany 25-27 September 2006*, H. Maas and D. Schneider, Eds., vol. 35, International Society for Photogrammetry and Remote Sensing (ISPRS), 2006, pp. 248–253.

[83] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration", in *2009 IEEE International Conference on Robotics and Automation*, 2009.

[84] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance", *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Dec. 2010.

[85] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes", in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[86] M. Kazama and T. Noda, "Damage statistics (summary of the 2011 off the pacific coast of tohoku earthquake damage)", *Soils and Foundations*, vol. 52, no. 5, pp. 780–792, 2012, Special Issue on Geotechnical Aspects of the 2011 off the Pacific Coast of Tohoku Earthquake.

[87] H. Y. Noh, D. Lallemant, and A. S. Kiremidjian, "Development of empirical and analytical fragility functions using kernel smoothing methods", *Earthquake Engineering and Structural Dynamics*, no. October 2014, pp. 1163–1180, 2015.

[88] A. T. Council, *Fema 306: Evaluation of earthquake damaged concrete and masonry wall buildings*, https://www.fema.gov/media-library-data/20130726-1506-20490-1995/fema-306.pdf, 1998.

[89] Z. Zhu, S. German, and I. Brilakis, "Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation", *Automation in Construction*, vol. 20, no. 7, pp. 874 –883, 2011.

[90] K. Kovler and V. Chernov, "2 - types of damage in concrete structures", in *Failure, Distress and Repair of Concrete Structures*, ser. Woodhead Publishing Series in Civil and Structural Engineering, N. Delatte, Ed., Woodhead Publishing, 2009, pp. 32 –56, ISBN: 978-1-84569-408-1.

[91] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram", 2010, pp. 2155–2162.

[92] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration", *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[93] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[94] C. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation", *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[95] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space", 2017.

[96] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds", 2017.

[97] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces", 2016.

[98] E. Hoffer and N. Ailon, "Deep metric learning using triplet network", A. Feragen, M. Pelillo, and M. Loog, Eds., Cham: Springer International Publishing, 2015, pp. 84–92, ISBN: 978-3-319-24261-3.

[99] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding", ser. SODA '07, USA: Society for Industrial and Applied Mathematics, 2007, pp. 10271035, ISBN: 9780898716245.

[100] D. A. Reynolds, "Gaussian mixture models.", *Encyclopedia of biometrics*, vol. 741, 2009.

[101] D Comaniciu and P Meer, "Mean shift: A robust approach toward feature space analysis", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[102] F. T. Liu, K. M. Ting, and Z Zhou, "Isolation forest", 2008, pp. 413–422.

[103] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution", *Neural Comput.*, vol. 13, no. 7, pp. 14431471, 2001.

[104] P. J. Rousseeuw and V. K. Driessen, "A fast algorithm for the minimum covariance determinant estimator", *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[105] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers", ser. SIGMOD '00, New York, NY, USA: Association for Computing Machinery, 2000, pp. 93104, ISBN: 1581132174.

[106] G. Brando, D. Rapone, E. Spacone, M. S. O'Banion, M. J. Olsen, A. R. Barbosa, M. Faggella, R. Gigliotti, D. Liberatore, S. Russo, L. Sorrentino, S. Bose, and A. Stravidis, "Damage reconnaissance of unreinforced masonry bearing wall buildings after the 2015 gorkha, nepal, earthquake", *Earthquake Spectra*, vol. 33, no. S1, S243–S273, 2017.

[107] *Cloudcompare (version 2.10)*, 2020.

[108] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[109] L.-F. Posada, A. Velasquez-Lopez, F. Hoffmann, and T. Bertram, "Semantic mapping with omnidirectional vision", *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1901–1907, 2018.

[110] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, "Label fusion: A pipeline for generating ground truth labels for real rgbd data of cluttered scenes", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3325–3242.

[111] M. Antonello, D. Wolf, J. Prankl, S. Ghidoni, E. Menegatti, and M. Vincze, "Multi-view 3d entangled forest for semantic segmentation and mapping", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[112] J. Chen, *Dynamic laser scanning dataset for multi-view incremental segmentation*, https://github.com/jingdao/multiview_segmentation, 2018.

[113] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[114] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain", in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 4758–4765.