BRIDGE SCOUR DETECTION USING TERRESTRIAL LIDAR AND ADVANCED QUANTIFICATION TECHNIQUES

by

Navanit Sri Shanmugam

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Infrastructure and Environmental Systems

Charlotte

2022

ABSTRACT

NAVANIT SRI SHANMUGAM. Bridge Scour Detection Using Terrestrial LiDAR and Advanced Quantification Techniques
(Under the direction of Dr. Shen-En Chen)

Scour is an important factor affecting the hydraulic structures of a bridge. Remote sensing techniques such as terrestrial LiDAR (Light Detection And Ranging) can help speed up the inspection process and provide high-resolution records of the extent of scour. With LiDAR point cloud data, a temporal record of scour can be established. However, there are limitations to LiDAR scans. For example, a scan would contain not just the scour but surfaces surrounding the scour as well. Thus, there is a need to identify and separate scour points from the rest. Moreover, laser light does not bend and can be obstructed by objects along the light path resulting in missing geometric information behind the obstacles thereby creating a void in the point cloud data. To address this data void issue and to 'reconstruct' likely scour void, innovative analytical processes are being explored in this dissertation.

1. To automate scour detection and classification, 3D Point Capsule Network (3D PCN) for processing LiDAR point clouds captured from bridge hydraulic structure scans is presented. Scan results were first processed to cut portions that contain scour points. Synthetic data resembling a scour were then generated and 3D PCN, powered by a dynamic routing algorithm, was used to label the points of a given scour point cloud into scour and non-scour points. If scour is identified, it is segmented (cut) out from the point cloud for documentation.
2. To 'fill in' the missing data, spatial interpolation of 3D LiDAR point cloud data using Ordinary Kriging (OK) method is suggested and actual field data from scanning a

scoured bridge pier is presented to demonstrate the application. Kriging is a geostatistical interpolation technique and OK assumes that the spatial variation of the phenomenon or object being considered is random and intrinsically stationary with a constant mean. Here, the complete scour envelope is reconstructed using kriging.

3. Interpolation of the point cloud data can result in either extremes of data density, very dense or very sparse. A method to find an 'optimum' point-to-point distance after interpolation using processing times, surface area and volume calculations is presented.

Scanned point cloud from the Phillips Road Bridge of the Toby Creek, Charlotte, North Carolina, has been used for the study. The different processes (OK and 3D PCN) are then applied to the point cloud data set separately.

The results from the distinct methodologies are summarized as follows:

- The 3D PCN was trained to detect scour using 1,000 sets of synthetic scour data, using a split of 750-150-100 for training, evaluation, and testing.
- The resulting model had an accuracy of 63% in identifying scour points from the input scour and non-scour point cloud.
- The network performed well on a real-world point cloud from the Phillips Road Bridge pier scour.
- Data voids were identified on the same real-world scour and OK process was used to fill in those voids.
- Post-kriging spatial resolution of the points was much higher (i.e., point-to-point distances were much lower) than the original, which had varying point density in different portions of the cloud.

- Scour depth was measured, and surface areas and volumes were calculated for scenarios consisting of nine different spatial resolutions. A point-to-point distance of 20mm was found to be the optimal spatial resolution considering total processing times and comparison of the scour parameters with the actual values.

ACKNOWLEDGEMENTS

I would like to sincerely thank my PhD advisor and mentor Dr. Shen-En Chen. Without his help, advice, expertise, patience, and guidance, this research and dissertation would not have happened. I would also like to thank my committee member and mentor Dr. Rajaram Janardhanam for his guidance and encouragement. I would like to thank my other committee members: Dr. Janos Gergely, Dr. Wenwu Tang and Dr. John Diemer for their brilliant questions, insight, encouragement, and suggestions to improve my journal articles and dissertation.

I would like to thank Dr. Wenwu Tang for giving me the opportunity to work on the DeepHyd project. I would also like to thank Dr. Craig Allan, who is very much a part of the DeepHyd journey. I would like to thank the North Carolina Department of Transportation (NCDOT) for the funding support during my PhD program. I also want to acknowledge the support of NCDOT Steering and Implementation committee: John W. Kirby, Matthew Lauffer, Tom Langan, Gary Thompson, Paul Jordan, Mark Swartz, Mark Ward, Derek Bradner, Brian Radakovic and Kevin Fischer.

I would like to thank my fellow student researchers from the DeepHyd project: Vidya Subhash Chavan, Tianyang Chen, Tarini Shukla, and Zachery Slocum.

Many others have helped and supported me, I’m sincerely grateful to them – Dr. Jy Wu, Dr. Erika Weber, Dr. David Boyajian, Dr. Richard Yang, Vincent Lee, Karl Lin, Sofia Lin, Christof Boyajian, Jasmin Boyajian, Sonu Mathew.

I would like to thank my family – my father (Shanmugam S.) and mother (Kavitha K.N.) for their sacrifice, continued support and understanding during my PhD program. A special word of thanks to my aunt (Agilandeswari N.) and uncle (Natrajan P.) for being there with me during this journey.

DEDICATION

To my family, teachers, and friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

2D	Two Dimension
3D	Three Dimension
AI	Artificial Intelligence
ASCE	American Society of Civil Engineers
CN	Capsule Network
FHWA	Federal Highway Administration
GB	Giga Bytes
IDW	Inverse Distance Weighted
LiDAR	Light Detection and Ranging
LOS	Line of Sight
m	Meters
MATLAB	Matrix Laboratory
MB	Mega Bytes
MLP	Multi-Layer Perceptron
NBI	National Bridge Inventory
NCHRP	National Cooperative Highway Research Program
nm	Nano meters
OK	Ordinary Kriging
PCN	Point Capsule Network
RK	Residual Kriging
SK	Simple Kriging
UAS	Unmanned Aerial Systems

UK	Universal Kriging

1 INTRODUCTION

1.1 Motivation and Background

Scour is the primary cause of bridge failures in the United States and is responsible for almost 60% of the bridge failures (Melville & Coleman, 2000). About 83% of the structures listed in the National Bridge Inventory (NBI) cross waterways and are exposed to the threats of flooding and scour. During the last 30 years, 600 bridges have failed due to scour problems (Shirole & Holt, 1991).

Scour can compromise the structural integrity of a bridge structure by removing soil and sand surrounding a bridge component (Lagasse et al., 2012). It is caused primarily by fast moving water. The most concerning fact about scour failures of bridges is that they can occur without any warning. So, monitoring scour surrounding a bridge structure is necessary and the monitoring techniques must be effective. River flow hydrodynamic characteristics like water velocity and the integrity of the surrounding geomaterial of a bridge pier dictate the scour process and define the extent of scouring problem.

To monitor scour conditions, several measurement techniques have been suggested and can be distinguished into static and dynamic measurement techniques (Prendergast & Gavin, 2014). Terrestrial LiDAR is a popular condition monitoring technique – Based on the detection of returned laser beams, the technique has been used for bridge monitoring needs such as detection of condition changes (Liu et al., 2012; Liu & Chen, 2013). In this research, LiDAR scanning of bridge hydraulic structures is proposed as a scour monitoring technique. LiDAR facilitates the collection high-resolution point cloud data of a bridge hydraulic structure. This data can then be used to quantify material losses. Laser scans taken over a period of time can generate periodic data and thus, periodic quantification of scours can be done. This would help determine the rate of removal or addition of streambed or bank material surrounding the bridge foundation.

The severity of a scour is typically determined using scour depth; however, area and volume can also be established as scour parameters. Since a bridge environment (i.e., bridge hydraulic structure geometry, bank surface) is highly irregular, it is difficult to automate the scour identification process. This study focuses on scour detection and quantification using deep learning algorithm, specifically the 3D Point Capsule Network (PCN) approach (Zhao et al., 2019). The scour detection is identified as a classification problem for scour and non-scour areas for given a rough set of points of scour and its surroundings. To find the scour volume, the points in the point cloud is first classified either as a scour or a non-scour using 3D PCN, then, the points are analyzed to calculate the scour parameters. Contrast to conventional neural networks that uses scalar values for feature representation, 3D PCN uses capsulated vectors to represent features, which enables better feature recognition capabilities. 3D PCN has been used successfully for applied to spatial problem types including local part extraction, 3D-feature reconstruction, and object classification. An advantage of using 3D Point Capsule Network is that it is not affected by translation, rotation, or the scale of point cloud data, which is advantageous if real world data is used.

However, there are limitations on LiDAR scans: Since LiDAR uses light, which does not bend, may be obstructed by an object along the light path and resulting in interference (Munoz Rodriguez, 2012). As a result, obstacles make the surfaces behind the obstacle ‘invisible’ to the LiDAR device and create a void in the point cloud data. A typical solution is to scan the structure at multiple angles and stitch the images to generate the full image. Due to the complexity and size of the scoured area, sometimes, even multiple scans will not be able to fully describe the full scoured void.

To address the issue of voided data due to obstacles, this study suggests the use of spatial interpolation: Spatial interpolation is the process to estimate values of locations that were not surveyed using the data from the network of known points (Hohn, 1991). There are several spatial interpolation techniques including both deterministic and statistical methods (Cressie, 1993). In particular, Kriging is useful for the population of voided data (Meng et al., 2013). In the case of complex and under-sampled problems, co-Kriging using auxiliary variables can be useful to populate the voided data (Knotters et al., 1995).

In this study, ordinary kriging (OK) will be used to populate missing scour geometry due to LiDAR scan unable to reach areas beyond the line-of-sight. Data from an actual bridge is used for the study - The Phillips Road Bridge at UNC Charlotte is recognized to have scouring issue. LiDAR scans were performed around the bridge piers experiencing soil mass losses (scour) over the span of three years.

1.2 Research Objectives

The objectives of this research, therefore, are:

1. To scan local scour on a case study bridge using terrestrial LiDAR.
2. To demonstrate application of AI in scour detection by using 3D Point Capsule Network to detect scour in a LiDAR point cloud.
3. To evaluate LiDAR scan on bridge scour problem, in particular the issue of point cloud missing data points by spatially interpolating the point cloud data using kriging.
4. To analyze the effects of spatial point density from the interpolated results.

1.3 Methodology

This dissertation will be presented as the summary of three journal papers. The methodology for this research is summarized in the figures in this section. Scanned point cloud from the Phillips Road Bridge of the Toby Creek, Charlotte, North Carolina, has been used for the study. The different processes (OK and 3D PCN) are then applied to the point cloud data set separately.

Synthetic data was created to train the AI in Chapter 2. Terrestrial LiDAR was used to scan the case study bridge. It was processed and then input into the network to evaluate its performance. Spatial interpolation using OK was used on the same scour point cloud to fill in the missing data voids in Chapter 3. Interpolated results at various point densities were analyzed to get an 'optimum' resolution for the point cloud in Chapter 4. Figure 1-1 to Figure 1-3 show the methodology schematics for Chapters 2-4 respectively. The overall methodology to automatically get scour parameters with a given 'near scour' point cloud as input is shown in Figure 1-4.

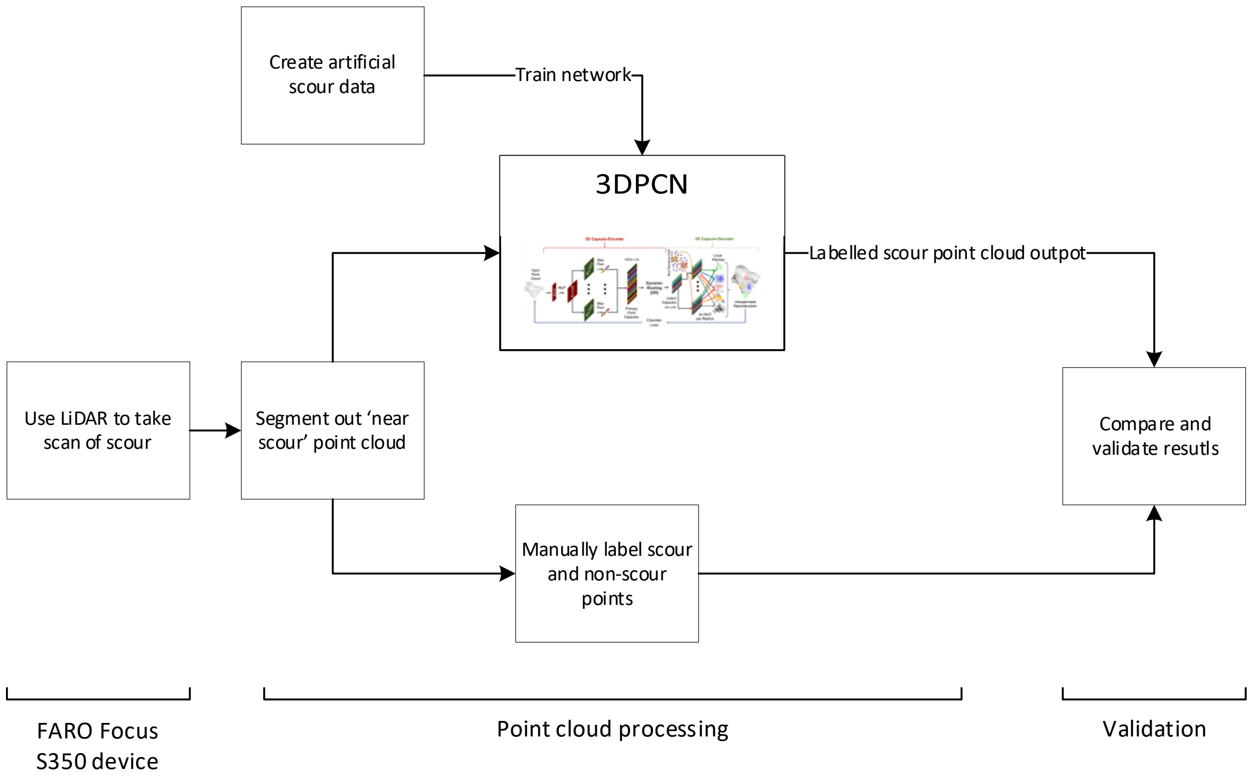

Create artificial scour data
Train network
3DPCN
Labelled scour point cloud outpot
Use LiDAR to take scan of scour
Segment out 'near scour' point cloud
Compare and validate resutls
Manually label scour and non-scour points
FARO Focus S350 device
Point cloud processing
Validation

Figure 1-1 A basic methodology schematic for Chapter 2

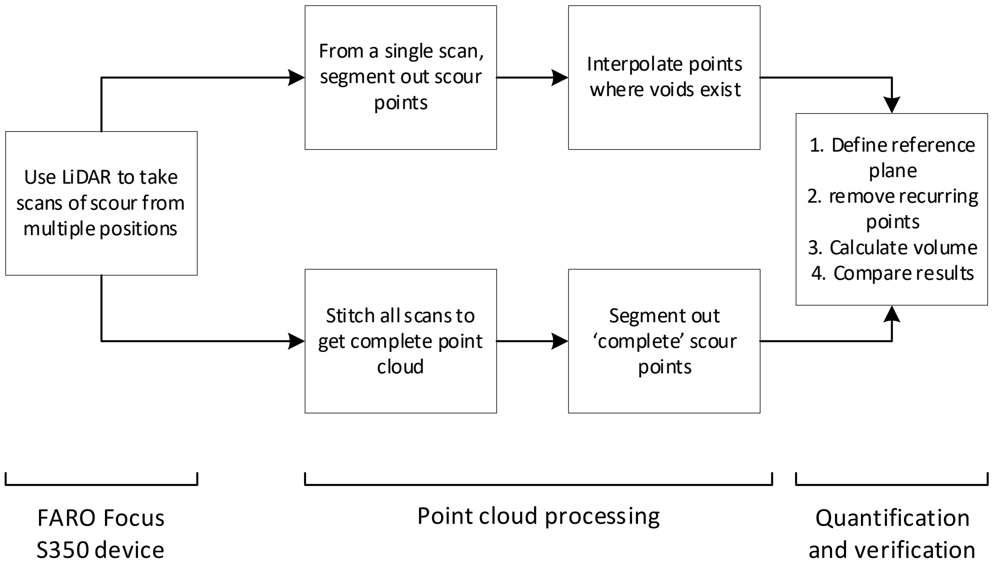

From a single scan, segment out scour points
Interpolate points where voids exist
Use LiDAR to take scans of scour from multiple positions
1. Define reference plane
2. remove recurring points
3. Calculate volume
4. Compare results
Stitch all scans to get complete point cloud
Segment out 'complete' scour points
FARO Focus S350 device
Point cloud processing
Quantification and verification

Figure 1-2 A basic methodology schematic for Chapter 3

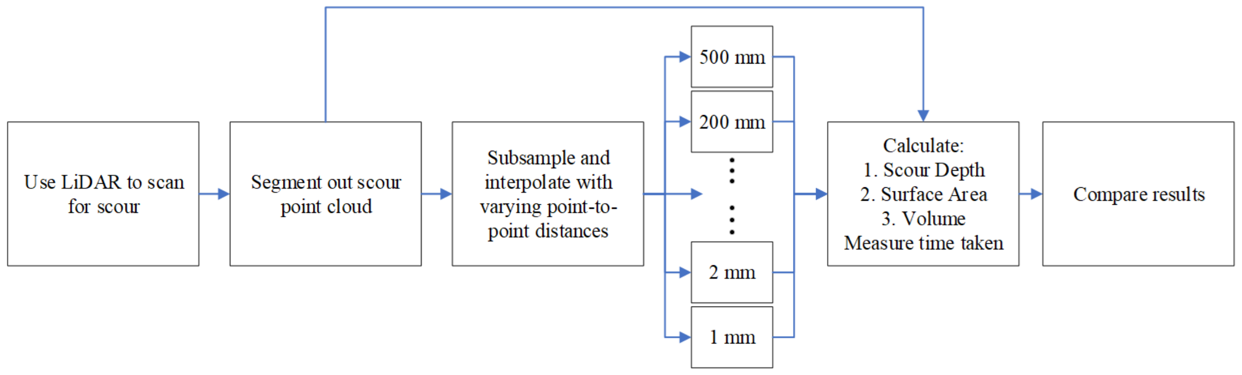

Use LiDAR to scan for scour
Segment out scour point cloud
Subsample and interpolate with varying point-to-point distances
500 mm
200 mm
2 mm
1 mm
Calculate:
1. Scour Depth
2. Surface Area
3. Volume
Measure time taken
Compare results

Figure 1-3 A basic methodology schematic for Chapter 4

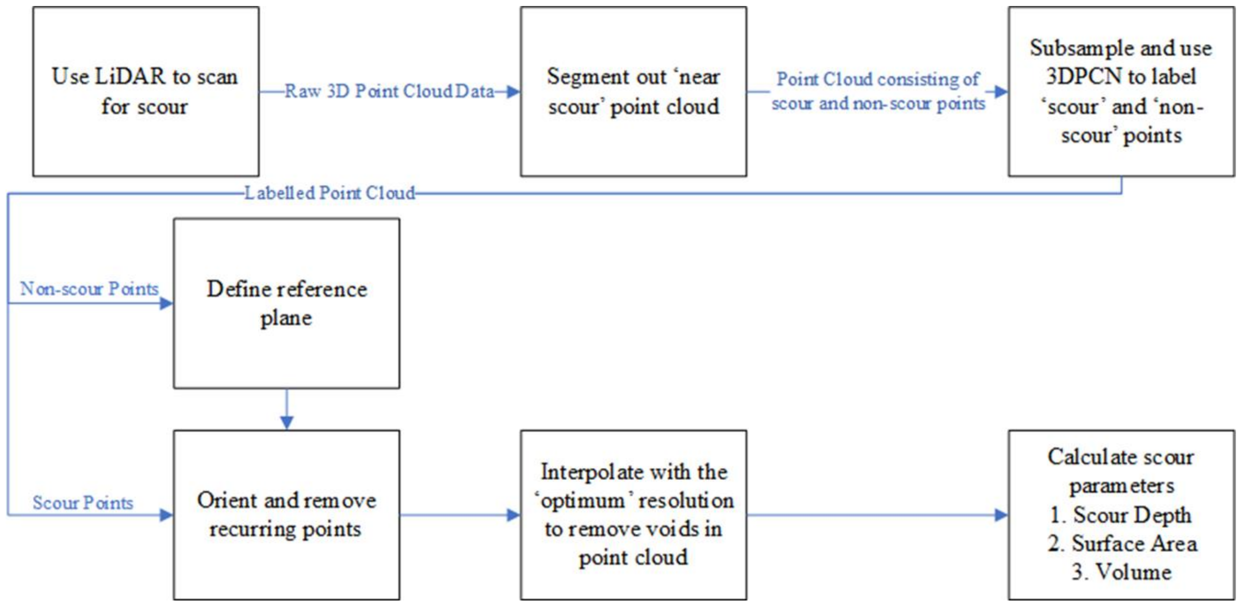

Use LiDAR to scan for scour
Raw 3D Point Cloud Data
Segment out 'near scour' point cloud
Point Cloud consisting of scour and non-scour points
Subsample and use 3DPCN to label 'scour' and 'non-scour' points
Labelled Point Cloud
Non-scour Points
Define reference plane
Scour Points
Orient and remove recurring points
Interpolate with the 'optimum' resolution to remove voids in point cloud
Calculate scour parameters
1. Scour Depth
2. Surface Area
3. Volume

Figure 1-4 Overall algorithm for this research

1.4 Scope

The scope of this research is limited to point cloud scour detection, interpolation, and the analysis of resulting spatial point density from the scanned scour image of the Phillips Road Bridge. In this study, we have explored two different advanced point cloud processing techniques namely Ordinary Kriging and the 3D Point Capsule Network. It is appreciated that scour problem for a bridge can vary significantly, hence, current study is not all encompassing. Furthermore, this study focuses on the pier-on-bank type of bridge, which can behave differently from pier in the river channel. The following are a summary of the specific scope conducted in current study:

1. Point cloud scour detection
    a. Use of 3D Point Capsule Network deep learning algorithm with 2048 points.
    b. Use a real-world scour to check results – Case from Phillips Road Bridge.
2. Spatial Interpolation of bridge pier scour point cloud data
    a. Use of Ordinary Kriging
    b. Local scour found in LiDAR data from Phillips Road Bridge.
3. Analysis of point density after scour point cloud spatial interpolation
    a. Scour depth, surface area and volume calculations

1.5 Dissertation Outline

Following this introduction, the dissertation includes chapters and appendices. Chapters consist of to-be-published peer-reviewed journal articles. Chapter 2 discusses the use of a novel artificial intelligence (AI) approach to label scour and non-scour points in a point cloud. As there could be voids present in a point cloud, Chapter 3 is an article presenting a spatial interpolation method to fill in the spatial data voids in a scour point cloud. Chapter 4 describes a more in-depth investigation of the relations between OK interpolated spatial density and the filling in of selected data voids. Overall conclusions from this research work and scope for future research are given in Chapter 5 and Chapter 6 respectively. Appendices include select photos of bridge pier, scour parameter results over time and computer codes.

2 USING 3D POINT CAPSULE NETWORK FOR POINT CLOUD SCOUR DETECTION

2.1 Abstract

Scour is an important factor affecting the hydraulic structures of a bridge. Remote sensing techniques such as terrestrial LiDAR (Light Detection And Ranging) can help speed up the inspection process and provide high-resolution records of the extent of scour. To automate scour detection and classification, this paper introduces the 3D Point Capsule Network (3D PCN) for processing LiDAR point clouds captured from bridge hydraulic structure scans. Scan results were first processed to cut portions that contain scour points. Synthetic data resembling a scour were then generated and 3D PCN, powered by a dynamic routing algorithm, was used to label the points of a given scour point cloud into scour and non-scour points. If scour is identified, it is segmented (cut) out from the point cloud for documentation. The existing method to achieve this requires manually opening the point cloud file and labeling the points as scour and non-scour. The work in this paper is a novel application for automated classification and segmentation of a scour point cloud, which significantly reduces the time taken for such processing.

Keywords: Bridge Scour Identification, LiDAR Point Cloud, 3D Point Capsule Network

2.2 Introduction

Scour, caused by swiftly moving water, can remove sand and soil, creating holes surrounding a bridge component and compromising the integrity of a structure. Scour has been identified as the primary cause of bridge failures in the United States responsible for almost 60% of the bridge failures (Melville and Coleman 2000). About 83% of the structures listed in the National Bridge Inventory (NBI) span waterways and are exposed to the threats of flooding and scour (ASCE 2017). The danger of bridge scour failures lies in the fact that they can occur without warning (Shirole and Holt 1991). Thus, there is a need for effective monitoring techniques for scour problems surrounding bridge structures. Figure 2-1 shows the mechanisms of scouring surrounding a bridge pier. The hydrodynamic characteristics of the river flow and the integrity of the surrounding geomaterial of a bridge pier dictate the scouring process and define the extent of the scour problem.

The scour shown in Figure 2-1 is typically considered as a 'local' scour which occurs around individual bridge piers and is caused by an acceleration of flow and resulting vortices induced by obstructions to the flow. Scouring removes the soil support around the bridge pier, thus reducing the soil bearing capacity. Scour alters static and dynamic characteristics of the bridge structure and may lead to excessive structural settlement and may result in load redistribution in structural members ultimately leading to failure of the bridge structure.

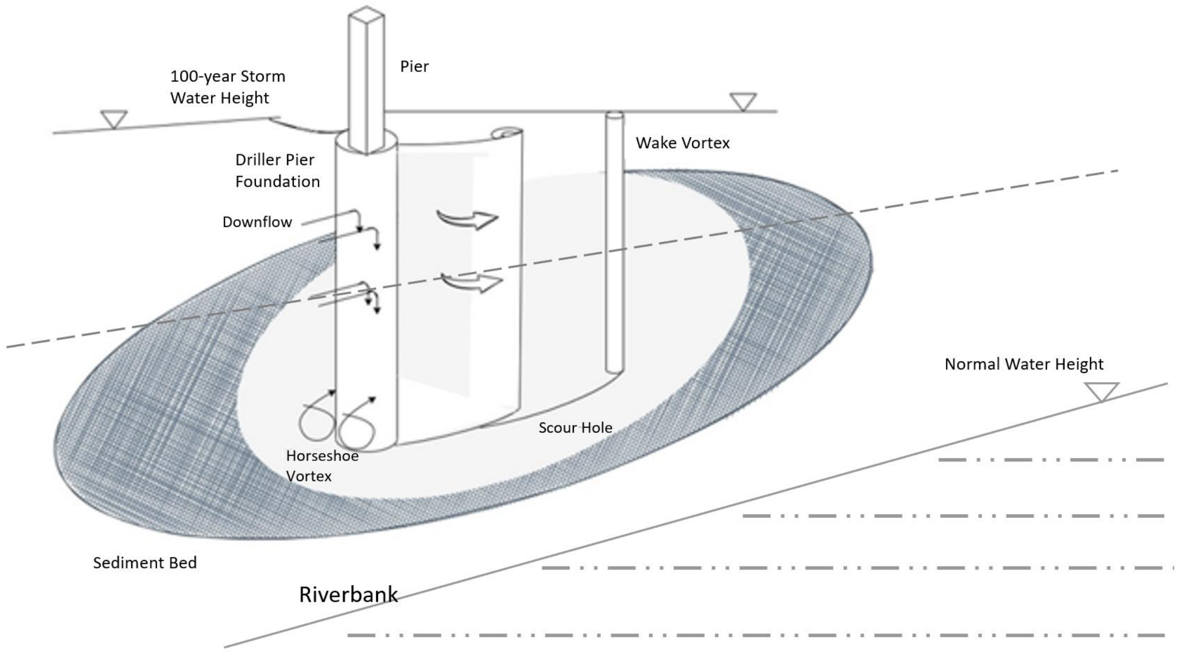


Figure 2-1 Mechanism of scour at a circular pier (Chavan et al. 2022)

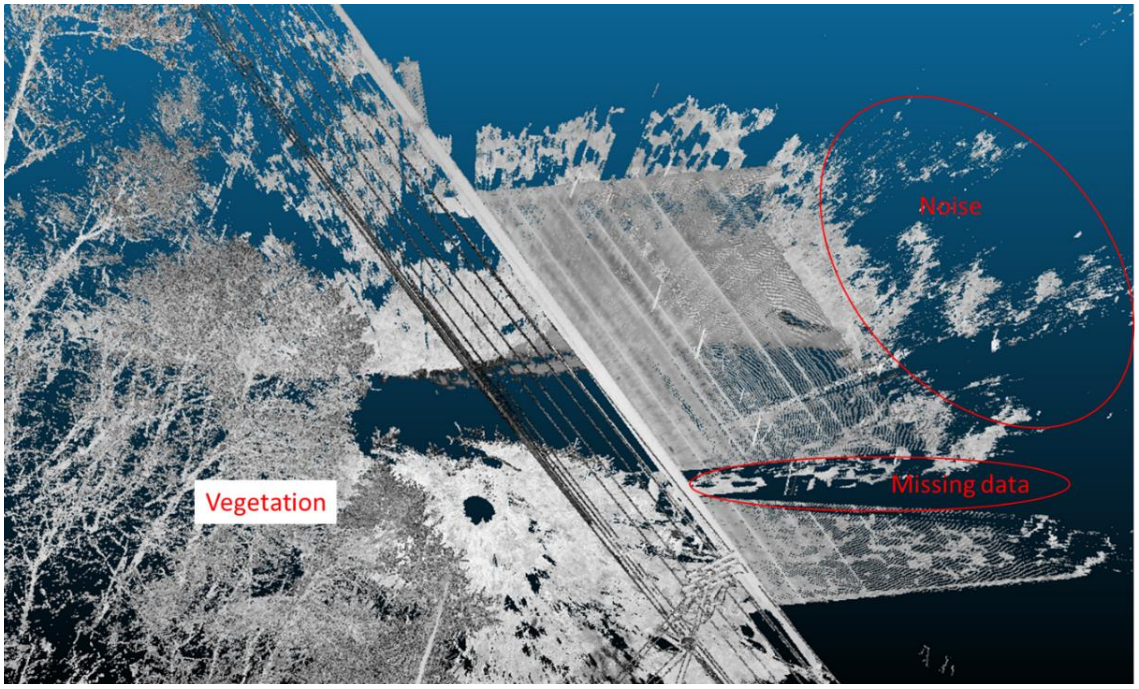


Figure 2-2 A Point Cloud of a Bridge with Missing Data, Vegetation and Noise Marked

Bridge scour is combatted in many ways, including through hydraulic and structural countermeasures (NCHRP, 2009). However, these have limitations and uncertainties. In addition, usually they can be implemented only on new structures during the design stage of a bridge. A more effective and economically viable method for existing structures is to monitor scour evolution over time (Briaud et al., 2011) and implement the required remediation works.

To monitor scour conditions, terrestrial LiDAR is suggested as a technique to quantify the mass loss due to bridge hydraulic condition changes (Chavan et al. 2022). LiDAR (Light Detection And Ranging) is a surveying technique that uses reflected pulsed laser light to make 3D representations of the target (Liu et al. 2012). The representations are in the form of a 3D point cloud, a set of digital points (XYZ coordinates). A LiDAR consists of a laser emitter, which emits a concentrated beam of light. The emitted laser then gets reflected back to the LiDAR device after hitting an object. Using the time difference between the emission and detection, the distance between the device and the object can be calculated. Based on the detection of returned laser beams, the technique has been used for bridge monitoring needs such as the detection of scour formation (Chavan et al. 2022) and condition changes (Liu et al. 2012, 2013). Rapid and repeated laser scans can generate periodic quantification of scours and help define the process of erosion and determine the rate of removal of streambed or bank material surrounding the bridge foundation. The severity of a scour is typically quantified using scour depth, however, scour area and volume can also be established as scour parameters.

Since lasers cannot penetrate solid objects, the area around the scour of interest should always be cleared along the line-of-sight (LOS). Furthermore, a data void or a 'shadow' is unavoidable for any single scan as the light cannot reach the regions behind the object. It is also inevitable to capture some obstacles (such as trees, shrubs, etc.) in the point cloud, which are collectively identified as noise, and noise removal must be performed. Figure 2-2 shows a point cloud with marked noise and missing data. The point cloud has a total of about 70 million points.

We have adopted a deep learning algorithm for the detection of bridge hydraulic structures (Tang et al. 2022) and this paper focuses on scour detection and quantification using a deep learning algorithm, specifically the 3D Point Capsule Network (PCN) approach (Zhao et al. 2019). Since a bridge environment (i.e., bridge hydraulic structure geometry, bank surface) is highly irregular, it is difficult to automate the scour identification process. Currently, there are no known published works with the objective of classifying point clouds of any local scour surface. The scour detection is recognized as a classification problem between scour and non-scour areas for an unedited LiDAR scan of bridge pier scour and its surroundings. To find the scour volume, the points in the point cloud are first classified either as a scour or a non-scour using 3D PCN. Then, the points are analyzed to calculate the scour parameters such as the scoured area.

To summarize, the contributions of the authors in this work are as follows:

- Application of 3D PCN as an artificial intelligence approach to label bridge pier local scour points in a point cloud
- Generation of a synthetic dataset consisting of 1000 scour-like point clouds.
- Evaluating 3D PCN on real data to demonstrate the generalization of the approach.

2.3 Point Clouds in Deep Neural Networks

Point clouds are preferred for many 3D applications due to their capabilities in explaining 3D data without assuming one modality (Zhao et al., 2019). Point cloud-specific architecture algorithms must be invariant to permutations of the input set, invariant to rigid transformations and should capture the interaction between points. PointNet (Qi et al., 2017), PointNet++ (Qi et al., 2017), and others (Li et al., 2018; Liu et al., 2018; Lei et al., 2018; Hermosilla et al., 2018), have exploited these properties. It is also possible to process point sets by taking projections to reduce the operation to two dimensions.

2.4 3D Point Capsule Network

Capsulated network is a modification of the classical deep learning techniques using neurons to obtain the vector representations of desired entities. Capsule Networks (CNs) have found many uses in 2D deep learning including object segment, classification and 2D image generation (LaLonde and Bagci, 2018; Durage et al., 2018; Jaiswal et al., 2018; Saqur et al., 2018; Upadhay et al., 2018). In contrast to conventional neural networks that use scalar values for feature representation, 3D PCN uses capsulated vectors to represent features, which enables better feature recognition capabilities. 3D PCN is used successfully for applications to spatial problems including local part extraction, 3D-feature reconstruction, and object classification (Xiang et al 2018; Ma et al. 2020). An advantage of using 3D Point Capsule Network is that it is not affected by translation, rotation, or the scale of point cloud data, which is critical if real world data are used (Xiang et al. 2018).

Several modified capsulated networks have been proposed for enhancing image processing capabilities (Xiang et al. 2018; Deng et al. 2018; Zhu et al. 2019). However, for LiDAR point cloud data processing, the 3D PCN remains the standardized approach (Park et al. 2020). Developed by Zhao et al. (2019), the 3D PCN can be summarized as a set of capsule-encoder and decoder couples that enable deep mapping through dynamic routing. Instead of point-representation, the capsule representation allows high-level feature vectorization and rapid convolution operations. The dynamic routing suggests that cascading evaluation of capsule-representations can be performed at higher levels (latent capsule mapping). Sabour et al. (2017) described the dynamic routing algorithm as the evaluation of different capsules with different coupling coefficients. In our study, the 3D-PointCapsNetwork (Zhao et al. 2019) is used for processing the LiDAR point cloud captured from a bridge pier for scour detection and classification.

Figure 2-3 summarizes the 3D-PointCapsNetwork architecture and the components of the network are described as follows:

Encoder: PointNet-like (Qi et al. 2017) layers are used in the 3D PCN, where the network input is a 2048x3 point cloud. A point-wise Multi-Layer Perceptron (MLP) (3-64-128-1024) is then used to extract local feature maps. These feature maps are then fed into multiple independent convolutional layers with different weights, each with a distinct summary of the input shape with diversified geometries. The responses are max pooled in order to obtain a global latent representation. These descriptors are concatenated into a set of vectors named primary point capsules. Dynamic routing is then used to embed the primary point capsules into higher-level latent capsules. Each capsule is independent and is considered as a cluster centroid of the primary point capsules. The total size of the latent capsules is fixed as 64x64.

Decoder: The decoder treats the latent capsules as a feature map and uses the MLP (64-64-32-16-3) to reconstruct a patch of points. The entire capsule is replicated several times and a synthesized grid is appended specifying a local area (similar to the FoldingNet, Yang et al. 2018). These patches are then glued together.

Loss function: The Discrete Chamfer metric (Zhao et al., 2019) is used to approximate the training loss over predicted and ground truth point cloud.

Optimizer: A first order gradient based stochastic objective function optimizer algorithm called Adam (Kingma and Ba, 2014) with a learning rate of 0.01 for the first 30 epochs, 0.001 for epochs less than 50, and 0.0001 for epochs beyond 50.

One hot encoding (Harris and Harris, 2012): This step is done so that point cloud labels do not affect the network with their values. Non-scour and scour points are represented by the numeric ‘1’ and ‘2’, respectively. Although the number 2 is greater than 1, one-hot encoding makes the labels ‘scour’ not ranked higher or lower than ‘non-scour’.

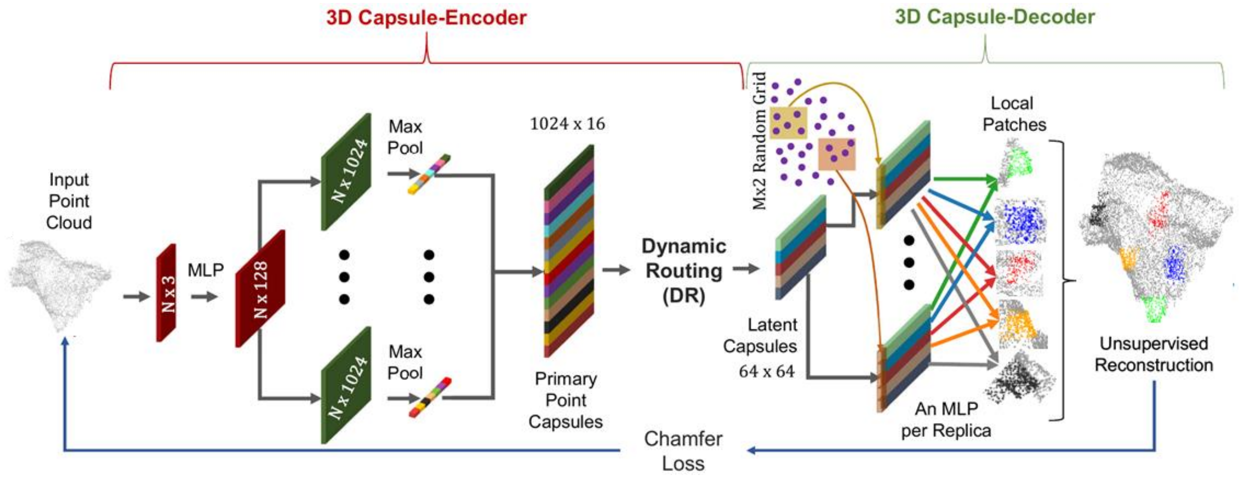


Figure 2-3 3D-PointCapsNetwork (3D PCN) Architecture (Zhao et al., 2019)

2.5 LiDAR Point Cloud Training Using 3D PCN

To verify and validate the 3D PCN algorithm for detecting scour, synthetic data resembling a scour were created using MATLAB. A 1,000 x 1,000 point matrix was created and five Gaussian depressions/peaks with random amplitudes were introduced. Each depression had a maximum amplitude of 50 units. This resulting point cloud was randomly sampled to obtain 2,048 points (to match the 3D PointCapsNet). Then, the points were classified into scour/non-scour based on their height metric. The XY plane was set as the ground plane. Thus, the Z coordinate determines if a point is scour or non-scour. The XYZ coordinates and the scour classification were then saved into a different data file (required by PointCapsNet). One thousand different randomly generated point clouds were used in this study.

Out of this dataset, 750 point clouds were used to train the 3D PCN. While testing, the network did not detect scour – all the test points were identified as non-scour. An analysis of the results indicated that the scour depth was not large enough for the capsules to differentiate between scour and non-scour cases. Figure 2-4 shows the results from the two sets of artificial data. The one on the left with clearly marked scour and non-scour in the original data.

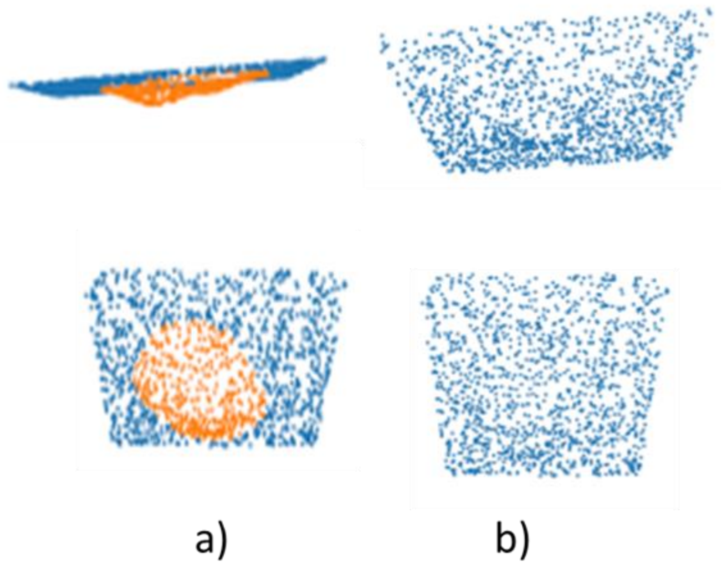


Figure 2-4 Scour Point Clouds. a) Two sets of manually labelled input b) Output result showing incorrectly labelled points

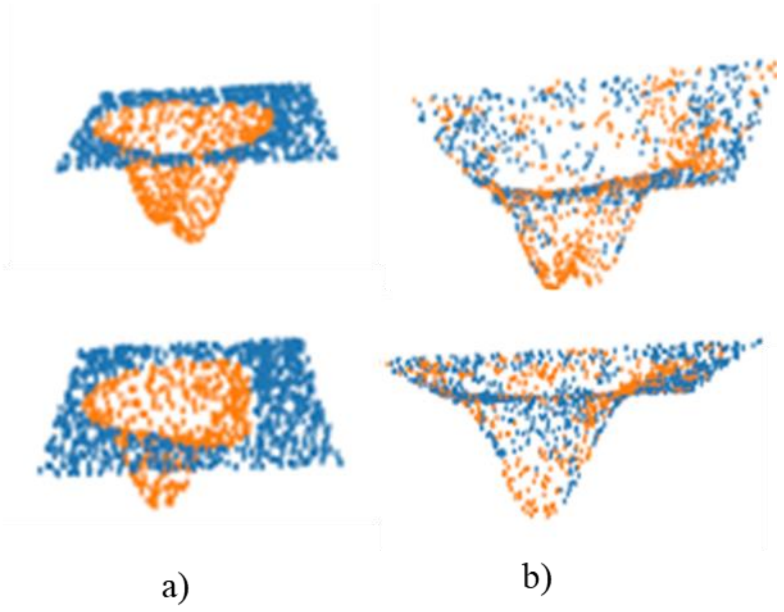


Figure 2-5 Result Comparison a) Manually labeled point cloud b) AI labeled point cloud

Another 1,000 artificial sets of data were then generated and used to train the model, this time with a maximum peak depth of 200 units for each depression/peak. The network performed better with a detection accuracy of about 63%. Figure 2-5 shows the two sets of data (original and scour detected point clouds). This shows that the scour points are used to construct some of the non-scour points and some non-scour points are used to construct scour points.

2.6 Case Study

The 3D PCN was applied to a scoured bridge pier that belongs to the Phillips Road Bridge over Toby Creek (35°18'28.2"N 80°44'16.6"W) in northern Charlotte, North Carolina, USA. The newly constructed Phillips Road bridge was opened on March 12, 2016. The bridge consists of three spans supported on prestressed concrete girders and drilled pier foundations (Chavan et al. 2022). Each pier consists of a reinforced concrete column supported on a concrete drilled pier foundation driven through four layers of different types of soil and fill material. Figure 2-6 shows the scoured bridge pier and Figure 2-7 depicts the LiDAR scan of the scoured bridge pier.

To detect the scour, a FARO Focus S 350 scanning laser was used. The FARO scanner uses a near infrared laser of 1,550 nm wavelength and has a shooting range of 350 m. After the point cloud was captured, the data must be processed before the determination of the scour. There are several steps needed to process the raw point cloud data including data stitching, which may be required when multiple scans are performed at a site. In cases where multiple scans were performed at a site to capture the object from different angles, the multiple point clouds were combined (stitched) together into a single point cloud with a single global coordinate system for all scans.

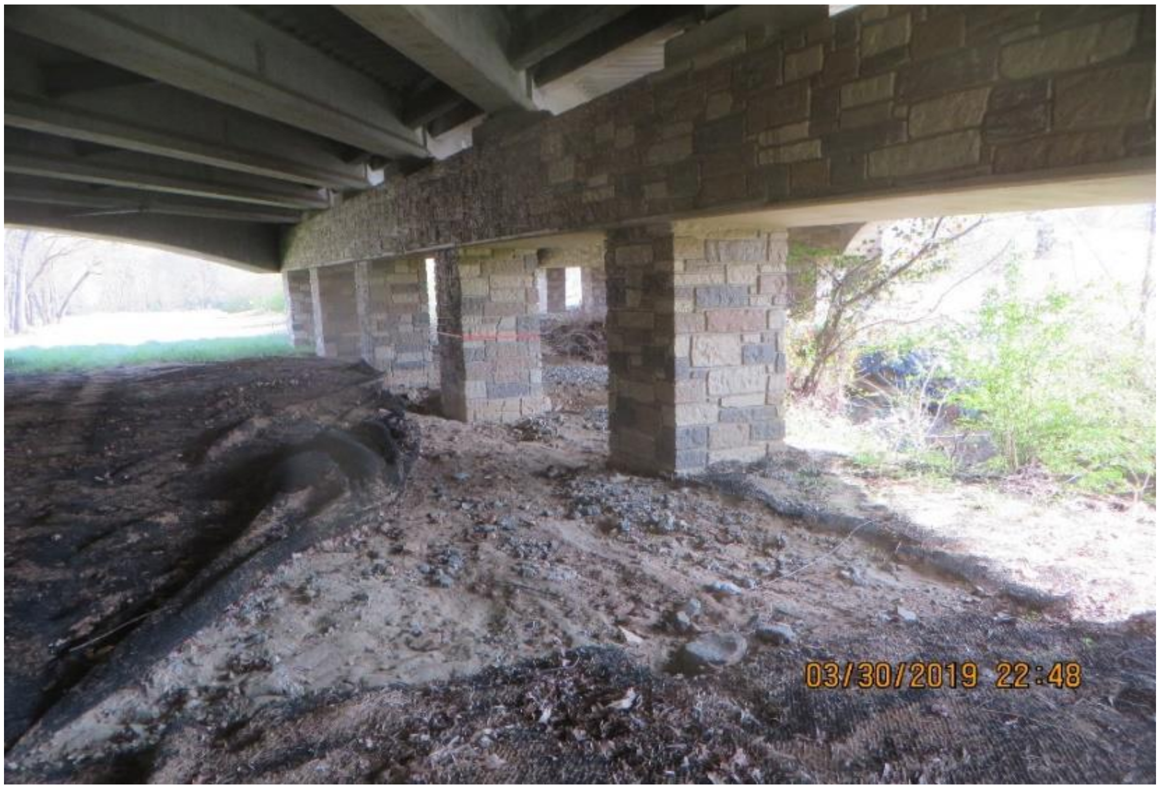
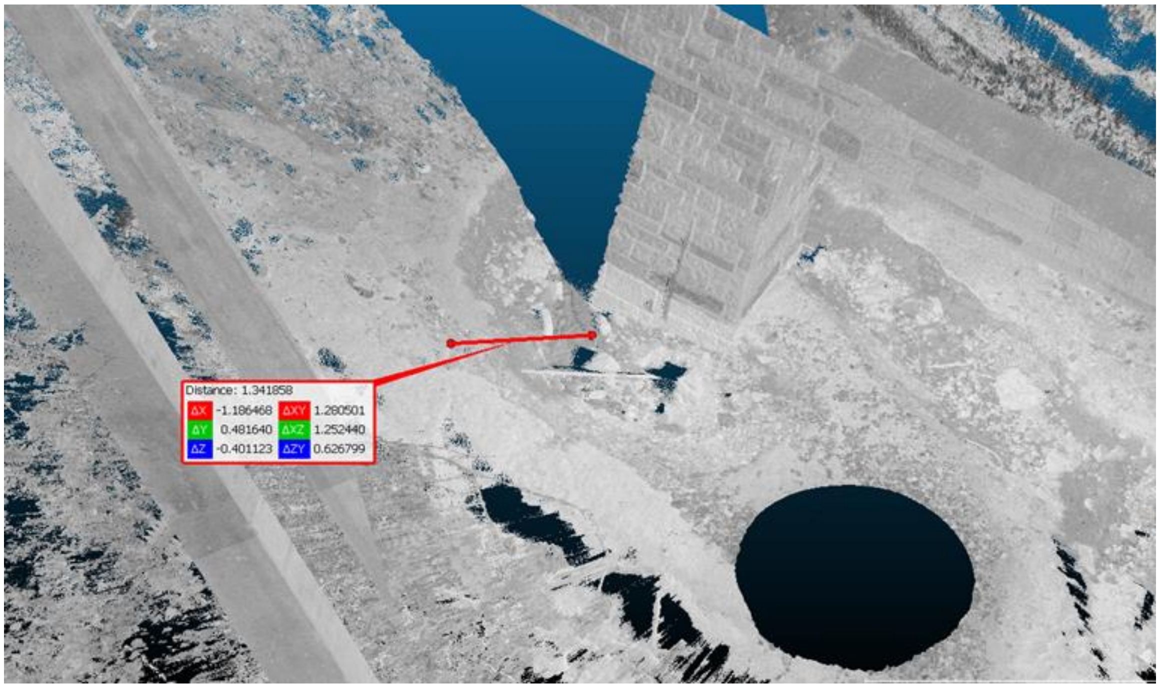


Figure 2-6 Scour surrounding piers on the north bank of Toby Creek



```
Distance: 1.341858
ΔX  -1.186468   ΔXY  1.280501
ΔY   0.481640   ΔXZ  1.252440
ΔZ  -0.401123   ΔZY  0.626799
```

Figure 2-7 LiDAR scan of the scour around one of the north bank piers of Philipps Road Bridge

The stitched point cloud is then segmented to reduce the size of the data file and also to isolate the scoured areas. The processed data can then be used for scour detection. The procedure of the point cloud analysis can be summarized as follows:

1. Identify scour surrounding the pier and scan the scour.
2. Transfer data and segment the point cloud to include only the points containing scour and its surrounding region.
3. Export the point cloud and use the 3DPCN algorithm to label scour.

2.7 Discussion

While training, the weights for the neural network were adjusted with the aim of minimizing an objective function. An epoch is when the entire training dataset passes through the neural network once. Training loss is the value of the objective function to be minimized. A whole point cloud dataset could not pass through the network all at once, rather the dataset was split into batches and thus, the value of the objective function was updated multiple times in a single epoch. Figure 2-8 shows the average training loss with respect to number of epochs. The average training loss stabilized at around 0.001222 corresponding to the 86th epoch. The training was stopped at epoch 97 after 10 additional epochs resulted in very small variations in the training loss.

The next step in this study would be to increase the number of scour points in the dataset. Currently, there is a huge difference in the distance between the scour points and the distance between non-scour points. Effects of changing the network parameters, like decreasing the number of latent capsules, would be interesting to see. This network was originally trained and tested on the Shapenet-Part dataset, consisting of 50 different parts (Chang et al., 2015). However, for the present scour case, only two conditions, scour or non-scour are considered.

There are many challenges associated with scour. The first is that scour is not always close to the structure. Second, a well-defined geometry is often not present. Scour can result in many shapes and sizes. It also important to note that the existence of scour does not necessarily mean structural instability. The Phillips Road bridge is an example of such a case, as scour is present but at present the structure is not unstable (Chavan et al. 2022). Another challenge is that scour can vary over time. Depending on the weather and flow conditions, over time more material can be washed away from, or material can be deposited, at least temporally filling an existing scour.

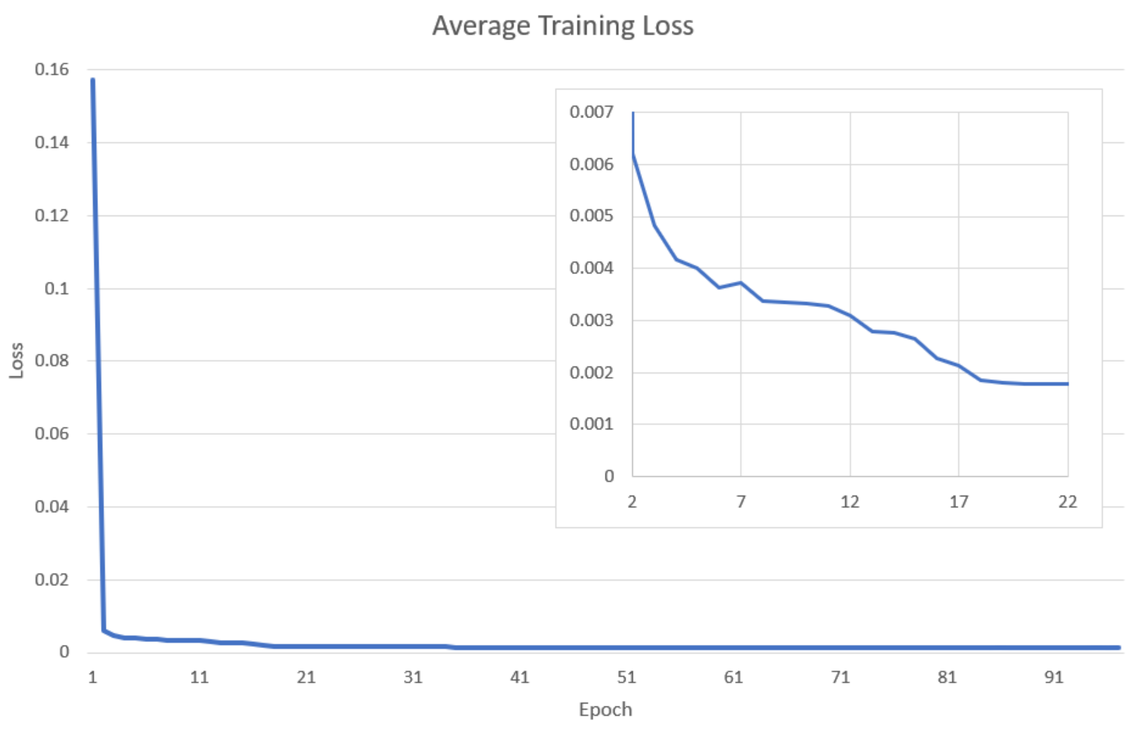


Figure 2-8 Average Training Loss

There are also technical challenges associated with the analysis of point clouds generated from LiDAR scans. One real-world LiDAR scan can generate millions of points, resulting in more than 700MB of data in compressed file format. For example, a single Phillips Road Bridge scan consists of 26.7 million points. Converting the scans to obtain XYZ point data format - which is used in PCN - would result in more than 6GB of data. Processing them would take considerable time/resources. A system with at least 64GB of RAM, CPU with 8 physical cores and a solid-state drive is recommended for processing a LiDAR point cloud project. Data voids in scans are another challenge. Currently, it is difficult for a machine to tell if the void is due to a defect or due to an object blocking the laser.

2.8 Conclusions

In this paper, 3D PCN method has been used to reconstruct scour shape from a LiDAR point cloud. To train the network, 750 synthetic data using Gaussian depression/compression sets of random amplitudes were generated. 150 sets of synthetic data were used to evaluate the training model. While training, the loss function rapidly decreased from 0.157 to 0.006 in just 2 epochs. However, it took 86 epochs to reduce to 0.001 (Figure 2-8). One hundred datasets were used for testing the final network model. The results of identifying scour and non-scour points averaged at 63% accuracy. This trained network was also applied to a real-world point cloud – the case study scours from Phillips Road Bridge and showed favorable results (Figure 2-9).

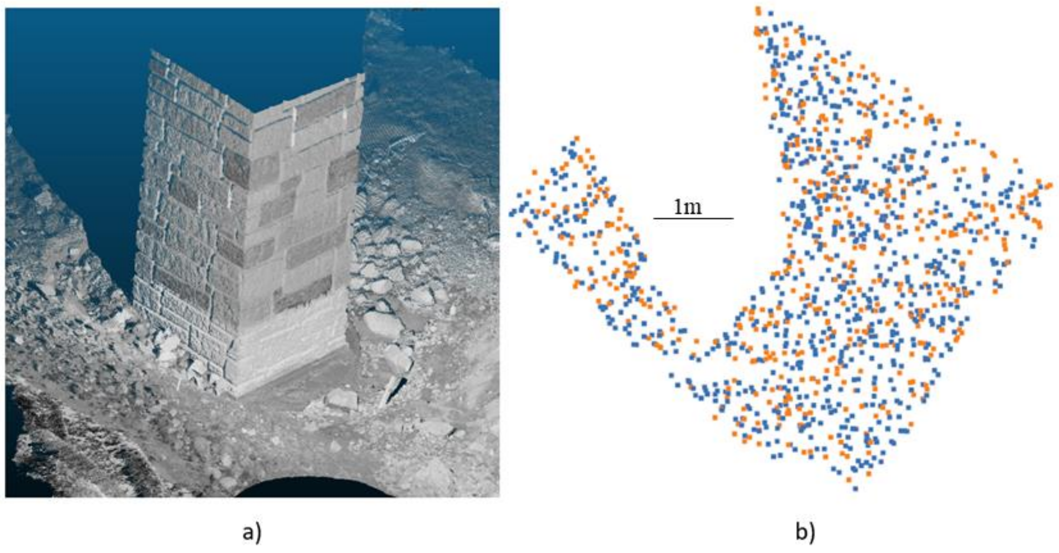


Figure 2-9 Phillips Road Bridge point cloud a) original point cloud showing scour around pier b) 3DPCN output of the scour point cloud – pier was removed before input

2.9 References

ASCE, (2017), Infrastructure Report Card, https://www.infrastructurereportcard.org/cat-item/bridges/.

Briaud, J. L., Hurlebaus, S., Chang, K. A., Yao, C., Sharma, H., Yu, O. Y., Darby, C., Hunt, B.E., and Price, G. R. (2011). Realtime monitoring of bridge scour using remote monitoring technology (No. Report 0-6060-1). Texas Transportation Institute, http://tti.tamu.edu/documents/0-6060-1.pdf

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

Chavan, V.S., Chen, S.E., Shanmugam, N.S., Tang, W., Diemer, J., Allan, C., Braxtan, N., Shukla, T., Chen, T. and Slocum, Z., (2022a) "An Analysis of Local and Combined (Global) Scours on bridge Piers-on-Bank," CivilEng, 3, doi.org/10.3390/civileng301001.

Deng, F., Pu, S., Chen, X., Shi, Y., Yuan, T. and Pu, S. (2018) "Hyperspectral Image Classification with Capsule Network Using Limited Training Samples," Sensors, 18, 3153.

Harris, D.M. and Harris, S.L. (2012). "Digital design and computer architecture (2nd ed.),". Morgan Kaufmann. p. 129.

Hermosilla, P., Ritschel, T., Vazquez, P.P., Vinacua, A. and Popinski, T. (2018) "Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds." Proceeding, Conference SIGGRAPH Asia, ACM, 235.

Jaiswal, A., AbdAlmageed, W., Wu, Y. and Natarajan, P., (2018) "Capsulegan: Generative Adversarial Capsule Network." European Conference on Computer Vision, 526–535. Springer.

Kingma, D. P., & Ba, J. (2014). "Adam: A Method for Stochastic Optimization." arXiv:1412.6980.

LaLonde, R. and Bagci, U. (2018) “Capsules for Object Segmentation.” arXiv:1804.04241.

Lei, H., Akhtar, N., and Mian. A. (2018) “Spherical Convolutional Neural Network for 3d Point Clouds.” arXiv: 1805.07872.

Li, J., Chen, B.M. and Lee, G.H. (2018) “So-Net: Self Organizing Network for Point Cloud Analysis,” Proceeding, IEEE Conference on Computer Vision and Pattern Recognition, 9397–9406.

Liu, W.Q., S.E. Chen and E. Hauser (2012). “Bridge Clearance Evaluation Based on Terrestrial LiDAR Scan,” ASCE Journal of Performance of Constructed Facilities, 26(4), 469-477.

Liu, W.Q. and S. Chen, (2013) “Reliability Analysis of Bridge Evaluations based on 3D LiDAR Data,” Structural Control and Health Monitoring, 20(12), 1397-1409.

Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., Wang, Y., Sun. Y. (2018) “Dynamic Graph CNN for Learning on Point Clouds.” arXiv:1801.07829.

Ma, L., Li, Y., Li, J., Yu, Y., Marcato Junior, J., Gonçalves, W.N. and Chapman, M.A. (2020) “Capsule-based Networks for Road Marking Extraction and Classification from Mobile LiDAR Point Clouds,” IEEE Transactions on Intelligent Transportation Systems,

Melville, B.W. and S.E. Coleman, (2000) Bridge Scour. Water Resources Publications, Highlands Ranch, CO.

NCHRP (2009), “Monitoring Scour Critical Bridges – A Synthesis of Highway Practice”, Traffic Safety, National Cooperative Highway Research Program (NCHRP), Washington DC.

Park, G., Im, D., Han, D., Yoo, H.J., (2020) “1.15 TOPS/W Energy-Efficient Capsule Network Accelerator for Real-Time 3D Point Cloud Segmentation in Mobile Environment,” IEEE Transactions on Circuits and Systems - II: Express Briefs, 67(9), 1594-1598.

Qi, C.R., Su, H., Mo, K. and Guibas, L.J. (2017) “Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation.” Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, 652–660.

Qi, C.R., Yi, L., Su, H., and Guibas, L.J., (2017) “Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.” Proceedings, Conference Neural Inference Processing Systems (NIPS).

Sabour, S., Frosst, N. and Hinton. G.E. (2017) “Dynamic Routing between Capsules,” Proceedings, Conference Neural Inference Processing Systems (NIPS), 3856-3866.

Saqur, R. and Capsgan, S.V. (2018) “Using Dynamic Routing for Generative Adversarial Networks.” arXiv preprint arXiv:1806.03968.

Shirole A.M. and R.C. Holt, (1991) “Planning for a Comprehensive Bridge Safety Assurance Program.” Transport Research Record, Transport Research Board, 137e42.

Tang, W., Chen, S.E., Diemer, J., Allan, C., Chen, T., Slocum, Z., Shukla, T., Chavan, V.S. and Shanmugam, N.S. (2022) DeepHyd: A Deep Learning-based Artificial Intelligence Approach for the Automated Classification of Hydraulic Structures from LiDAR and Sonar Data, Final Report, North Carolina Department of Transportation, FHWA/NC/2019-03.

Upadhyay, Y. and Schrater, P. (2018) “Generative Adversarial Network Architectures for Image Synthesis Using Capsule Networks,” arXiv preprint arXiv:1806.03796.

Xiang, C., Zhang, L., Tang, Y., Zou, W. and Xu, C. (2018) “MS-CapsNet: A Novel Multi-Scale Capsule Network,” IEEE Signal Processing Letters, 25(12), 1850-1854.

Yang, Y, Feng, C., Shen, Y. and Tian, D. (2018) “Foldingnet: Point Cloud Auto-Encoder via Deep Grid Deformation.” Proceedings, IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Zhao, Y., Birdal, T., Deng, H., Tombari, F., (2019) “3D Point Capsule Networks,” Proceedings, IEEE/CVF Conf. Compu. Vis. Pattern Recognition (CVPR), Long Beach, CA, USA, 1009-1018.

Zhu, K., Chen, Y., Ghamisi, P., Jia, X. and Benediktsson, J.A. (2019) “Deep Convolutional Capsule Network for Hyperspectral Image Spectral and Spectral-Spatial Classification,” Remote Sensing, 11,223.

3 SPATIAL INTERPOLATION OF BRIDGE SCOUR POINT CLOUD DATA USING ORDINARY KRIGING

3.1 Abstract

Scour is a critical condition change for a bridge hydraulic system and terrestrial LiDAR scans have been suggested as a way to quantify the scour conditions. With LiDAR point cloud data, a temporal record of scour can be established. However, there are limitations to LiDAR scans, for example laser light does not bend and can be obstructed by objects along the light path resulting in missing geometric information behind the obstacles thereby creating a void in the point cloud data. To 'fill in' the missing data, spatial interpolation of 3D LiDAR point cloud data using Ordinary Kriging (OK) method is suggested and actual field data from scanning a scoured bridge pier are presented to demonstrate the application. Kriging is a geostatistical interpolation technique and OK assumes that the spatial variation of the phenomenon or object being considered is random and intrinsically stationary with a constant mean. Here, the complete scour envelope is reconstructed using kriging and is shown to have excellent results.

Keywords: Bridge Scour, LiDAR Scan, Data Void, Kriging

3.2 Introduction

Scour is the primary cause of bridge failures in the United States and is responsible for almost 60% of bridge failures (Melville and Coleman 2000). During the last 30 years, 600 bridges have failed due to scour problems (Shirole and Holt 1991). Scour, caused by swiftly moving water, can remove sand and soil, create holes surrounding a bridge component and compromise the integrity of a structure (FHWA 2012). The danger of bridge scour failures lies in the fact that they can occur without prior warning. Thus, there is a need for effective monitoring techniques for the assessment of scour problems surrounding a bridge structure.

Figure 3-1 shows the mechanisms of scouring surrounding a bridge pier. The hydrodynamic characteristics of the river flow and the integrity of the surrounding geomaterial of a bridge pier dictate the scour process and define the extent of the scouring problem. The scour shown in Figure 3-1 would typically be considered as ‘localized’ scour which occurs around individual bridge piers and abutments and is caused by an acceleration of flow and resulting vortices induced by obstructions to the flow. Once scouring removes the soil support around the bridge pier, the soil bearing capacity can be reduced significantly and may lead to excessive structural settlement resulting in load redistribution amongst structural members ultimately leading to the failure of a bridge structure.

Bathymetric surveys of bridges, stream channels, and wetlands are pivotal to spatially explicit modeling of flood, drainage, and scours (Prendergast et al. 2014). Remote sensing techniques are especially useful for bathymetric surveys. In particular, terrestrial LiDAR has been gaining popularity as a condition monitoring technique for highway bridge systems; based on the detection of returned laser beams, the technique has been used for bridge monitoring needs such as detection of condition changes (Liu et al. 2012, 2013). For scour detection, LiDAR scanning of bridge hydraulic structures has been proposed as a scour monitoring technique, where high-resolution point cloud data of a bridge hydraulic structure can be used to quantify material mass losses (Suro et al. 2020). Rapid and repeated laser scans can generate periodic quantification of scours and help define the process of erosion and determine the rate of removal of streambed or bank material surrounding the bridge foundation. For large riverine scans, Lewis et al. (2020) suggested the combined LiDAR-UAS (Unmanned Aerial Systems) technique for bank erosion studies.

However, there are limitations to the LiDAR scans: Since traditional LiDAR uses infrared light, it gets absorbed by water resulting in the inability to scan underwater surfaces. For the same reason, they cannot be used in rainy or foggy weather. Light does not bend, so it can be obstructed by an object along the light path, resulting in interference (Rodriguez 2012). As a result, obstacles make the surfaces behind the obstacle ‘invisible’ to the LiDAR and thereby create a void in the point cloud data. A typical solution is to scan the surface from multiple locations and stitch the point clouds together to generate the full image. However, due to the complexity and size of the scoured area, sometimes, even multiple scans will not be able to fully describe the scoured void. In experiments utilizing circular holes with different surface finishes, Bian et al. (2017) demonstrated that the backscattering of light within circular holes can result in complicated light scattering effects on point cloud data.

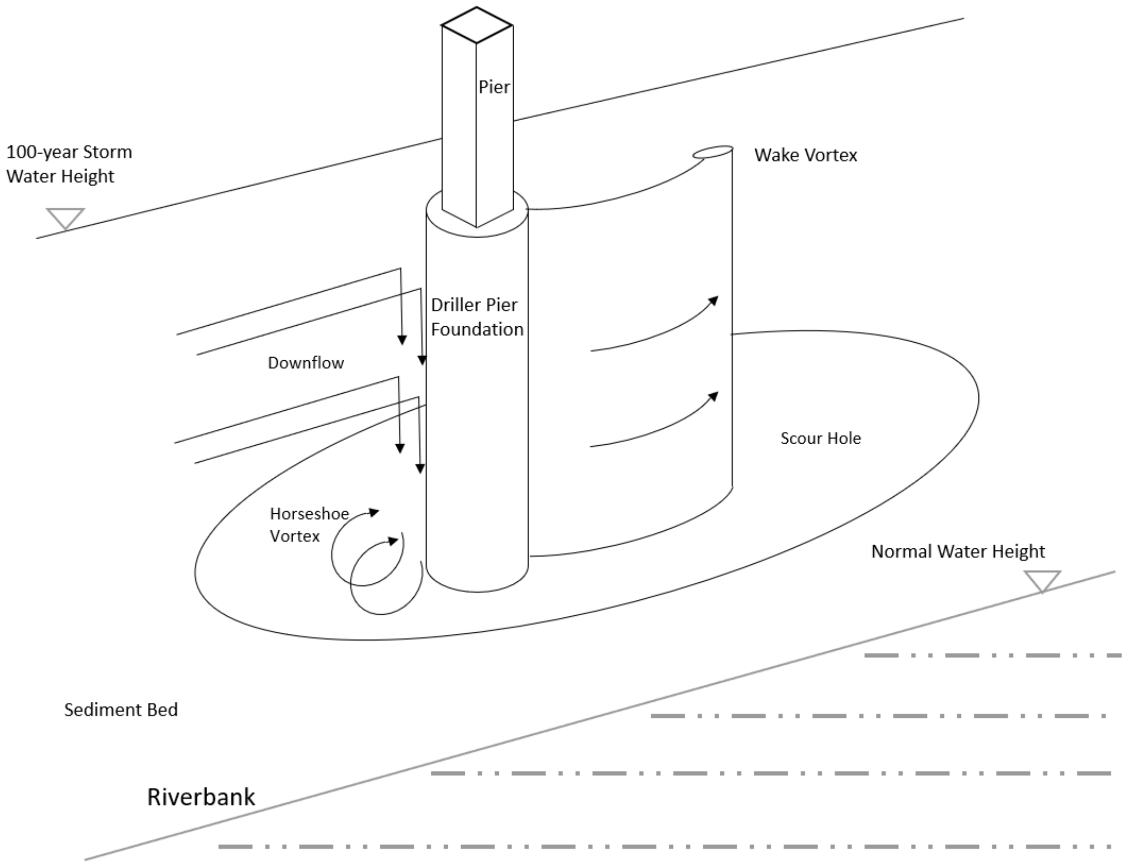


Figure 3-1 Mechanism of Scour at a Circular Pier

To address the issue of voided data, this paper proposes the use of spatial interpolation, which is the process of estimating values of locations that are not surveyed, using data from the network of known points, to help fill-in the voided point cloud data. There are several spatial interpolation techniques including both deterministic and statistical methods (Isaaks et al. 1989; Cressie 1993). In particular, Kriging is useful for populating areas of voided data (Meng et al. 2013). Kriging is a spatial prediction technique based on the assumption of random processes and the use of spatial covariance estimations. Kriging methods may be classified into Ordinary Kriging (OK), Simple Kriging (SK), Universal Kriging (UK), Residual Kriging (RK), and Co-Kriging (Knotters et al.

1995; Montero et al. 2015). In the case of complex and under-sampled problems, Co-Kriging using auxiliary variables has been used to populate the area of voided data (Knotters et al. 1995).

In this study, we explore the spatial interpolation of a 3D LiDAR point cloud of a bridge scour using OK method and test the interpolated results using a separate point cloud scan of the missing area (with additional scans). LiDAR scan data from the Phillips Road Bridge at UNC Charlotte, Charlotte, North Carolina, USA, were used in the current study (Chavan et al. 2022). LiDAR scans were performed around the piers of the Phillips Road bridge which experienced soil mass losses (scour) and the point cloud data were generated for the current study. The complete scour envelope is reconstructed using the OK method and is shown to have excellent results in filling the voided point cloud data.

3.3 Spatial Interpolation and Kriging

The inherent assumptions for spatial interpolation approaches include the autocorrelation of spatial data and that the values are continuous over space (Oliver and Webster 2015). There are several different types of spatial interpolation methodologies including the Inverse Distance Weighted (IDW), different variations of Kriging, and local polynomial interpolations, to name a few (Montero et al. 2015). In particular, Kriging is a geostatistical interpolation technique used to predict the unknown value of a function at a given point by computing a weighted average of known values of the function in the neighborhood of the point. In this study, the OK method, which relies on the spatial autocorrelation of the data to determine the weighted values that can be used to predict the unsampled point and suggests an unknown mean value (m), is used:

$$\mathrm{Z(s) = m + e(s)} \tag{1}$$

where Z(s) = predicted value and e(s) = a random quantity with a mean value of zero and covariance c(h):

$$c(h) = E[e(s)-e(s+h)] \qquad (2)$$

and h is the separation between samples, s, and E is the expectation.

Under the assumption of intrinsic stationarity, the expected differences are zero.

$$E[Z(s) - Z(s+h)] = 0 \qquad (3)$$

Covariance can be replaced by half of the variance of differences (semivariance).

$$\gamma(h) = \tfrac{1}{2}\,\mathrm{var}[Z(s) - Z(s+h)] = \tfrac{1}{2}\,E[\{Z(s) - Z(s+h)\}^2] \qquad (4)$$

The semivariance depends only on h, and the function γ can be used to construct the variogram. The variogram function is a measure of the spatial autocorrelation. When the distance increases away from the sample points, the autocorrelation (similarity) between the sample points tends to decrease and, when their variances begin to flatten out, the sample values are no longer related to one another. The value at which the sample values are no longer associated (completely spatially independent) is indicated as a 'sill'.

Figure 3-2 shows the general idea of Kriging where one-dimensional randomly distributed points are used (Figure 3-2a). The Kriging estimator usually relies on a weighted function that may be a function of the covariance of the sampled points (measured values). As shown in Figure 3-2b, two clusters of missing data can be filled by Kriging estimator (Figure 3-2c and Figure 3-2d). The resulting data may have different M and ê(s) values:

$$Z(s) = M + \hat{e}(s) \qquad (5)$$

The OK predictor is a linear combination of the data values. It is unbiased because it attempts to keep the mean residual to zero and tries to minimize the residual variance. OK is also statistically called the 'best linear unbiased estimator' (Schabenberger and Gotway 2005).

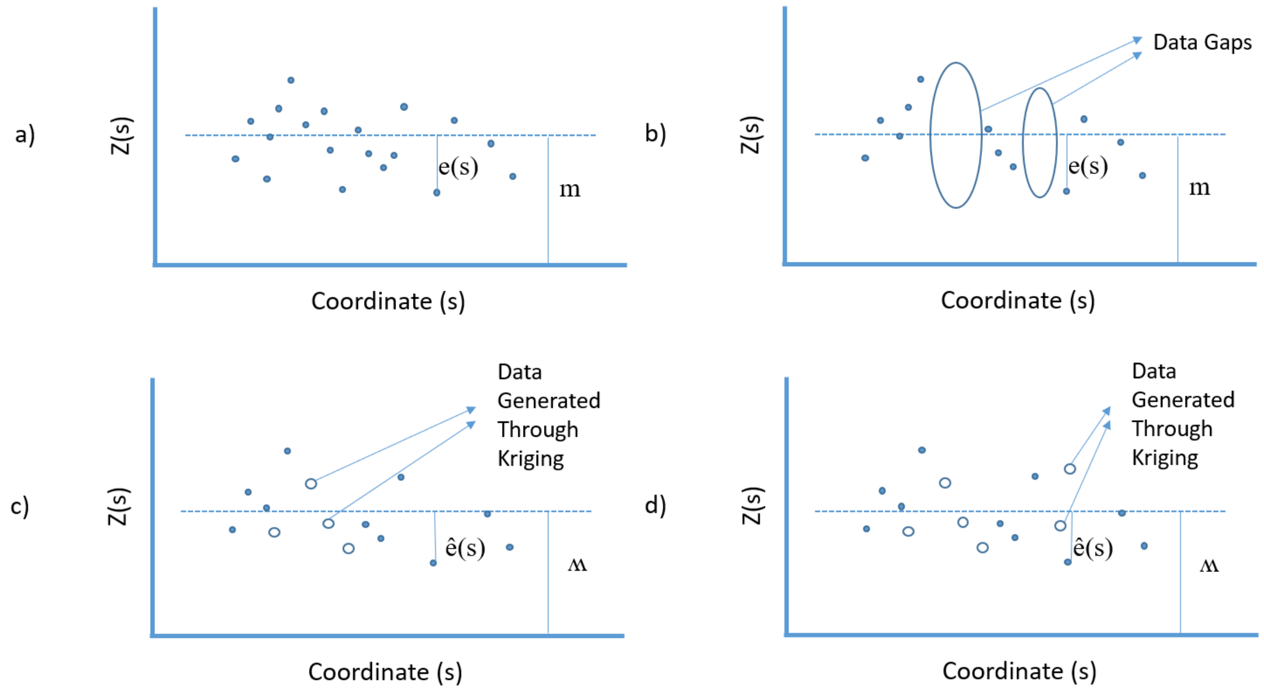


Figure 3-2 Spatial Variation in One Dimension (s)

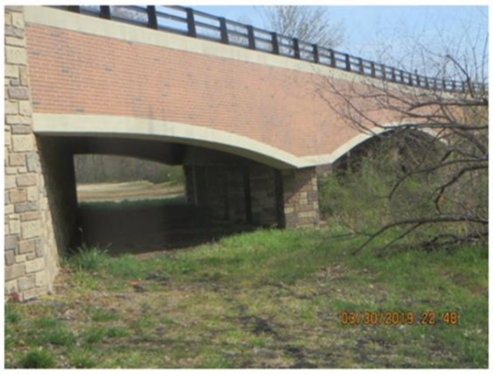
Identifying a scoured pier is non-trivial as the scour depression around the pier may be obscured by vegetation, blocking the line-of-sight of the LiDAR. Due to the size of a bridge pier and the geometric shape of a scour, a full scan of the scour cannot be made from a single scan position. Hence, to completely capture a scoured area, LiDAR must be shot from multiple positions (while at the same time keeping track of the scanning angles and the heights of the laser head above the ground surface). The multiple scans can then be stitched together to generate a more complete point cloud dataset. However, this may not always be feasible at a bridge site. For example, obstacles such as constricted site space can prevent a full-circle scan of the bridge pier.

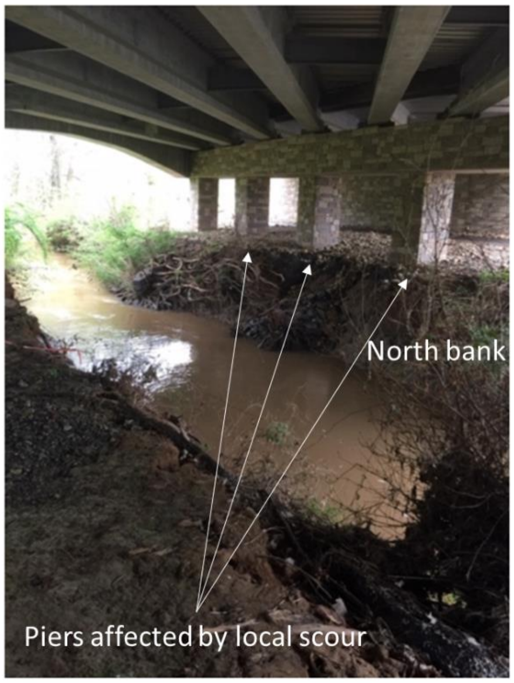
Due to the scanning mechanism of a laser head, there is a 60° cone underneath the laser that cannot be scanned. As a result, it is usually not feasible to place the laser directly above a scoured area and additional space is required between the laser position and the scour. Furthermore, a stitched point cloud takes more time to process.

3.4 The Phillips Road Bridge Case Study

The scoured pier considered in this study belongs to the six-year-old Phillips Road Bridge over Toby Creek (35°18'28.2"N 80°44'16.6"W) located at the University of North Carolina at Charlotte (Figure 3-3). Phillips Road Bridge is a three-span continuous prestressed concrete girder bridge with span lengths of 14, 25, and 14 m, respectively. The bridge has a clear roadway width of 9.75 m and supports two traffic lanes of 4.88 m each. The overall width of the bridge deck is 15.54 m. The cast-in-situ concrete slab has a uniform thickness of 20.96 cm and is supported by seven prestressed concrete girders at each span.

The bridge has semi-integral abutments with expansion joints at the abutments and end bents. At the intermediate bents, the bridge girders are resting on fixed elastomeric bearings above the piers. Figure 3-3 shows different views of the bridge. Bents 1 and 2 are supported on drilled pier foundations. The end bent abutments include cast-in-place stem walls with wings turning back parallel to the roadway. The End Bent 1 wall has exposed heights up to approximately 5.18 m and End Bent 2 wall has exposed heights up to approximately 5.48 m. A strip footing connecting the piles are a minimum of 0.61 m below the existing ground and founded on a pile supported strip footing. The scours surrounding the piers on the north bank were observed on 4 of the 6 piers and were scanned on June 9, 2020.





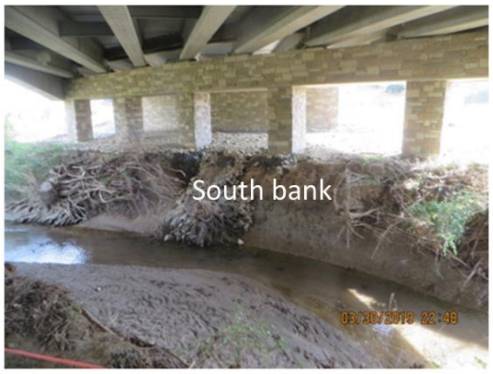


Figure 3-3 Snapshots of Phillips Road Bridge (on 03/30/2019)

3.5 Scour Scan and OK Analysis

Figure 3-4 shows the studied pier and a full LiDAR scan of the Phillips Road Bridge. The scanning was performed using a FARO Focus S 350 LiDAR. The FARO LiDAR uses a mono-dyne laser with a wavelength of 1,550 nm. Multiple scans were conducted on the bridge over a three-year span (2018-2020). A point cloud processing program (CloudCompare, v. 2.6.1, 2015) was used to open the proprietary FARO file format (.fls) and to segment out all the points excluding the scoured points. The point cloud of the scoured area contains the x, y, and z coordinates. Figure 3-5a) shows the point cloud of the scour surrounding the selected pier of the Phillips Road Bridge. Figure 3-5a) also indicates a voided area (there are multiple voided areas).

To quantify the scour, a reference plane is first defined and projections of points to the reference plane are then determined. Figure 3-5b shows the defined reference plane and the projections. The projections are the perpendicular distances from the reference plane to the data points in the point cloud. This is used to calculate the perpendicular distances of points to the plane and remove recurring points. The recurring points must be removed so that the data have only one depression (z value) at a particular point on the x-y plane. These data are then interpolated to fill the missing data between the points. To assess the interpolation results, a stitched point cloud, derived from a combined point cloud from multiple scans of the selected scour area, are compared with the interpolation results.

The reference plane was used with a criterion that the number of data points should be minimized allowing only a single normal vector passing through the reference plane. Where multiple points passing through the same plane (recurring projections from the same normal vector to the reference plane) occurred, the additional data points were removed. Three points were selected, and their coordinates were used to define the reference plane. One of the points chosen comprised the origin and the entire dataset was translated and rotated to simplify the analysis.

If s_1, s_2, s_3 are the coordinates of the three selected points, and s_1 is chosen to be the origin, the equation of the plane is:

$$(s-s_1).n = 0 \tag{6}$$

where s is an arbitrary vector and n is the normal vector to the plane defined by

$$n =(s_2-s_1) \text{ x } (s_3-s_1)$$

This normal vector is rotated such that it is oriented towards the z axis and makes the plane parallel to the xy plane. Thus, the z coordinates of the points will be the distance of that point from the reference plane. This can be achieved by rotating the normal vector by an angle of r_{angle} about r_{axis}, which are given by:

$$r_{angle} = \cos^{-1}\left(\frac{n}{||n||}.[0\ 0-1]\right) \tag{7}$$

$$r_{axis} = \frac{n}{||n||} \text{ x } [0\ 0\ 1] \tag{8}$$

where ||n|| is the norm of n and [0 0 1] is the z axis vector.

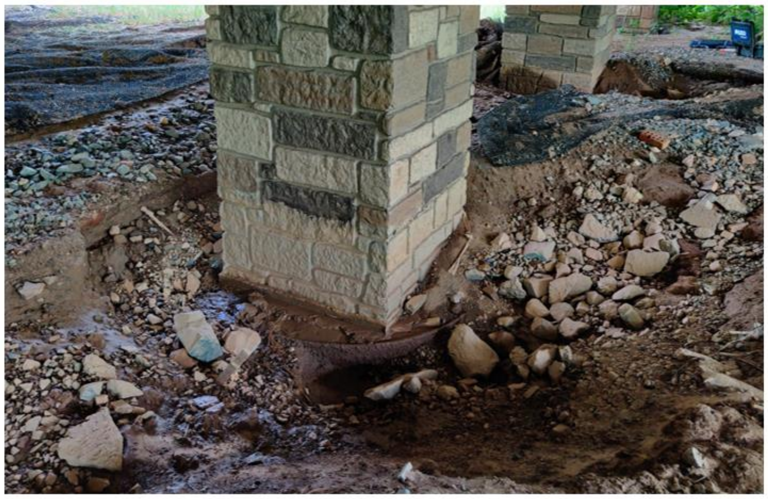
Finally, the transformation matrix T is given as:

$$T = \begin{bmatrix} t*u1*u1+C & t*u1*u2-S*u3 & t*u1*u3+S*u2 \\ t*u1*u2+S*u3 & t*u2*u2+C & t*u2*u3-S*u1 \\ t*u1*u3-S*u2 & t*u2*u3+S*u1 & t*u3*u3+C \end{bmatrix} \tag{9}$$

where $C = \cos(r_{angle})$

$S=\sin(r_{angle})$

$t=1-C$



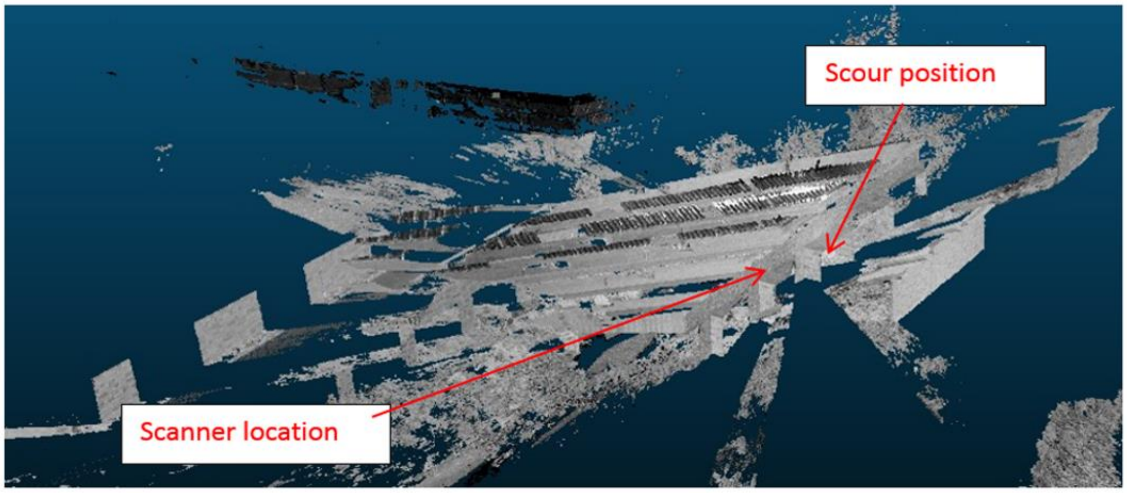


Figure 3-4 The Scoured Pier of the Phillips Road Bridge: a) Scoured Pier on the North bank and b) Bridge Laser Scan

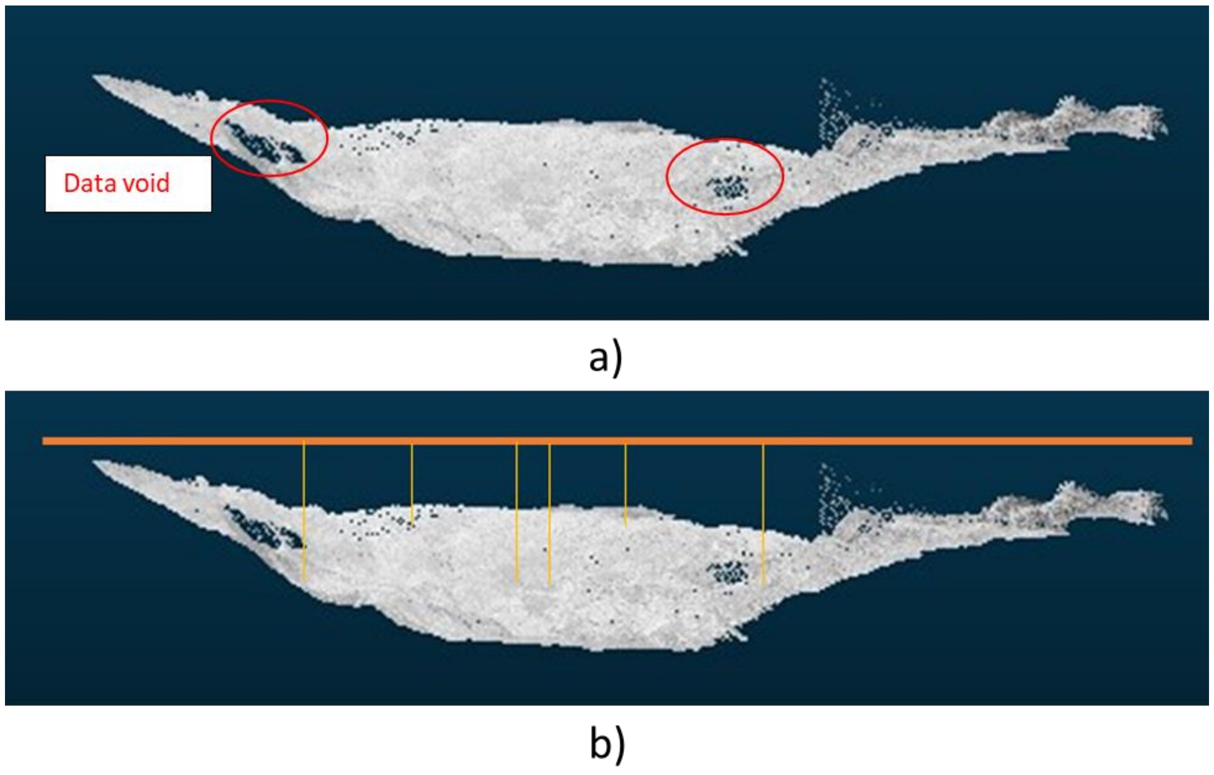


Figure 3-5 Sample Scour Point Cloud with a) Data voids marked, b) Sample reference plane with plane perpendiculars going towards the scour data

and [u1 u2 u3] is the unit vector for the axis of rotation, r_{axis}. Transforming the data by this matrix, the data coordinate will become (x,y,z) – where (x,y) is the coordinate of the perpendicular projection of a point on the reference plane and z is the distance of the point from the reference plane. If a point was found to have repeated projections, the one that was furthest away from the reference plane was removed. The data now have the position (x,y) and a single z (scour depth) value at each position. The algorithm was programmed in MATLAB and ArcGIS Pro software.

A semivariogram plot was constructed for the spatial data and to evaluate multiple fits for the semivariogram, a Stable curve fit with a parameter of 1.871 was found to be the most suitable. A Stable parameter of 1 corresponds to the Exponential model and 2 corresponds to the Gaussian model. Using the fit, the range of the data, the sill and the nugget effect were calculated. The term 'range' refers to the distance at which the model falls off and there is no more spatial relation in the data; this is a representation of the data correlation. 'Sill' is the value at which the model is out of range and is an indication of the data variance. Finally, 'nugget' is the value at which the semivariogram intercepts the y axis (distance = 0). Figure 3-6 shows the semi-variance curve for the Kriging model, where OK was performed to spatially interpolate the data using these parameters.

For this study, a stitched point cloud was used to test the interpolated results. The stitched point cloud was then segmented where the data void existed to define the scour boundary for the Kriging analysis.

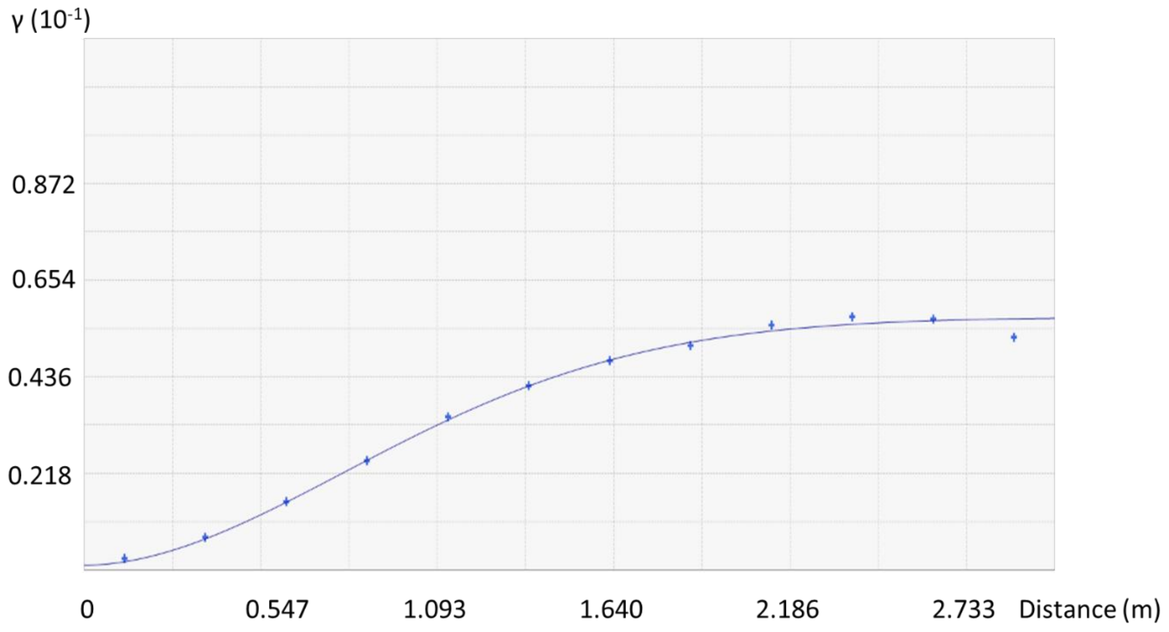


Figure 3-6 Semi-Variance Graph for the Selected Point Cloud

3.6 Results and Discussions

The semivariogram range and sill were calculated to be 2.16 m and 0.05, respectively. The voided scour hole scan has a total of 1,000 data points (segmented out from the original stitched point cloud with a void). The average interpolation error calculated using the x and y coordinates was approximately 0.02 m. This was calculated by comparing the interpolated point cloud with the stitched point cloud. The transformation matrix, T, for the sample problem was calculated to be

$$T = \begin{bmatrix} 0.9996 & 0.0292 & 0.0007 \\ 0.0292 & -0.9985 & -0.0466 \\ -0.0007 & 0.0466 & -0.9989 \end{bmatrix}$$

Moreover, the cos(r_{angle}) value was equal to 0.9985 (a value = 1 would indicate that the reference plane and the original coordinate plane perfectly overlay each other).

Figure 3-7 below shows the interpolation data of the final scour hole. The numbers in the legend are the depth range of scour at various locations. As the r_{axis} was defined by multiplying [0 0 1] (unit vector in the z direction) in Equation 8, the depths of the scour were presented in negative values. The maximum depth region lies near the center of the data region, and it decreases as the processing moves outwards, which matches with the actual field data. The cut-out section in the scour boundary is due to a portion of the square bridge pier and the rest of the boundaries are selected to capture only the scoured area. A comparison between the original point cloud and the interpolated cloud is shown in Figure 3-8. The Kriging result is shown to be significantly denser than the original point cloud.

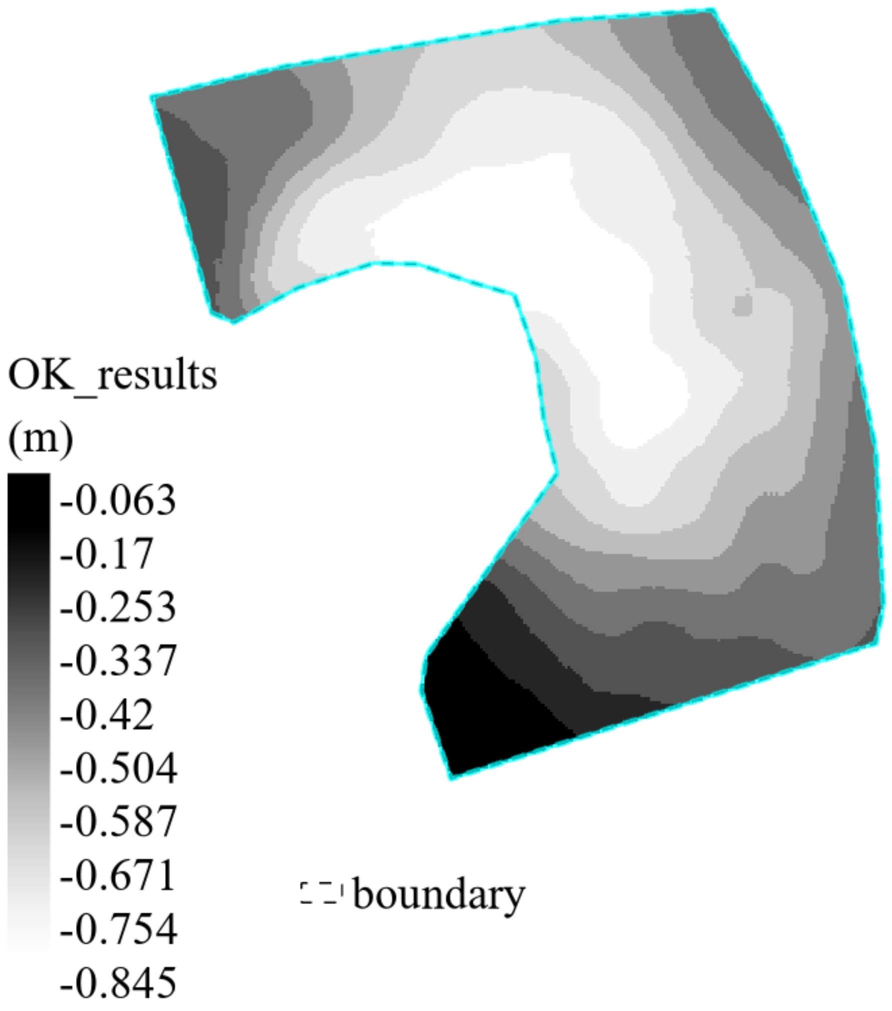


Figure 3-7 Ordinary Kriging Interpolation Results – Legend Shows the Scour Depth Range (in Meters)

It is important to recognize that segmenting out the point cloud for scour points is time-consuming and dependent on the size of the original point cloud. Also, the point density in a single scan point cloud decreases with distance from the scanner, i.e., points are closer to each other when the location of the scanned surface is close to the LiDAR device. It is possible to sample down the point cloud by reducing the number of spatial data points in the point cloud to reduce computation time. However, this should only be performed when the whole scoured area and voided volume should not be compromised. For example, when the scour location is not close to the scanner.

Finally, the current scour case studied did not have any data points removed as there were no redundancies in the original point cloud. The post-Kriging point cloud shown in Figure 3-8 has a spatial resolution of 2 mm resulting in a 3D point cloud of 1.86 million data points. As shown in Figure 3-8, the post-Kriging data density is extremely high as a result of the analysis trying to match the smallest distance between the points. Since the original point cloud has variable data spacing, the fine scale post-Kriging eliminated any likely data voids.

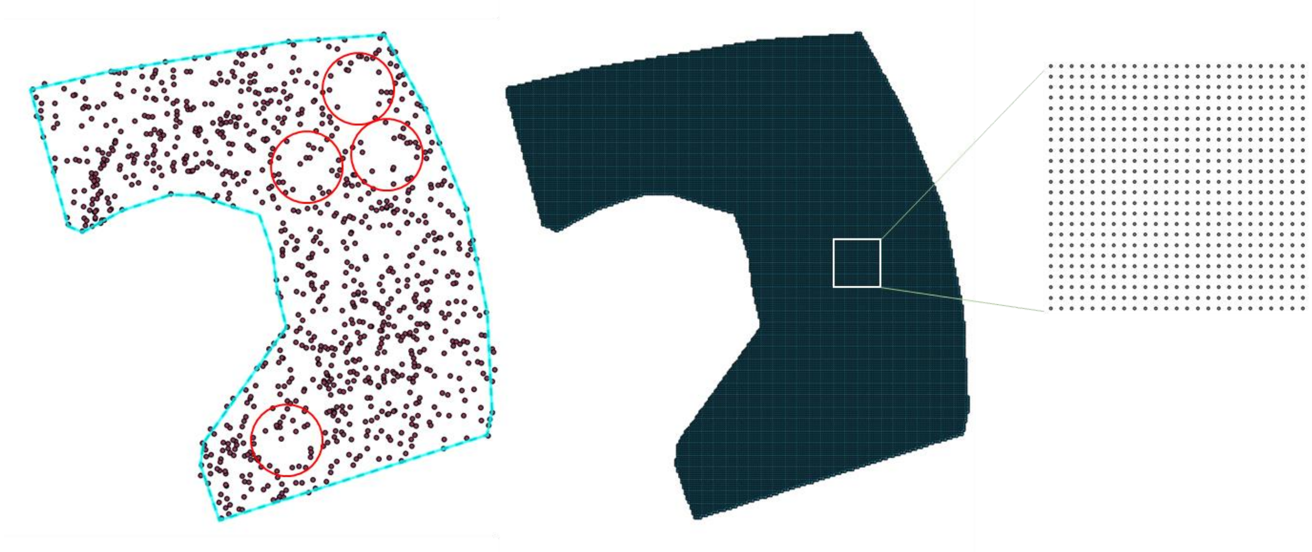


Figure 3-8 Original Point Cloud (1,000 Points, Left) with voids marked in circles and Point Cloud (1.86 million Points, Right) after Kriging Interpolation

The OK method utilized in our analysis generated an extremely dense point cloud, which can make further analysis time consuming. Future studies should investigate approaches to reduce the spatial density of the data (i.e., reduce the number of data points).

3.7 Conclusions

In this study, a LiDAR scan of a scoured hole from a pier of the Phillips Road Bridge was identified with data voids generated to investigate the possible obstruction of the LiDAR scan from vegetation surrounding the pier or physical site constraints which might limit the viewscape of a scan. The point cloud of the scour was segmented using CloudCompare and an OK process was performed on the data to populate the data voids that were generated. A reference plane was manually selected by choosing three points and the dataset was transformed in its orientation to Cartesian (xyz) coordinates with z as the scour depth. Furthermore, the distances of the points from the selected reference plane were determined. The accuracy of the kriging results is reported in a semivariogram along with the sill, range, and nugget effect values. The results show that the OK procedure generated data points that can accurately fill the spatial voids of the scanned scour hole.

This study represents a first attempt in using OK to populate voided scour LiDAR scans, which can be developed into an effective scour monitoring technique. Currently, bridge inspections typically only report scour depth. LiDAR scans with the suggested Kriging data interpolation technique can be used to quantify scour parameters such as the scoured surface area and the scoured volume. The data can also be used to define a reference surface for the scour point data, allowing bridge engineers to quantify the actual volume and area of the mass/volume losses during the hydrodynamic processes of scouring over time.

3.8 References

Bian, H., Chen, S.E., and Liu, W. (2017) "Error Sources in Processing LiDAR-Based Bridge Inspection," ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 455-459.

Chavan, V.S., Chen, S.E., Shanmugam, N.S., Tang, W., Diemer, J., Allan, C., Braxtan, N., Shukla, T., and Chen, T. (2022) "Analysis of Local and Combined (Global) Scours on Bridge Piers-on-Bank," CivilEng, 3, doi.org/10.3390/civileng301001.

CloudCompare (2015) User's Manuel, version 2.6.1, http://cloudcompare.org/, last accessed: 9/1/2020.

Cressie, N.A.C. (1993) Statistics for Spatial Data, John Wiley and Sons, New York, NY.

FHWA (2012) Evaluating Scour at Bridges, Hydraulic Engineering Circular (HEC) No. 18, 5th edition, Federal Highway Administration, US Department of Transportation, Washington, D.C.

FHWA (2012) Stream Stability at Highway Structures, Fourth Edition, FHWA-HIF-12-004, Hydraulic Engineering Circular (HEC) No. 20, Federal Highway Administration, US Department of Transportation, Washington, D.C.

Isaaks, E.H., and Srivastava, R.M. (1989) An Introduction to Applied Geostatistics, Oxford University Press, New York, NY.

Knotters, M., Brus, D.J., and Oude Voshaar, J.H. (1995) "A Comparison of Kriging, Co-Kriging and Kriging Combined with Regression for Spatial Interpolation of Horizon Depth with Censored Observations," Geoderma, 67, 227-246.

Lewis, Q.W., Edmonds, D.A., and Yanites, B.J. (2020) "Integrated UAS and LIDAR Reveals the Importance of Land Cover and Flood Magnitude on the Formation of Incipient Chute Holes and Chute Cutoff Development," Earth Surface Processes and Landforms, 45(6), 1441-1455.

Liu, W.Q., and Chen, S.E. (2013) “Reliability Analysis of Bridge Evaluations based on 3D LiDAR Data,” Structural Control and Health Monitoring, 20(12), 1397-1409.

Liu, W.Q., Chen, S.E., and Hauser, E. (2012). “Bridge Clearance Evaluation Based on Terrestrial LiDAR Scan,” ASCE Journal of Performance of Constructed Facilities, 26(4), 469-477.

Melville B.W., and Coleman, S.E. (2000) Bridge Scour. Water Resources Publications, Highlands Ranch, CO.

Meng, Q., Liu, Z., and Borders, B.E. (2013) “Assessment of Regression Kriging for Spatial Interpolation – Comparisons of Seven GIS Interpolation Methods,” Cartography and Geographic Information Science, 40(1), 28-39.

Montero, J.M., Fernandez-Avilés, G., and Mateu, J. (2015) Spatial and Spatio-Temporal Geostatistical Modeling and Kriging, John Wiley and Sons, West Sussex, UK.

Oliver, M.A., and Webster, R. (2015) Basic Steps in Geostatistics: The Variogram and Kriging, Springer, London, UK.

Prendergast, L.J., and Gavin, K. (2014) “A Review of Bridge Scour Monitoring Techniques,” Journal of Rock Mechanics and Geotechnical Engineering, 6(2), 138-149.

Rodriguez, A.M. (2012) Laser Scanner Technology, Intech Pub., Rijeka, Croatia.

Schabenberger, O., and Gotway, C.A. (2005) Statistical Methods for Spatial Data Analysis. Chapman & Hall/CRC.

Suro, T.P., Huizinga, R.J., Fosness, R.L., and Dudunake, T.J. (2020) “Assessment of Bridge Scour Countermeasures at Selected Bridges in the United States, 2014-18,” Scientific Investigation Report 2019-5080, U.S. Geological Survey, Reston, VA.

4 SPATIAL DENSITY ANALYSIS OF BRIDGE SCOUR POINT CLOUD USING ORDINARY KRIGING

4.1 Abstract

In this study, we propose using terrestrial LiDAR to capture bridge pier scour hole point clouds and use Ordinary Kriging (OK) as a spatial interpolation method to normalize the point cloud density and establish the scour hole signatures (volume and area). Scour hole signatures are important to establish the extent of hydraulic erosion surrounding a bridge pier. Failure of managing scour surrounding bridge piers can lead to bridge failures. However, LiDAR point cloud data for bridge scours often display uneven spatial distributions resulting in data voids that can be critical to the quantification of scour hole area and volume. To 'fill in' the missing data, OK method has been used to spatially interpolate 3D LiDAR point cloud data. In this study, different spatial data densities have been generated with a goal to determine what is the 'optimal' data resolution that can help minimize later computational efforts. Actual field data from scanning a scoured bridge pier are used in the sensitivity analysis. Two scans from different time periods were also compared.

Keywords: Bridge Scour, LiDAR Scan, Data Void, Kriging

4.2 Introduction

Scour is one of the primary causes of bridge failures worldwide and is responsible for almost 60% of bridge failures in the United States (Melville and Coleman 2000). Scour can be defined as the erosion of soil and sand surrounding a bridge component caused by swiftly moving water. This compromises the integrity of a bridge structure (Arneson et al. 2012; Lagasse et al. 2012). Scour failures can occur without prior warning and thus is very dangerous. So, there is a need for effective monitoring techniques for assessing scour potential surrounding bridge structures.

Remote sensing techniques have been identified as a transformative technology for both terrestrial and bathymetric surveys (Lewis et al. 2020). Terrestrial LiDAR, in particular, has been found to be useful as a condition assessment technique for highway bridges including bridge displacement and as-built condition assessments (Dai et al. 2014; Fuchs et al. 2004; Watson et al. 2013). More recently, LiDAR has been used for monitoring bridge construction processes, which generates a temporal record of the building process and is useful for geometric confirmation of design drawings (Lin et al. 2021).

Scour is a unique bridge problem interfacing three disciplines: structural engineering, geotechnical engineering, and hydraulic engineering. Terrestrial lasers have been used to capture the physical conditions of the bridge hydraulic structures and to generate high-resolution point clouds of the bridge to quantify surficial damage (Chavan et al. 2021; Suro et al. 2020). The most significant benefit of scour detection using LiDAR scans is the potential of quantifying mass losses, which can reflect on both the hydrodynamic history and bridge stability. However, there are obvious limitations to the LiDAR scans including the obstruction of laser beam pathways by unintended objects resulting in imaging interferences. As a result, it is hard to construct a complete image of the targeted area without conducting laser scans from multiple locations.

Due to the roughness of the ground surface, each scanned area can be impeded by voided areas similar to the shadows of pebbles or rocks within a scour. This presents a different issue to the LiDAR scan applications to bridge scour monitoring and can remain even when scans from multiple locations are performed. The stitched point cloud may cover the full scour surfaces, but some data voids could remain within the stitched point cloud due to subtle covered areas created by, for example, pebbles. Furthermore, the surface textures can also have effects on the laser energy returned to the transceiver. By experimenting on circular holes with different surface finishes, Bian et al. (Bian et al. 2017) demonstrated that the back-scattering of light within the circular holes can result in a complicated light energy scattering effect on point cloud data.

To address the above-mentioned voided point cloud data issue, this paper proposes the use of spatial interpolation using Ordinary Kriging (OK) technique. To help fill-in the voided point cloud data, Kriging, which is a process to estimate values of locations by interpolation, can help generate additional data points that are not surveyed by using the data from a network of known points (Cressie 1993; Isaaks and Srivastava 1989; Krige 1951; Meng et al. 2013). In this study, the most common OK technique is used for spatial interpolation of 3D LiDAR point cloud data from the scanning of an existing bridge scour.

The case study bridge site is located in Charlotte, North Carolina (Figure 4-1). Figure 4-2 shows a scanned image of the scour near the bridge (Phillips Road Bridge) at one of the piers: Figure 4-2a) shows the actual pier scour and Figure 4-2b) shows the point cloud data generated from a laser scan that is used in this study.

OK technique is capable of generating different spatial densities from the same scanned point cloud dataset. To determine the sensitivity of the OK technique to the filling of original data voids and the associated data density, the same scanned point cloud is evaluated at different spatial densities. Since a high spatial density corresponds to long computational time for post-processing of data (such as the calculation of the scour hole soil mass loss rate), there is a significant implication to the computational budget. Hence, the objective of this study is to determine the ‘optimal’ spatial density to minimize subsequent computation time associated with a massive number of data points. At the same time, we also attempt to understand the effect of normalized data spacing on the aforementioned data voids.

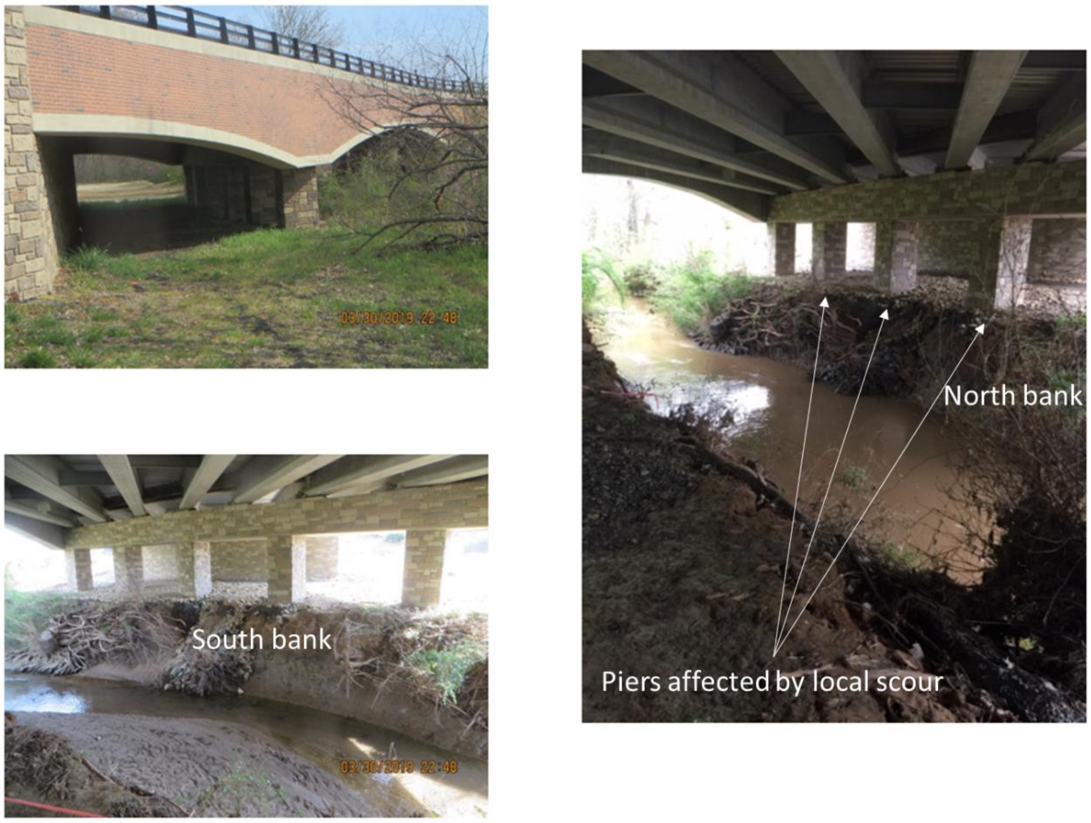


Figure 4-1 Study bridge: Phillips Road Bridge in Charlotte, NC (Photos were taken on 03/30/2019; Photo credit: Shen-En Chen)

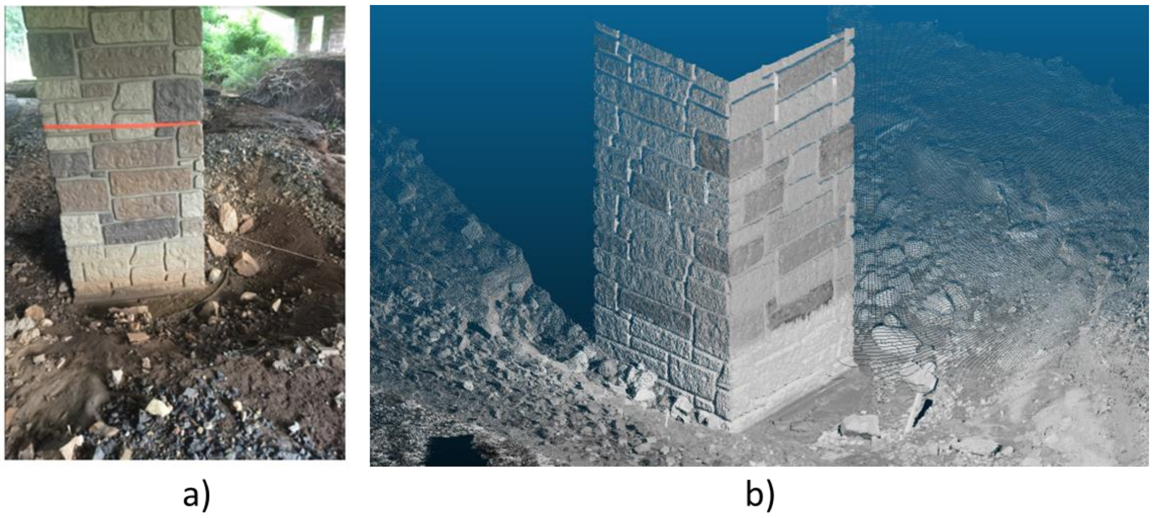


Figure 4-2 Scoured pier of the study bridge: a) scour hole with pebbles and b) LiDAR scan of the scour and pier (Photo credit: Shen-En Chen).

4.3 Spatial Interpolation of Point Cloud Using Ordinary Kriging

There are different types of spatial interpolation techniques including the Inverse Distance Weighted (IDW), different variations of Kriging, and local polynomial interpolations (Montero et al. 2015; Oliver and Webster 2015). Of these, the Ordinary Kriging (OK) is probably one of the most used geostatistical interpolation techniques. In this study, OK is used to predict the unknown value of a variable at a given point by computing a weighted average of known values of the variable at the neighborhood of the point:

$$Z(s) = m + e(s) \tag{1}$$

where *Z(s)* is a predicted value, m is the mean value and *e(s)* is a random quantity with a mean value of zero and an associated covariance c(h):

$$c(h) = E[e(s)e(s+h)] \tag{2}$$

and h is the separation between samples, s, and E is the expectation.

Under the assumption of intrinsic stationarity, the expected differences are zero:

$$E[Z(s) - Z(s+h)] = 0 \quad (3)$$

And the covariance can be replaced by half of the variance of differences (semivariance):

$$\gamma(h) = ½ \operatorname{var}[Z(s) - Z(s+h)] = ½ E[\{Z(s) - Z(s+h)\}^2] \quad (4)$$

The semivariance depends only on h, and the function γ can be used to construct the variogram. Some Kriging terms need to be explained: The term 'range' refers to the distance at which the model falls off and where there is no more spatial relation in the data and is a representation of the data autocorrelation. 'Sill' is the value at which the model is out of range and is an indication of the data variance. Finally, the 'nugget effect' is the value at which the semivariogram intercepts the y axis (distance = 0). The variogram function is a measure of the spatial autocorrelation. As the distance increases away from the sample points, there is no longer a relationship between the sample points and their variances begin to flatten out and the sample values are no longer related to one another

The missing data to be filled by the Kriging estimator may have different M and ê(s) values:

$$Z(s) = \mathcal{M} + \hat{e}(s) \quad (5)$$

Figure 4-3a) and Figure *4-3*b) show pre-Kriging results with Z(s) and M and e(S), whereas Figure 4-3c) and d) show post-Kriging results with Z(s) and M and ê(s).

The OK predictor is a linear combination of the data values. It is unbiased because it attempts to keep the mean residual to zero and tries to minimize the residual variance. OK is also statistically called the 'best linear unbiased estimator' (Schabenberger and Gotway 2017).

Kriging has been used for laser point cloud processing and for different data improvement applications. For example, Hui et al. (Hui et al. 2016) suggested a multi-level kriging interpolation for filtering airborne LiDAR point clouds with the challenges of managing ground versus non-ground spatial points. Large spatial region interpolations are typically complicated by land covers such as vegetation and man-made structures. Scour detection is more akin to the construction of microtopography such as the establishment of wetland boundaries (Stovall et al. 2019) and vegetated sites(Nouwakpo et al. 2016; Zhang et al. 2021). In this case, Kriging can be useful in filling in the voided areas. Da Costa et al. (Da Costa et al. 2018) compared different interpolation techniques including Kriging, spline, and machine learning, for spatial data points and noted that Kriging and spline models are similar in some cases.

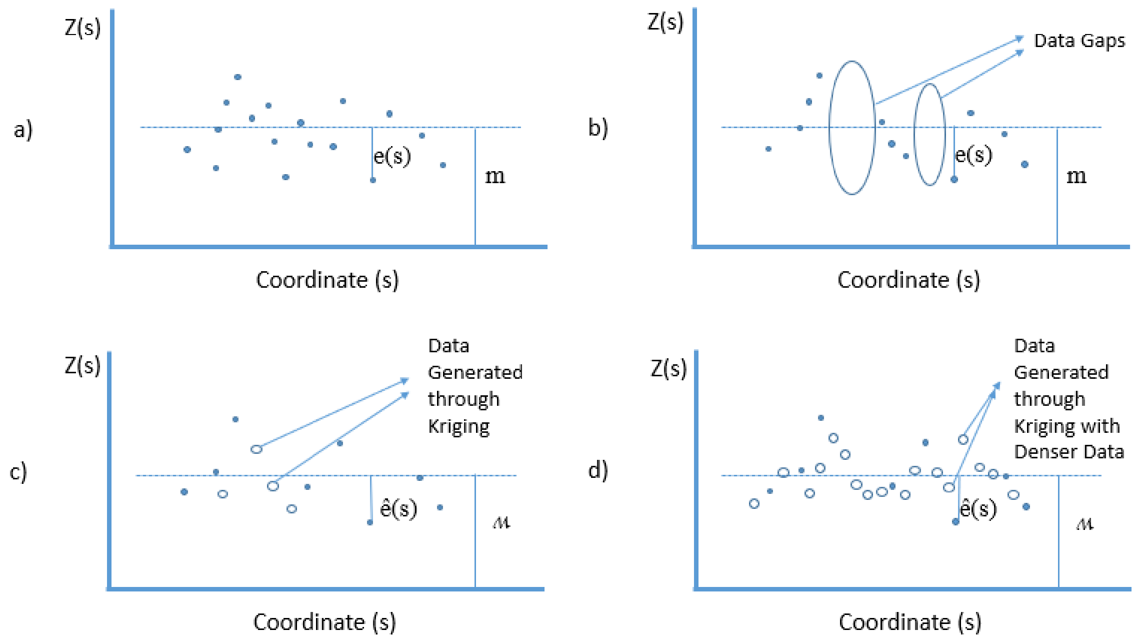


Figure 4-3 Example Kriging predictions with original scanned data (a and b) and post-Kriging results (c and d).

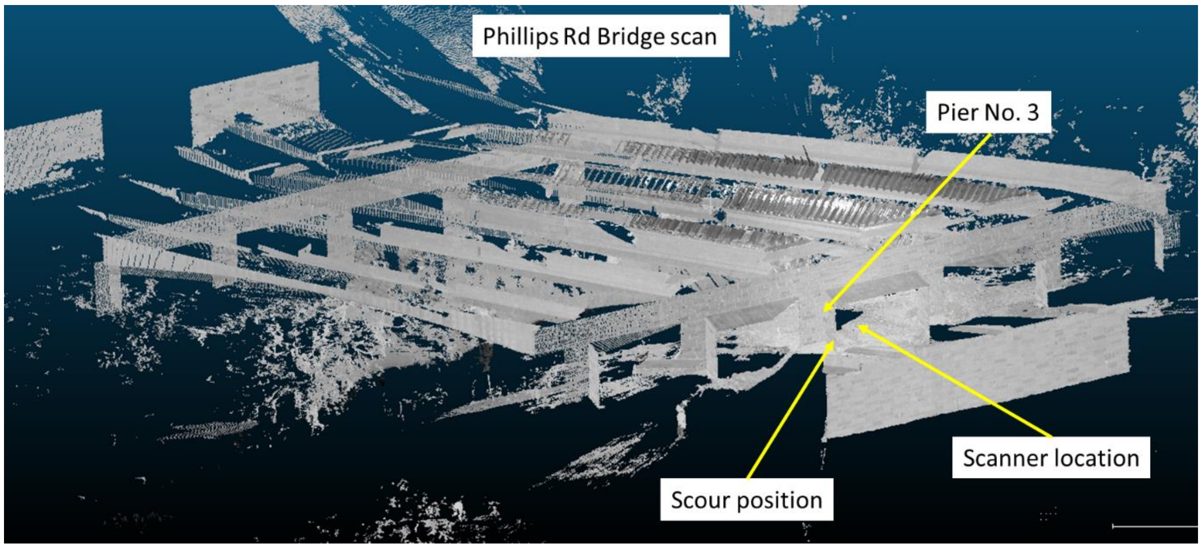


Figure 4-4 Full LiDAR scan with laser scanner position and the location of the studied scour (number of points: 24,248,705).

4.4 Captured Scour Point Cloud and Spatial Density Analysis

To evaluate the effect of Kriging data interpolation on scour quantitative measures (i.e. scour volume and scoured area), a scour hole on a pier of the Phillip's Road Bridge (Chavan et al. 2021) is used in this study. The Phillips Road Bridge over Toby Creek (35°18'28.2"N 80°44'16.6"W) consists of three spans supported on prestressed concrete girders and drilled pier foundations. To detect the scour hole, a FARO Focus S 350 scanning laser was used. The FARO scanner uses a near infrared laser of 1,550 nm wavelength and has a maximum shooting range of 350 m. It is recognized that the bridge piles of the Phillips Road bridge are round, but the bridge piers are square in shape. As a result, the laser scanned scour images have a relatively sharp cornered shape shown partially in Figure 4-2b.

For this study, a scanned point cloud was processed by removing everything except the scoured area. It is possible to detect the scours automatically using artificial intelligence (AI), but that is beyond the scope of this paper, which focuses on the scanned data quality with different OK interpolations. Figure 4-4 shows the original unprocessed scan indicating the bridge piers and the data scattered from the surrounding vegetation. The scanner and the scour positions are also identified from the image. For this study, the area surrounding the scour of interest is extracted from Figure 4-4 and is shown in Figure 4-5. As described earlier, the square bridge pier also needs to be extracted from the point cloud before analysis. Finally, a point cloud sample that contains 1,000 spatial points (i.e., control points; see Figure 4-6) is considered for Kriging. The figure also shows the voids present in the point cloud.

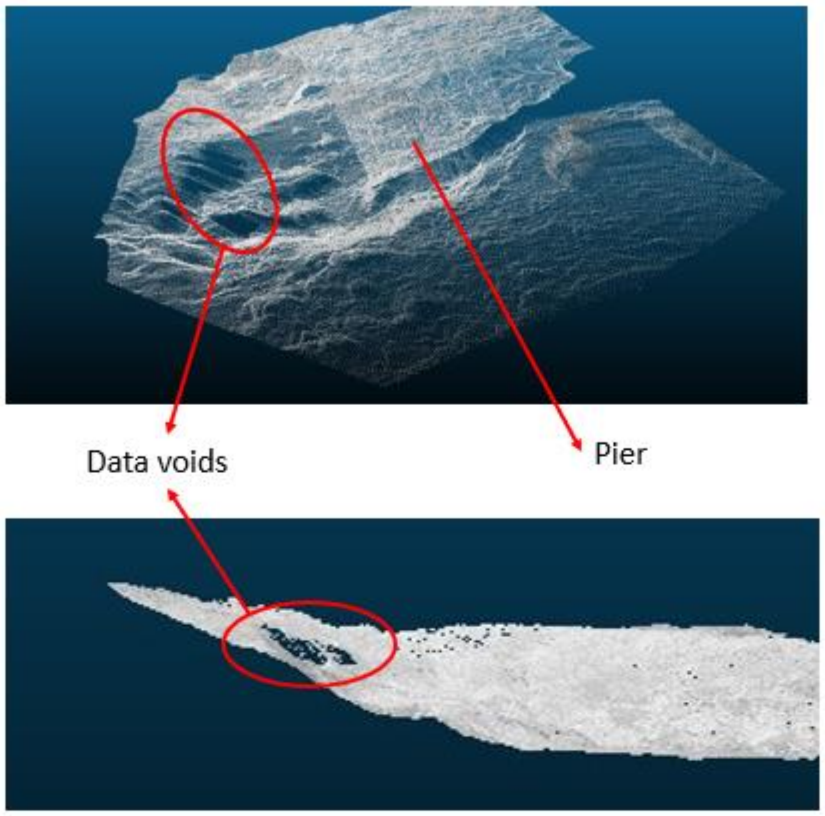


Figure 4-5 Extracted image of the scoured area showing data voids (circled area).

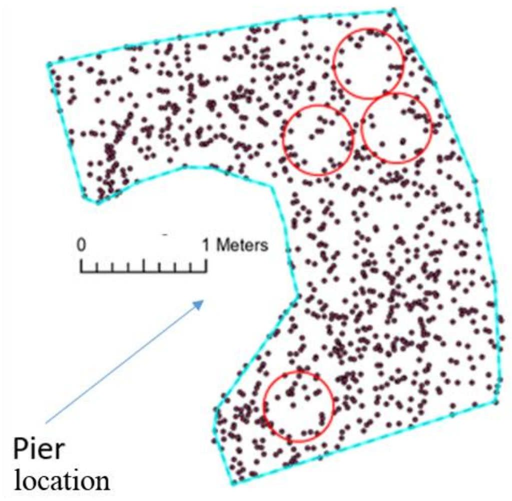


Figure 4-6 Sample scour points with voids marked (in red circles).

The input point cloud and the interpolated points are aligned in such a way that the Z-coordinate is considered the scour depth. The resulting points appear in a uniform grid format with their projections displaced on the XY (spatial) plane. We designed a sensitivity analysis experiment with nine scenarios in terms of point resolution (inter-point distance of interpolated points). Using the Kriging model, nine different point-to-point (inter-point) distances – 500 mm, 200 mm, 100 mm, 50 mm, 20 mm, 10 mm, 5 mm, 2 mm, and 1 mm – are used in this study. The processing was done on a workstation with Intel i7-11800H processor and a RAM of 32GB. Multiple software packages including CloudCompare, MATLAB, and ArcGIS were used in this study.

4.5 Results and Discussions

Figure 4-7 shows the semivariance plot with its range, sill and nugget effect marked. The Kriging results are shown in Figure 4-8. Table 4-1 summarizes the results of this study, where the time reported in the table includes the time taken for scour identification, segmentation, sampling, boundary drawing, interpolation, and quantification processes.

The first scenario, using a point resolution of 500 mm, generated 31 new points. From visual inspection (Figure 4-9), these points did not fully fill the voids in the input cloud. The projected 2D area of the point cloud on the XY plane was 7.47 m^2. This value was used in calculating the point density of the generated point clouds.

Table 4-1 Results of the experiment used in this study.

Scenario No.	Point to Point distance	No. of points generated	Point density	Processing time	Surface area	Volume
	mm		points/m^2	second	m^2	m^3
1	500	31	4.15	306	4.49	2.56
2	200	185	24.76	307	6.37	3.25
3	100	743	99.43	307	7.20	3.59
4	50	2,991	400.25	307	7.68	3.76
5	20	18,680	2,499.70	310	7.95	3.86
6	10	74,731	10,000.29	330	8.07	3.89
7	5	298,861	39,992.74	367	8.17	3.91
8	2	1,867,735	249,935.03	596	8.33	3.92
9	1	7,471,074	999,758.06	1,890	8.46	3.92

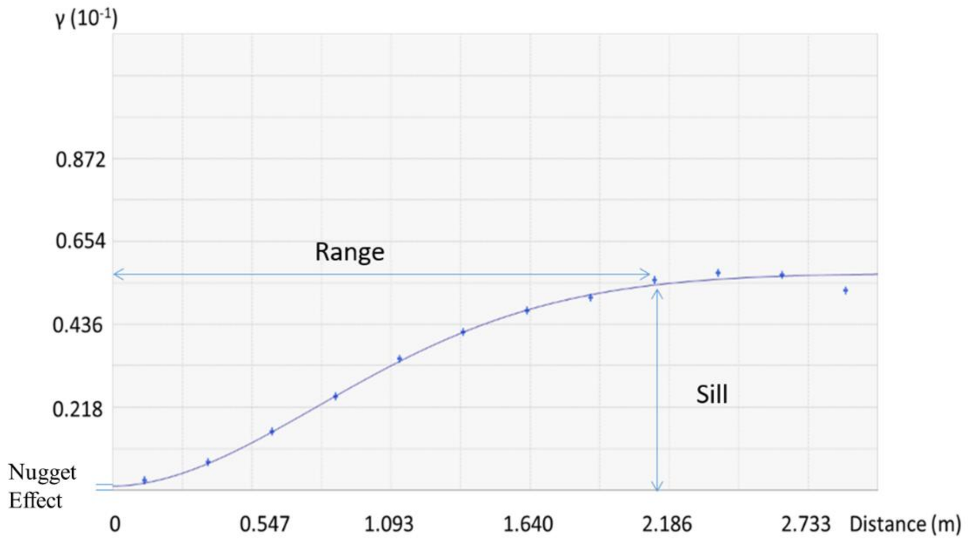


Figure 4-7 Semi-variance graph (semivariogram) for the selected point cloud (γ: semivariance).

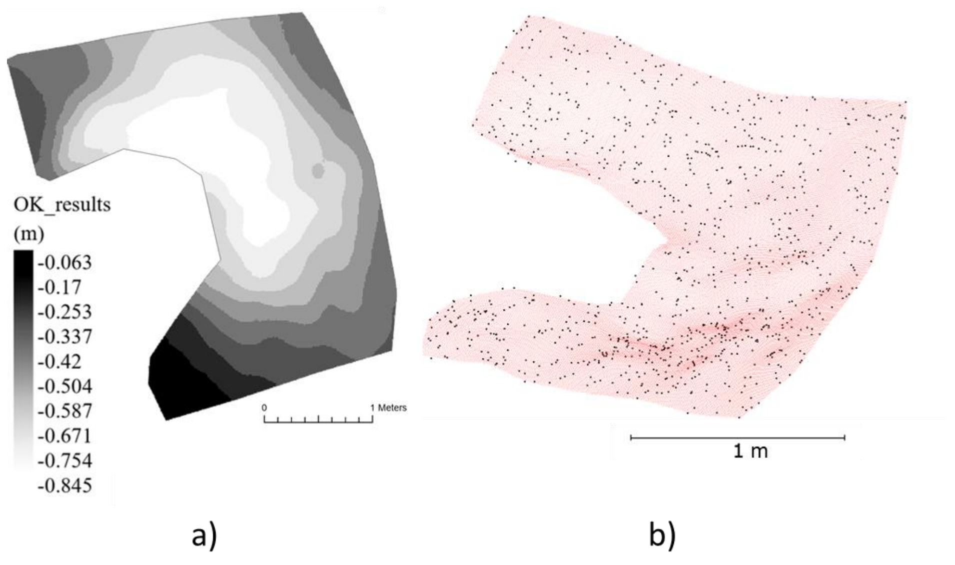


Figure 4-8 Ordinary Kriging results a) raster plot (OK: Ordinary Kriging); b) point cloud view.

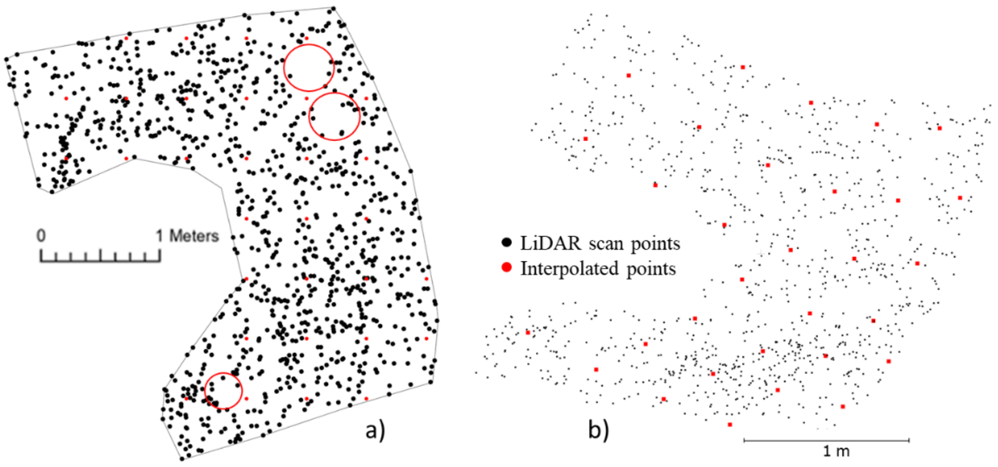


Figure 4-9 Point cloud and interpolated points with point-to-point distance of 500 mm a) XY view b) 3D view.

Figure 4-10 shows the results for scenario 2, with decreased point-to-point distance of 200 mm. It had a point density approximately 6 times greater than the 500 mm scenario. The processing time increased marginally, and the data voids were still visible. When the data density increased further such as in scenario 3 (Figure 4-11), there were no visible voids. The time taken for the processing was about the same as in the previous scenarios. The scour depth, the surface area and volume are parameters that are important for scour monitoring and assessing the health of a bridge structure. From the figures, there is very little change in the maximum depths observed using different resolutions of data interpolation. However, the differences are clearly observed in both surface area and volume. The surface area and volume for the scenario 3 point cloud were 7.20 m^2 and 3.59 m^3, respectively, and these are different from the actual values of the Kriged surface (scenario 1) by 18% and 8.6%, respectively. Further decreasing the point distance resulted in an increased number of points generated whilst creating a decrease in the differences in quantification values. Figure 4-13 shows this observation clearly, where the x-axis is presented in natural logarithm of the spatial resolution. As the number of points increased, the differences in quantification parameters decreased. Naturally, more interpolation points would correspond to a better-defined surface. Thus, the decrease was faster when the initial number of points were on the lower side. Consequently, processing times increased by just a few milliseconds to hundreds of seconds. Figure 4-14 shows the increase in processing time with respect to decrease in inter-point distances.

One of the critical questions that inspired the current study is “What is the optimal interpolation density?” Based on the nine scenarios generated, the ‘optimal’ point cloud density is identified with point distances about 20 mm, which has less than 10% difference in surface area (when compared to the original point cloud) and the corresponding volume difference is lower than 2%.

To demonstrate the effect of temporal changes in scouring, we delve into more detailed discussion by comparing the scour scans at two different times in the following section.

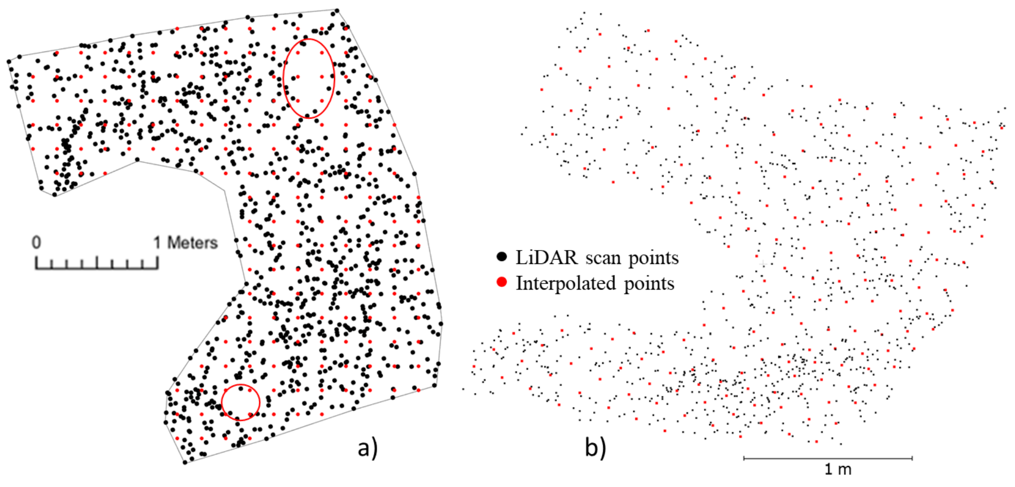


Figure 4-10 Point cloud and interpolated points with point-to-point distance of 200 mm a) XY view b) 3D view.

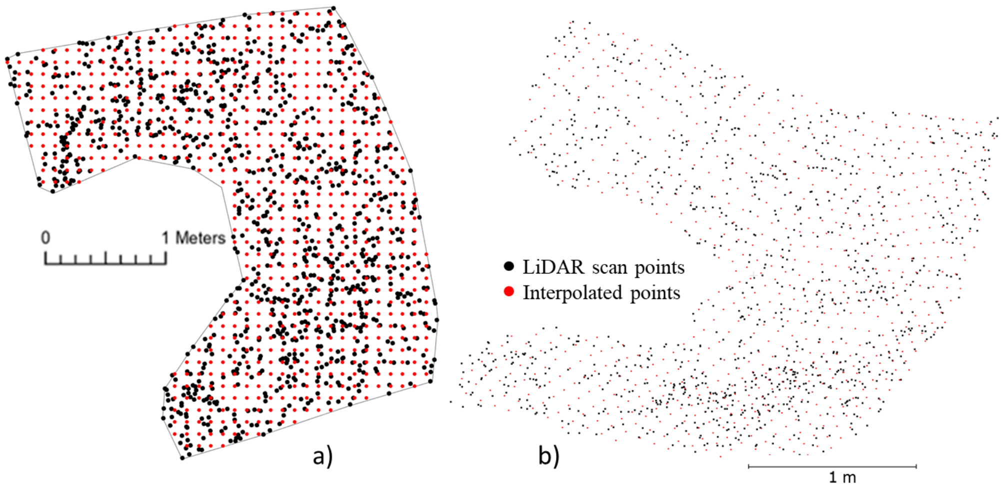


Figure 4-11 Point cloud and interpolated points with point-to-point distance of 100 mm a) XY view b) 3D view

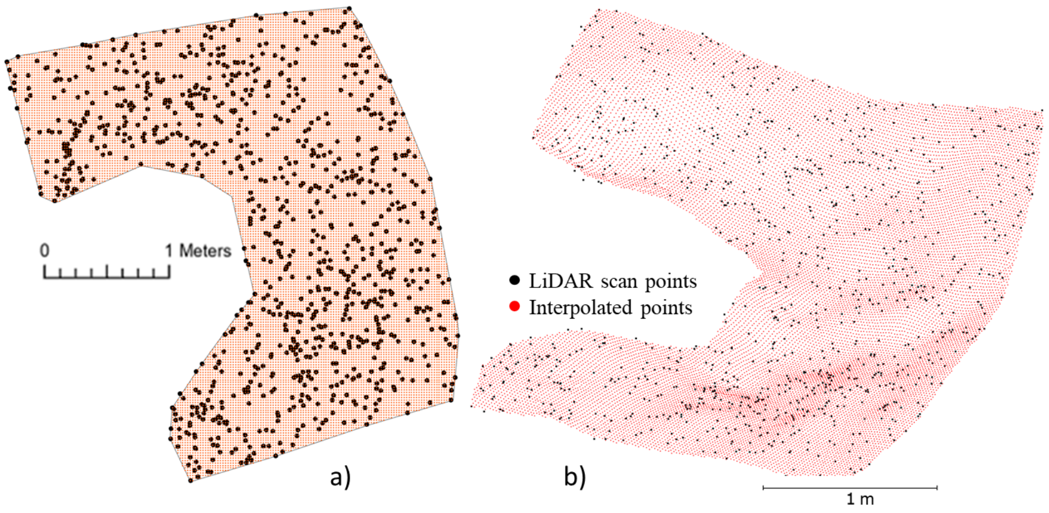


Figure 4-12 Point cloud and interpolated points with point-to-point distance of 20 mm a) XY view b) 3D view

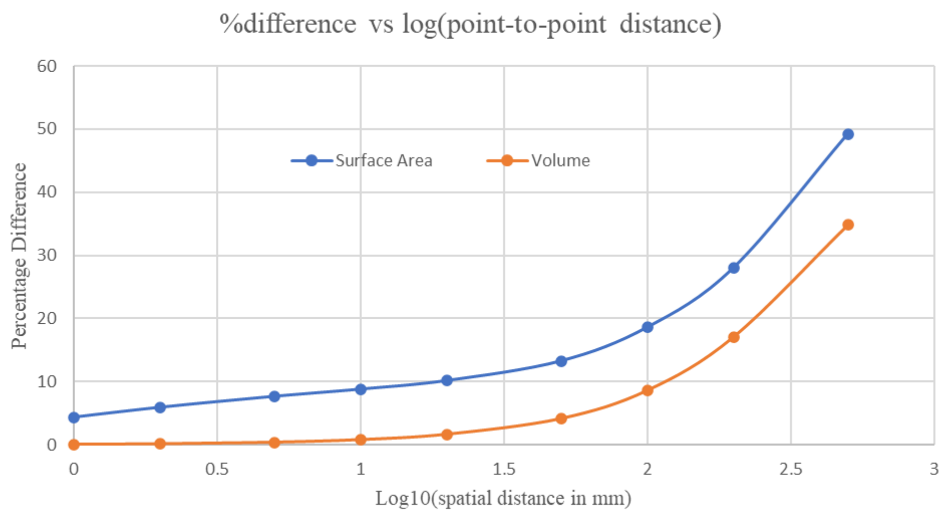


Figure 4-13 Percentage difference in surface area and volume with decreasing point-to-point distances.

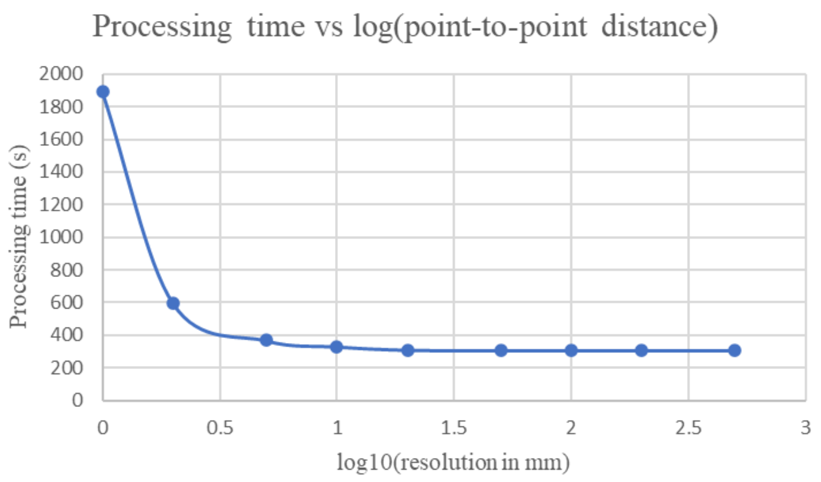


Figure 4-14 Processing times increase with decreasing point distances

4.6 Scour Change Over Time

In order to study the development of scour over time, the Phillips Road bridge scour was scanned multiple times over a period of several months. The scour described in the previous section was scanned on June 9, 2020. A prior scan was taken on August 17, 2019, and is used here to describe how it had changed over the course of ten months. The LiDAR point clouds from 2019 and 2020 are referred to as PointCloud1 (PC1) and PointCloud2 (PC2), respectively. The LiDAR was positioned approximately at the same location for both scans.

Visually inspecting the two point clouds indicates that the PC1 scour had a greater number of small loose rocks when compared to PC2. This means that some of the rocks were washed away during the scouring process, thus PC2 has a smoother surface than PC1. The positions of the rocks were also different, indicating that flooding over the ten-month period had caused the rocks to move. As a result, many of the locations of the voids in the scans were not identical. However, the nature of the voids closest to the scanner location was similar in both PC1 and PC2, both due to the sudden change in ground elevation due to scour depth. However, PC1 had a more gradual change, and thus smaller voids. Figure 4-15 shows the segmented point cloud of the study scour in PC1. Voids were noted in the point cloud and OK was used to interpolate the data using the same procedure. The raster plot of the interpolated point cloud is shown in Figure 4-16.

Comparing the interpolated data from PC1 and PC2, it is clear that the scour shape had also changed over the months. The change in scour parameters also reflects this observation. The maximum scour depth increased from 0.833 m in PC1 to 0.845 m in PC2. This increase can also be noticed visually from the location of the pier cap - While the pier cap was partially visible on both the scans, the pier cap from PC2 was more apparent because of deeper scour. The volumes of the scour remained approximately equal for both PC1 and PC2 scans and is about 3.91 m^3. However, surface areas decreased from 8.94 m^2 to 8.46 m^2, which may be due to different scanning angles.

Last, it should be recognized that this study focuses on the scouring process of a pier-on-bank scenario – The pier is situated on the riverbank, which allows the scanning of a 'dry' scour hole, thus, the laser energy returned is not affected by water at the bottom of the scour hole. In contrast, a scour that occurs to a pier in the riverbed would not be visible to the terrestrial LiDAR. In such cases, a different frequency LiDAR (e.g., bathymetric LiDAR) can be used. Unfortunately, the scan data alone cannot explain the scouring history of the Phillip's Road bridge and additional data from the hydraulic and hydrological processes of the Toby Creek will be needed.

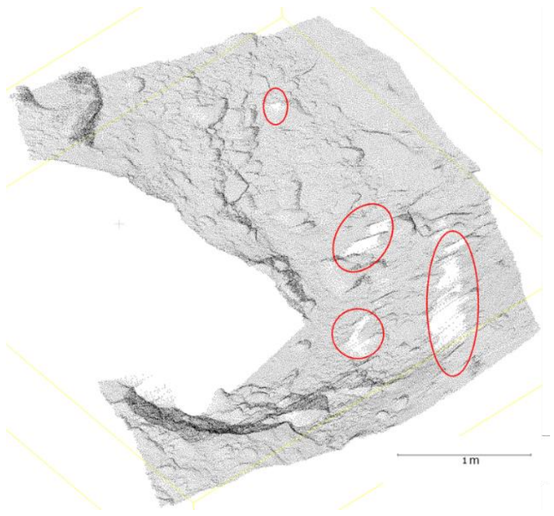


Figure 4-15 Phillips Road bridge scour point cloud as on Aug 17, 2019. Data voids are marked in red circles.

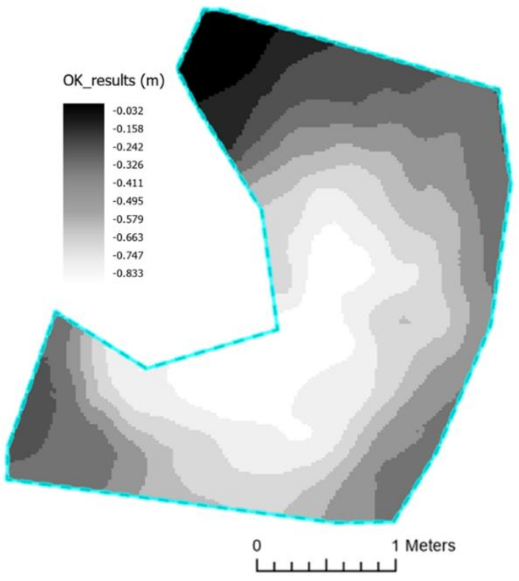


Figure 4-16 Raster plot of OK results (Aug 2019 scour).

4.7 Conclusions

In this study, a LiDAR scan of a scoured hole from a pier of the Phillips Road Bridge was identified with data voids generated to investigate the possible obstruction of the LiDAR scan from vegetation surrounding the pier or physical site constraints which might limit the viewscape of a LiDAR device. OK was performed on the segmented data to fill in the voids. The results show that the OK procedure generated data points that can accurately fill the spatial voids of the scanned scour hole. The Kriging result was used to generate point clouds with varying point density. It was observed that a spatial resolution of 20 mm performed well in terms of accuracy of scour surface area, volume, and processing time and can be considered the 'optimal' resolution for interpolated scour point cloud. A different scan performed at the same location with a gap of ten months between them was used to compare the scour parameters. From the results, scour depth had increased, surface area had decreased, and the volume had remained the same at the end of the ten-month period.

LiDAR scans with the suggested Kriging data interpolation technique can be used to reliably quantify scour parameters such as the scoured surface area and the scoured volume. The data can also be used to define a reference surface for the scour point data, allowing bridge engineers to quantify the actual volume and area of the mass/volume losses during the hydrodynamic processes of scouring over time.

However, it is acknowledged that the natural scour process is not straightforward – as the comparison of the two scans of the same scour indicated. Future studies should include scans before and after each significant storm over an extended period. Current scour effect rating is only based on scour depth measurements, which is insufficient to establish the stability analysis of a bridge due to scouring. The continued use of LiDAR scans can potentially help devise more robust scour quantifiers for assessing the overall stability of a bridge.

4.8 References

Arneson, L., Zevenbergen, L., Lagasse, P., and Clopper, P. (2012). "Evaluating scour at bridges." National Highway Institute (US).

Bian, H., Chen, S. E., and Liu, W. (2017). "Error sources in proccessing lidar based bridge inspection."

Chavan, V. S., Chen, S.-E., Shanmugam, N. S., Tang, W., Diemer, J., Allan, C., Braxtan, N., Shukla, T., Chen, T., and Slocum, Z. (2021). "An Analysis of Local and Combined (Global) Scours on Piers-on-Bank Bridges." *CivilEng*, 3(1), 1-20.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*.

Da Costa, J. J., Chainet, F., Celse, B., Lacoue-Nègre, M., Ruckebusch, C., and Espinat, D. (2018). "Comparing Kriging, Spline, and MLR in Product Properties Modelization: Application to Cloud Point Prediction." *Energy & Fuels*, 32(4), 5623-5634.

Dai, K., Boyajian, D., Liu, W., Chen, S.-E., Scott, J., and Schmieder, M. (2014). "Laser-based field measurement for a bridge finite-element model validation." *Journal of Performance of Constructed Facilities*, 28(5), 04014024.

Fuchs, P., Washer, G., Chase, S., and Moore, M. (2004). "Laser-based instrumentation for bridge load testing." *Journal of Performance of constructed facilities*, 18(4), 213-219.

Hui, Z., Hu, Y., Yevenyo, Y. Z., and Yu, X. (2016). "An improved morphological algorithm for filtering airborne LiDAR point cloud based on multi-level kriging interpolation." *Remote Sensing*, 8(1), 35.

Isaaks, E. H., and Srivastava, R. M. (1989). *An Introduction to Applied Geostatistics*, Oxford University Press, New York.

Krige, D. G. (1951). "A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand." *Journal of the Chemistry, Metallurgy and Mining Society of South Africa*(51(6)), 119-139.

Lagasse, P. F., Zevenbergen, L., Spitz, W., and Arneson, L. (2012). "Stream stability at highway structures." United States. Federal Highway Administration. Office of Bridge Technology.

Lewis, Q. W., Edmonds, D. A., and Yanites, B. J. (2020). "Integrated UAS and LiDAR reveals the importance of land cover and flood magnitude on the formation of incipient chute holes and chute cutoff development." *Earth Surface Processes and Landforms*, 45(6), 1441-1455.

Lin, Y.-C., Liu, J., Cheng, Y.-T., Hasheminasab, S. M., Wells, T., Bullock, D., and Habib, A. (2021). "Processing Strategy and Comparative Performance of Different Mobile LiDAR System Grades for Bridge Monitoring: A Case Study." *Sensors*, 21(22), 7550.

Melville, B. W., and Coleman, S. E. (2000). *Bridge scour*, Water Resources Publication.

Meng, Q., Liu, Z., and Borders, B. E. (2013). "Assessment of regression kriging for spatial interpolation–comparisons of seven GIS interpolation methods." *Cartography and geographic information science*, 40(1), 28-39.

Montero, J.-M., Fernández-Avilés, G., and Mateu, J. (2015). *Spatial and spatio-temporal geostatistical modeling and kriging*, John Wiley & Sons.

Nouwakpo, S. K., Weltz, M. A., and McGwire, K. (2016). "Assessing the performance of structure-from-motion photogrammetry and terrestrial LiDAR for reconstructing soil surface microtopography of naturally vegetated plots." *Earth Surface Processes and Landforms*, 41(3), 308-322.

Oliver, M. A., and Webster, R. (2015). *Basic steps in geostatistics: the variogram and kriging*, Springer.

Schabenberger, O., and Gotway, C. A. (2017). *Statistical methods for spatial data analysis: Texts in statistical science*, Chapman and Hall/CRC.

Stovall, A. E., Diamond, J. S., Slesak, R. A., McLaughlin, D. L., and Shugart, H. (2019). "Quantifying wetland microtopography with terrestrial laser scanning." *Remote Sensing of Environment*, 232, 111271.

Suro, T. P., Huizinga, R. J., Fosness, R. L., and Dudunake, T. (2020). "Assessment of bridge scour countermeasures at selected bridges in the United States, 2014–18." US Geological Survey.

Watson, C., Chen, S.-E., Bian, H., and Hauser, E. (2013). "LiDAR scan for blasting impact evaluation on a culvert structure." *Journal of performance of constructed facilities*, 27(4), 460-467.

Zhang, X., Meng, X., Li, C., Shang, N., Wang, J., Xu, Y., Wu, T., and Mugnier, C. (2021). "Micro-Topography Mapping through Terrestrial LiDAR in Densely Vegetated Coastal Environments." *ISPRS International Journal of Geo-Information*, 10(10), 665.

5 OVERALL CONCLUSIONS

To address the issue of bridge scour monitoring, this dissertation explored effective approaches to identify and quantify local scour on pier-on-bank bridges addressing specifically the issues of LiDAR point cloud data processing pertaining to the data void infilling and automated volumetric and surface area quantification. As scour is the most critical and hidden risk for a bridge, the contributions from this research are many folds:

1. It establishes the potential of LiDAR scans as a viable technology for scour detection.
2. It suggests automation for scour detection in a massive point cloud data, and
3. It explores the more theoretical issues of quantifying scour hole dimensions.

The conclusions from this research are summarized as follows.

1. Bridge failures caused by scour account for most of the bridge failures in the US, resulting in a substantial repair and replacement costs.
2. Scour depth, surface and volume are parameters that can be used to estimate the severity of a scour.
3. LiDAR technology is a more effective monitoring technique for scour investigations. It has the advantages of being more accurate, consistent, faster, lower cost, light independent, and ground control point independent over other surveying techniques.
4. The Phillips Road Bridge over Toby Creek (35°18'28.2"N 80°44'16.6"W) located at the University of North Carolina at Charlotte was known to have local scour and was used as a case study.
5. No previous research was found on the use of AI to detect scour. 3D PCN method was used to reconstruct scour shape from LiDAR point cloud. One thousand sets of synthetic data were used to train this network and it was applied on a real-world scour point cloud.

6. Discrete chamfer metric was used as the loss function governing the network training. It stabilized at 0.000122 at the 86th epoch.
7. This network currently has an average accuracy of 63% in labelling the points as scour and non-scour.
8. OK process was performed on a terrestrial LiDAR scour point cloud data to populate the data voids that were present. A reference plane was manually selected by choosing three points and the dataset was transformed in its orientation to Cartesian (xyz) coordinates such that z was the scour depth. The $\cos(r_{angle})$ value was 0.9985 for the case study scour point cloud, meaning that the reference plane chosen was almost overlapping the ground plane.
9. Kriging can result in varying point densities, with processing times increasing with larger number of points.
10. Kriging results with a wide range of point densities were studied, varying from just 31 new points at 500 mm distance to millions of points at 1 mm distance.
11. Voids were filled with a resolution of 100 mm. However, the surface area and volume for this point cloud were $7.20m^2$ and $3.59m^3$ respectively, but those were off by 18% and 8.6% from the actual values of the Kriged surface.
12. As the number of points increased, the differences in surface area and volume decreased. The decrease was faster when the density was low.
13. Processing times increased by just few milliseconds to hundreds of seconds as the number of points increased.
14. This method is applicable for any scour point cloud, not just from a terrestrial LiDAR. The results from this can be used to make an informed decision on the health of a bridge structure.

6 SCOPE FOR FUTURE RESEARCH

This study revealed a very interesting subject on scour quantification, which currently still limited to scour depth determination during bridge evaluation (Melville et al. 2000).

The use of LiDAR scan to establish the volumetric and area information of a scour can be significant in determining the actual criticality of the changed condition of a bridge support structure. This modern-day tool has already seen a dramatic increase in its applications for the past ten years. What current study is contributing is the potential integration of automatable numeric techniques that can significantly improve the workflow of a hydraulic engineer.

Considering the vast advanced analytical techniques that can be used for point cloud data manipulation, current study is only scratching the surface of study. Figure 6-1 shows the potential evolution of this subject as part of the workflow of scour evaluation. As shown, there are several other techniques that can be experimented for the current subject.

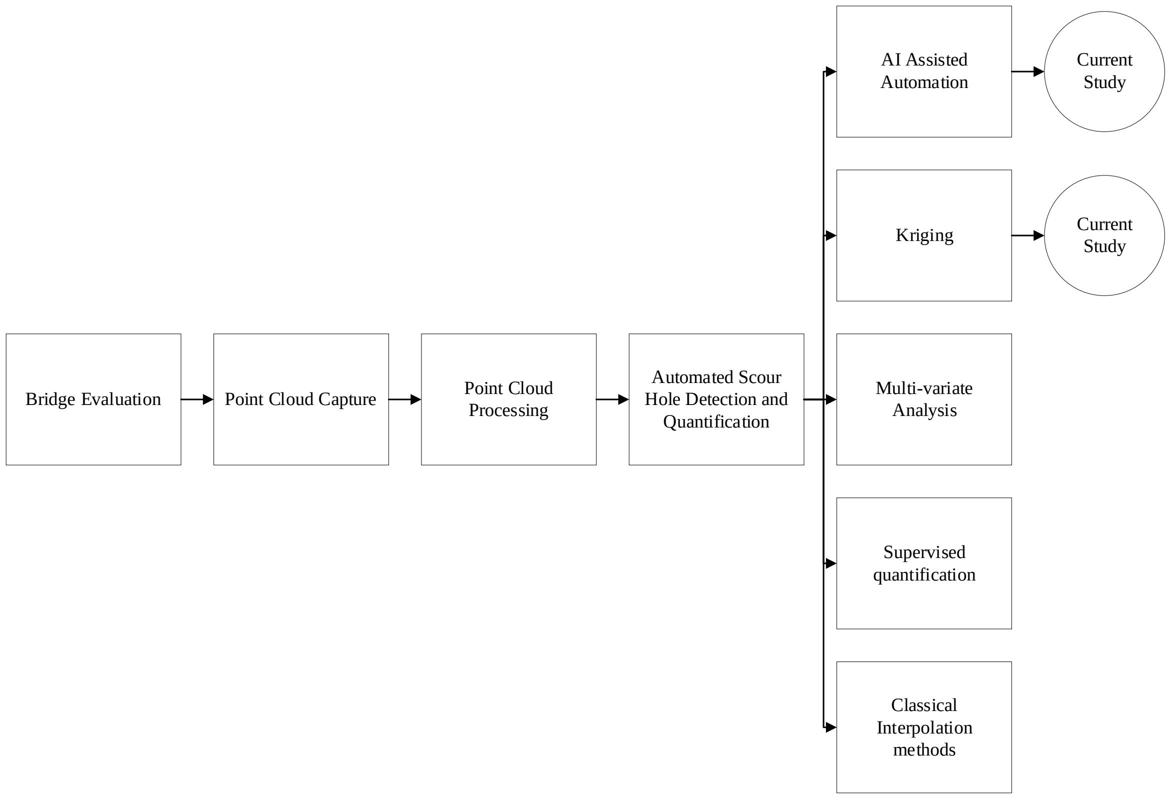


Figure 6-1 Potential future research

REFERENCES

Melville, B.W. and S.E. Coleman, Bridge scour. 2000: Water Resources Publication.

Shirole, A. and R. Holt, Planning for a comprehensive bridge safety assurance program. Transportation Research Record, 1991. 1290(3950): p. 290-005.

Lagasse, P.F., et al., Stream stability at highway structures. 2012, United States. Federal Highway Administration. Office of Bridge Technology.

Prendergast, L.J. and K. Gavin, A review of bridge scour monitoring techniques. Journal of Rock Mechanics and Geotechnical Engineering, 2014. 6(2): p. 138-149.

Liu, W. and S.E. Chen, Reliability analysis of bridge evaluations based on 3D light detection and ranging data. Structural Control and Health Monitoring, 2013. 20(12): p. 1397-1409.

Liu, W., S.-e. Chen, and E. Hasuer, Bridge clearance evaluation based on terrestrial LIDAR scan. Journal of Performance of Constructed Facilities, 2012. 26(4): p. 469-477.

Munoz Rodriguez, J.A., Laser Scanner Technology. 2012.

Hohn, M.E., An Introduction to Applied Geostatistics: by Edward H. Isaaks and R. Mohan Srivastava, 1989, Oxford University Press, New York, 561 p., ISBN 0-19-505012-6, ISBN 0-19-505013-4 (paperback), $55.00 cloth, $35.00 paper (US). Computers & Geosciences, 1991. 17(3): p. 471-473.

Cressie, N.A., Statistics for spatial data. John Willy and Sons. Inc., New York, 1993.

Meng, Q., Z. Liu, and B.E. Borders, Assessment of regression kriging for spatial interpolation–comparisons of seven GIS interpolation methods. Cartography and geographic information science, 2013. 40(1): p. 28-39.

Knotters, M., D. Brus, and J.O. Voshaar, A comparison of kriging, co-kriging and kriging combined with regression for spatial interpolation of horizon depth with censored observations. Geoderma, 1995. 67(3-4): p. 227-246.

Zhao, Y., Birdal, T., Deng, H., Tombari, F., (2019) "3D Point Capsule Networks," Proceedings, IEEE/CVF Conf. Compu. Vis. Pattern Recognition (CVPR), Long Beach, CA, USA, 1009-1018.

APPENDICES

Appendix A. Phillips Road Bridge North Bank Photos

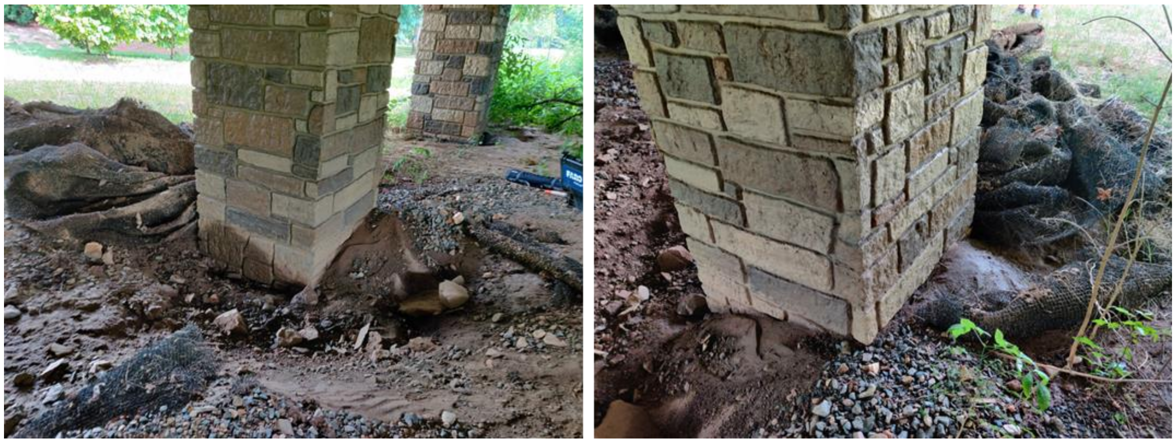


Figure A-1 North Bank Pier No. 2 (June 9, 2020)

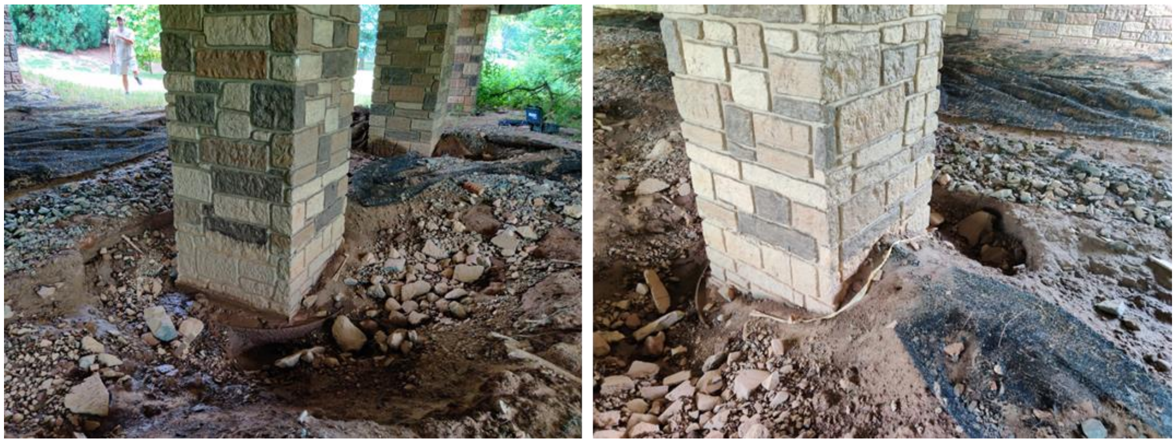


Figure A-2 North Bank Pier No.3 (June 9, 2020)

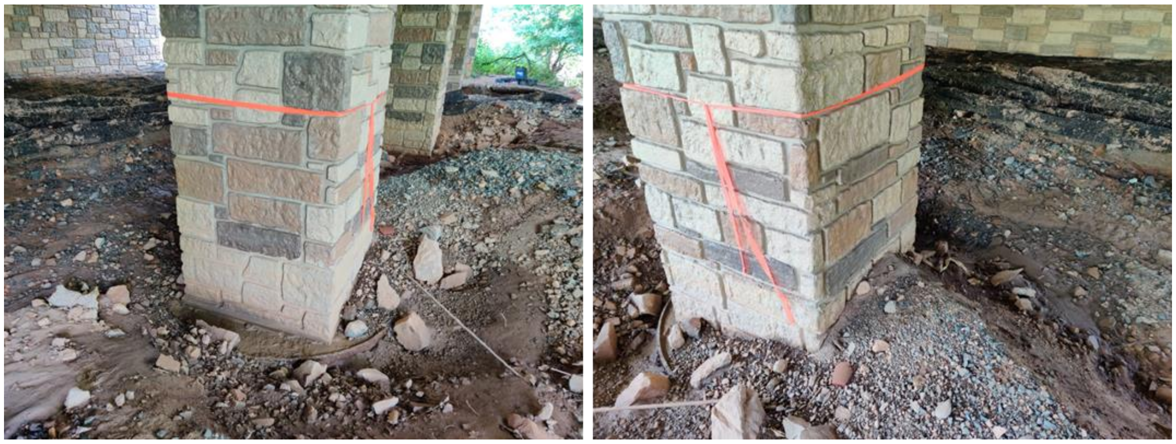


Figure A-3 North Bank Pier No.4 (June 9, 2020)

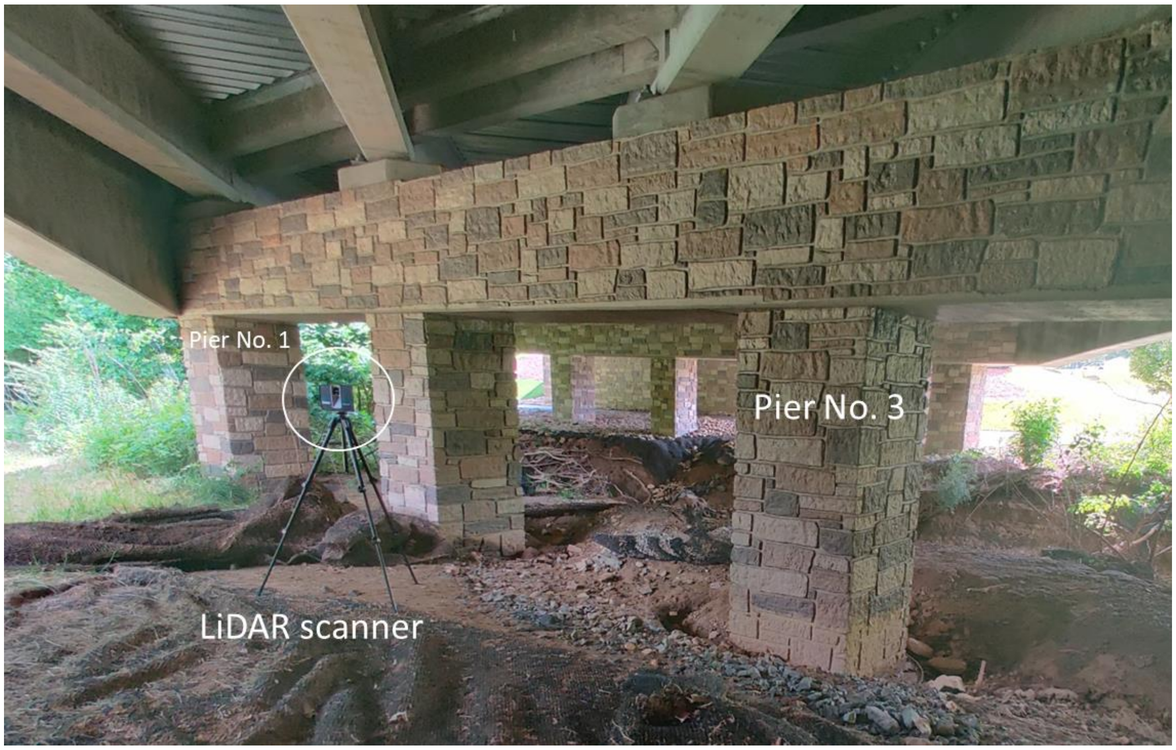


Figure A-4 LiDAR scan in progress (June 9, 2020)

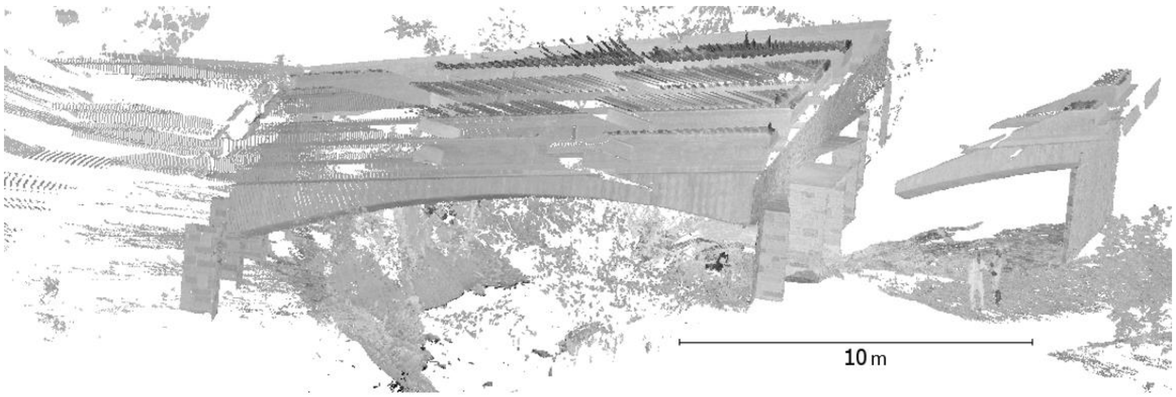


Figure A-5 Sample Phillips Road Bridge LiDAR point cloud visualized using CloudCompare

Appendix B. Phillips Road Bridge North Bank Pier No. 2 Scour Over Time

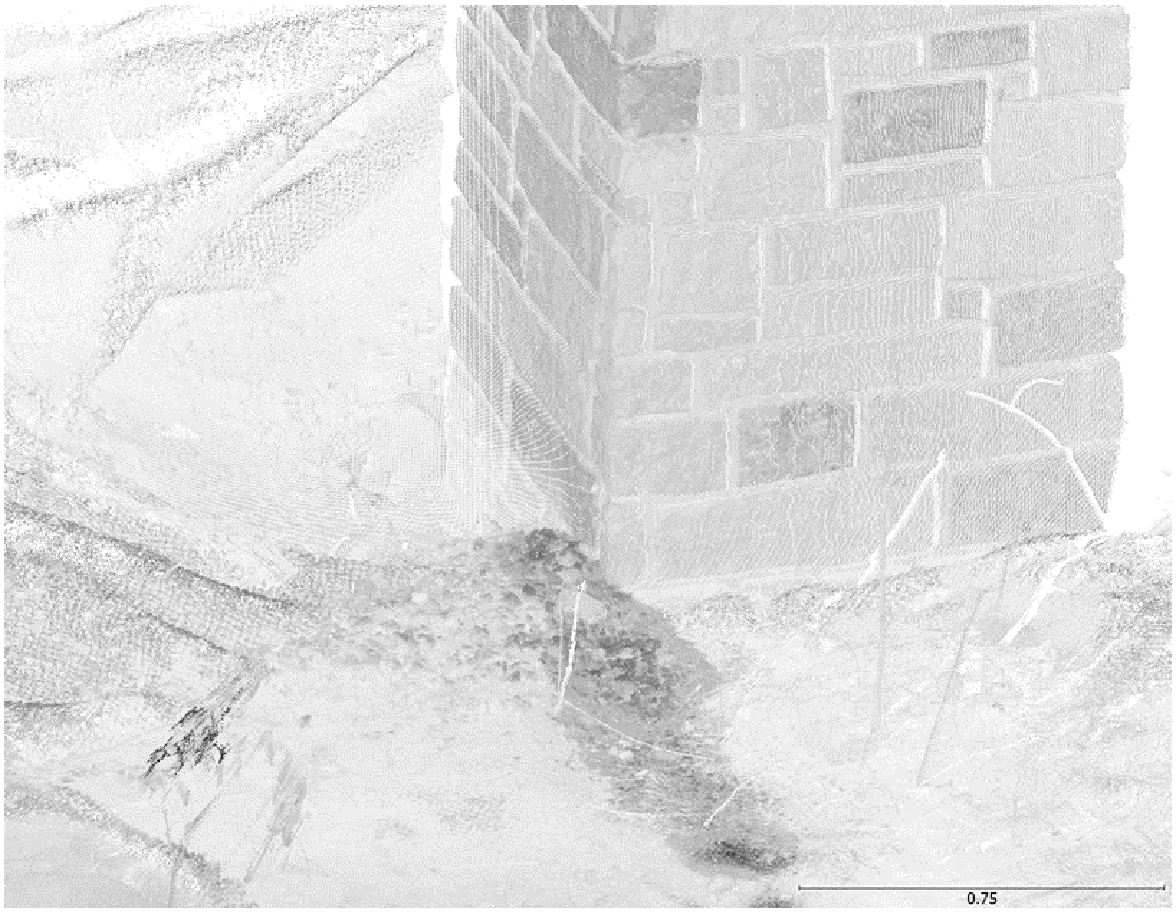


Figure B-1 Point cloud of Phillips Road Bridge North Bank Pier No. 2 and scour as on Feb 11, 2019

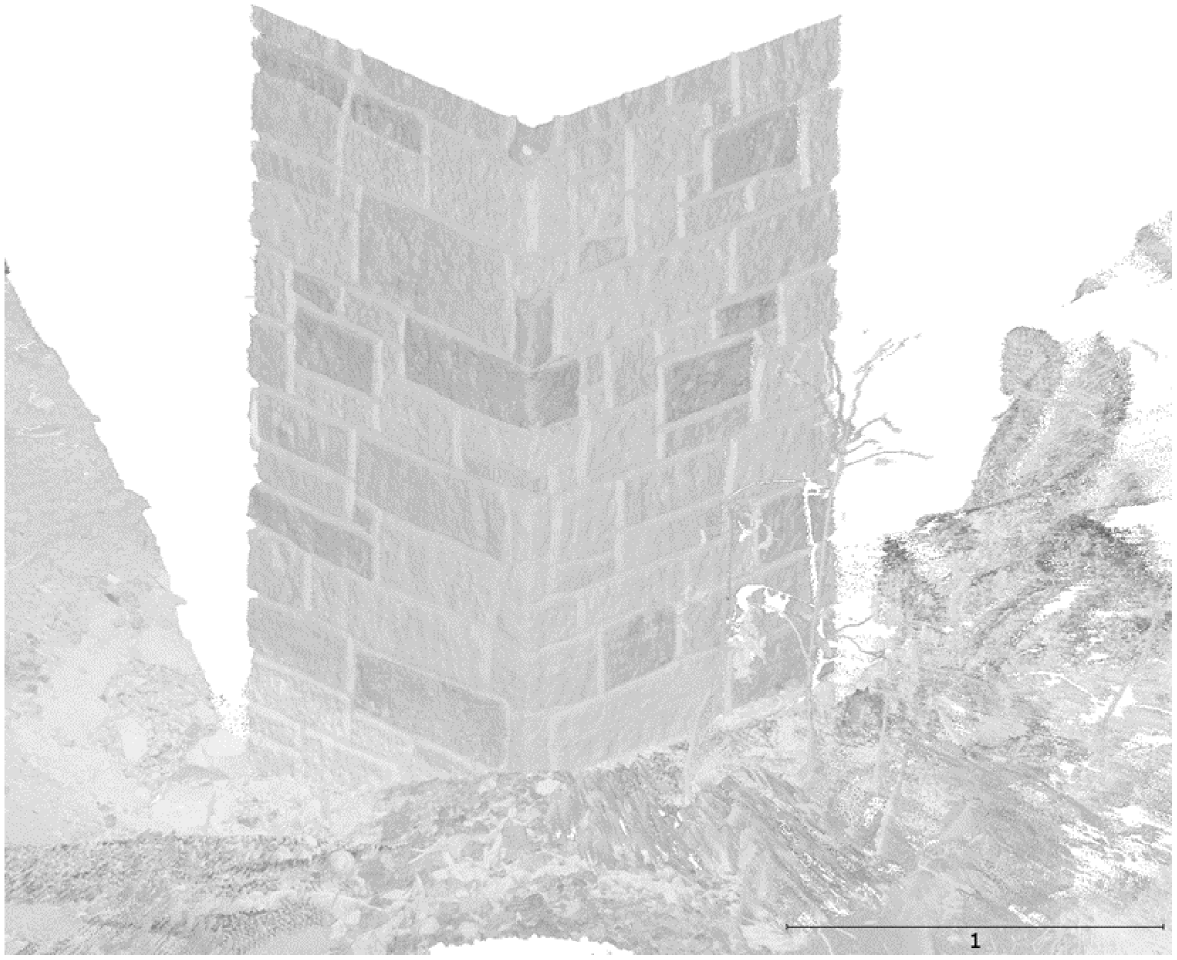


Figure B-2 Point cloud of Phillips Road Bridge North Bank Pier No. 2 and scour as on Aug 17, 2019

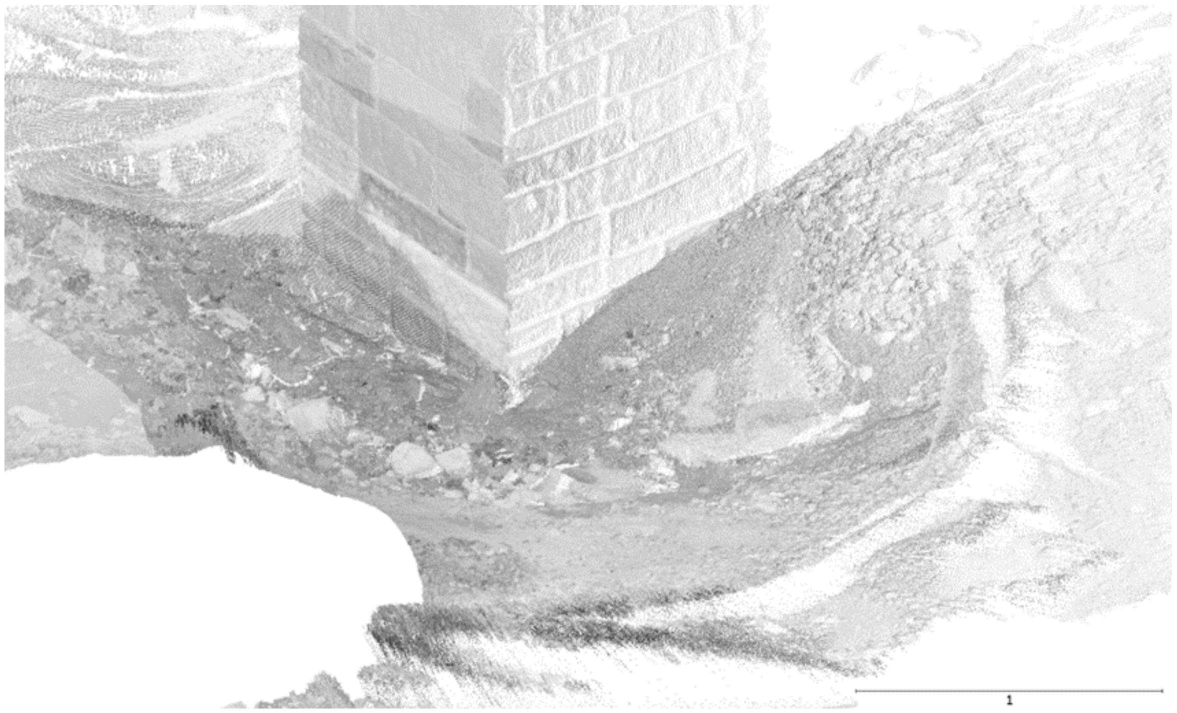


Figure B-3 Point cloud of Phillips Road Bridge North Bank Pier No. 2 and scour as on June 9, 2020

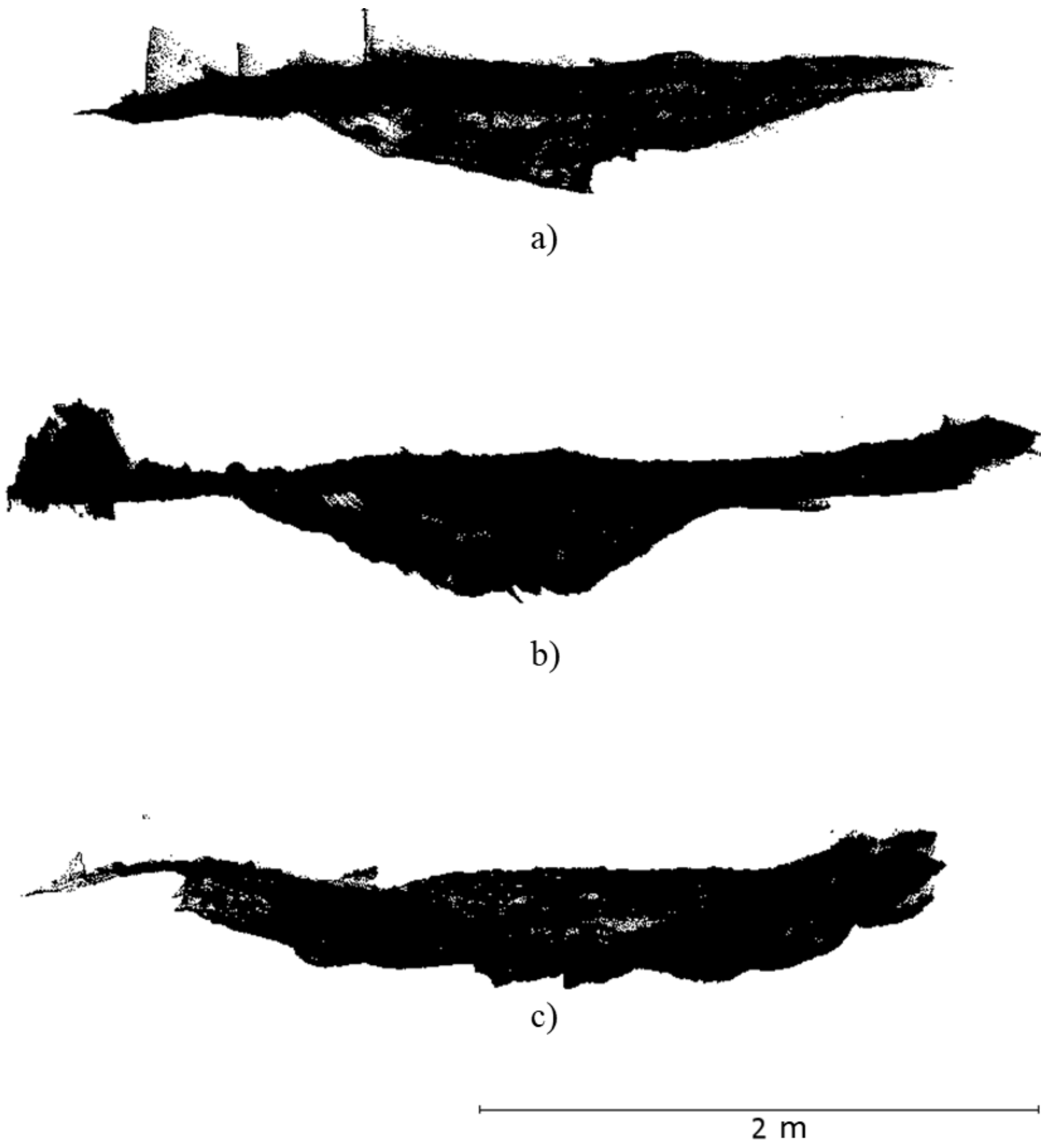


Figure B-4 Pier No. 2 scour point cloud comparison a) Feb 11, 2019; b) Aug 17, 2019; c) June 9, 2020

Table B-1 Pier No. 2 scour parameters over time

Date	Pier No.	Scour Depth (m)	Surface Area (m^2)	Volume (m^3)
2019-02-11	2	0.26	2.05	0.15
2019-08-17	2	0.43	5.72	0.77
2020-06-09	2	0.38	5.48	0.99

Appendix C. Phillips Road Bridge North Bank Pier No. 3 Scour Over Time

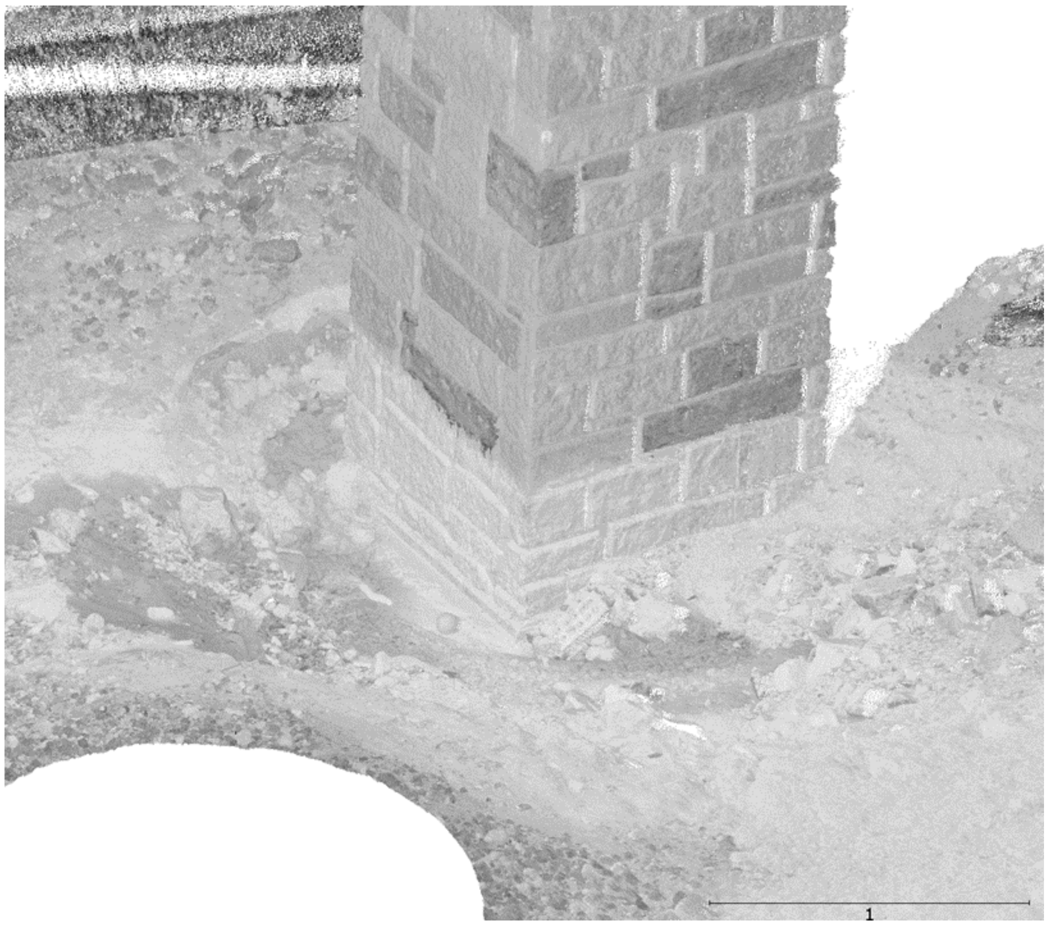


Figure C-1 Point cloud of Phillips Road Bridge North Bank Pier No. 3 and scour as on Aug 17, 2019

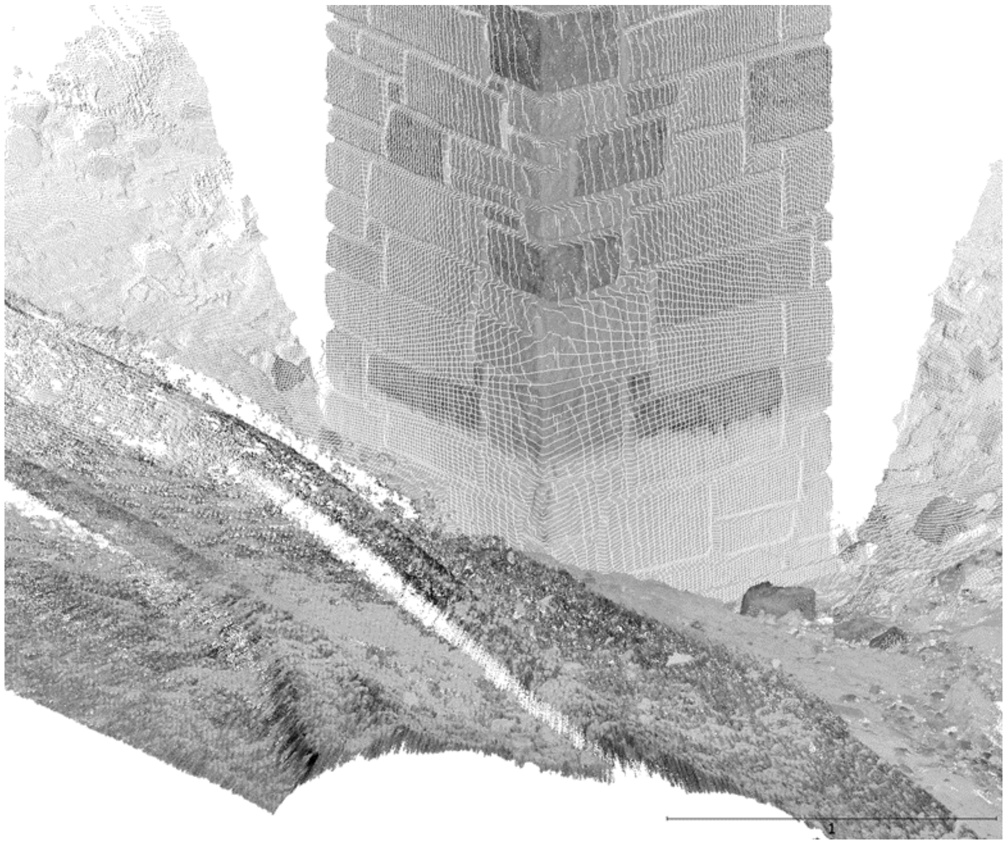


Figure C-2 Point cloud of Phillips Road Bridge North Bank Pier No. 3 and scour as on Jan 24, 2020

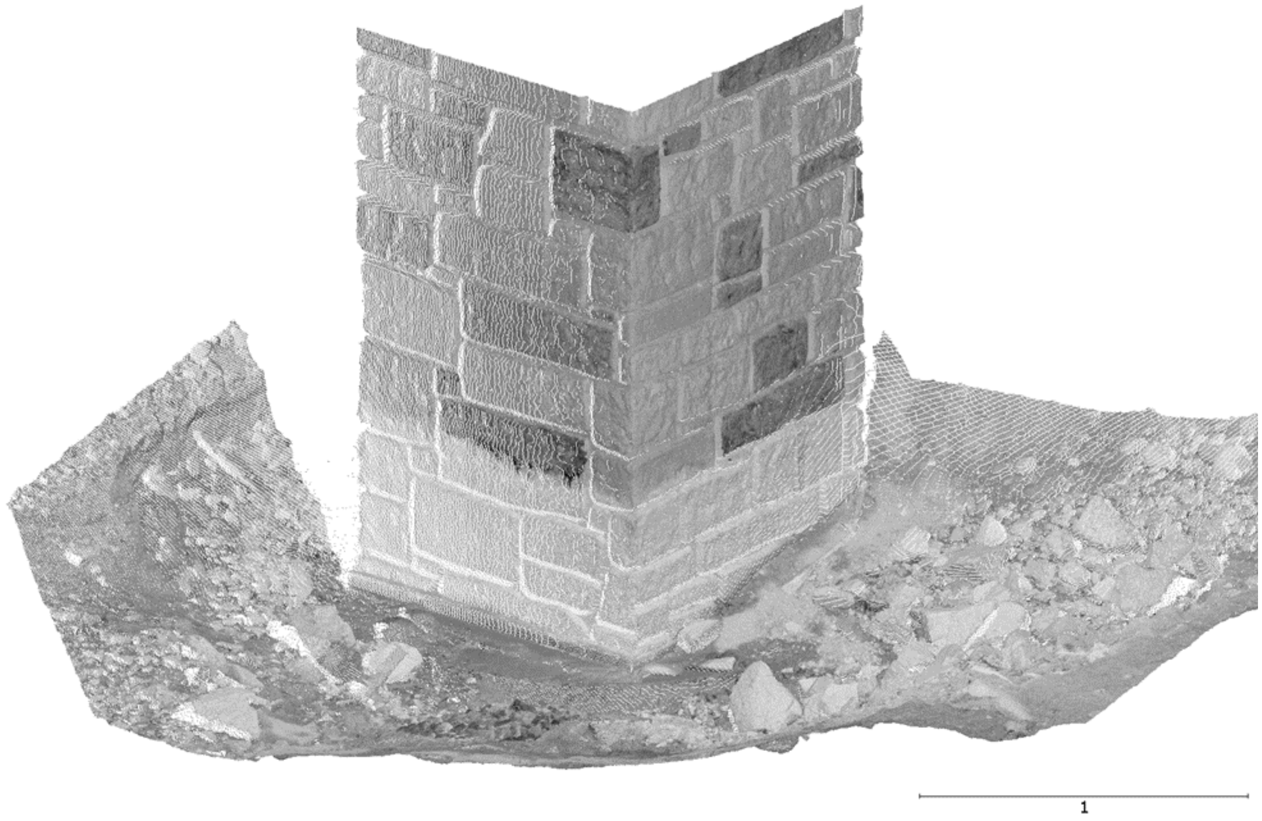


Figure C-3 Point cloud of Phillips Road Bridge North Bank Pier No. 3 and scour as on June 6, 2020

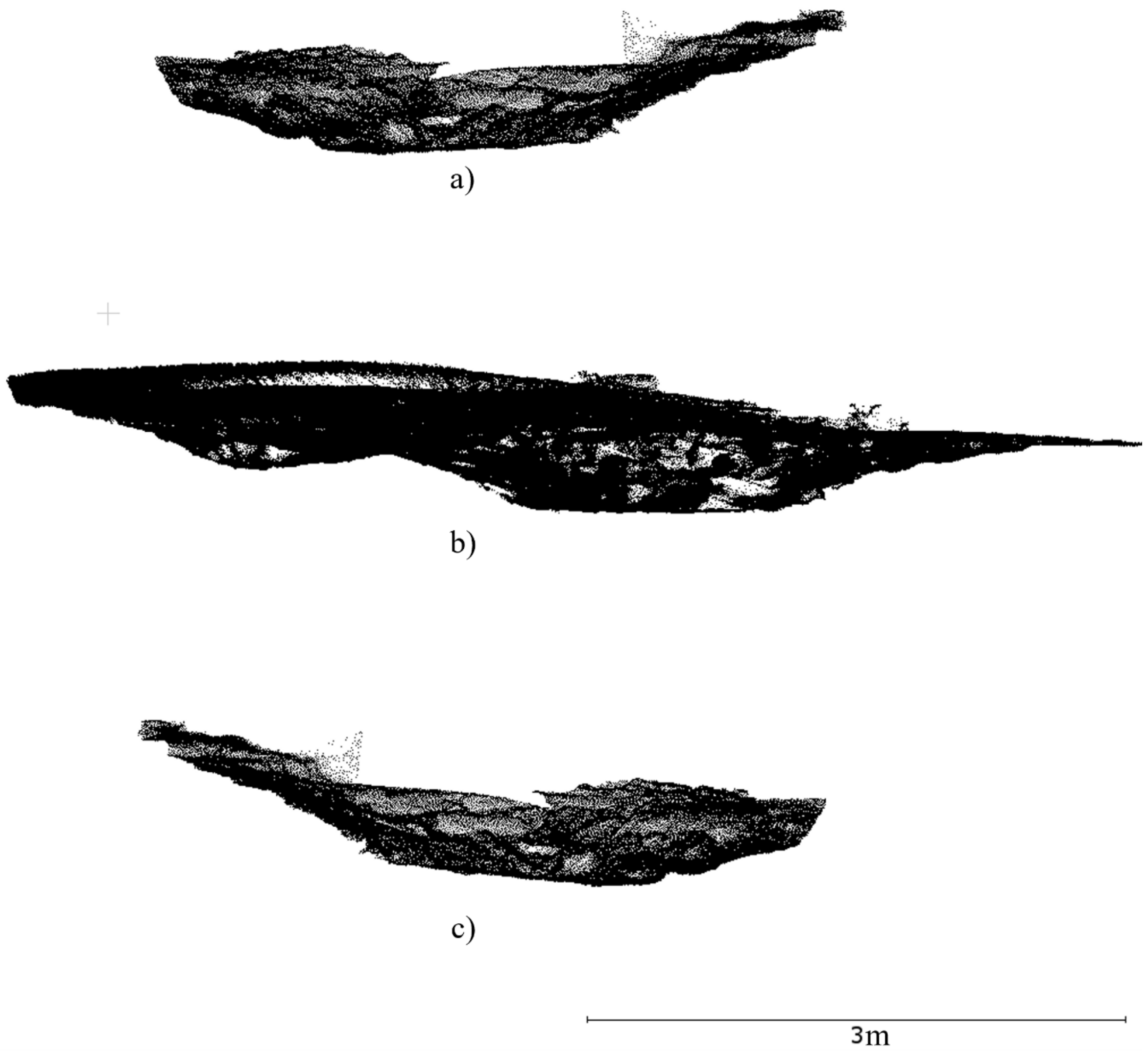


Figure C-4 Pier No. 3 scour point cloud comparison; a) Aug 17, 2019; b) Jan 24, 2020; c) June 9, 2020

Table C-1 Pier No. 3 scour parameters over time

Date	Pier No.	Scour Depth (m)	Surface Area (m^2)	Volume (m^3)
2019-08-17	3	0.83	8.94	3.91
2020-01-24	3	0.58	12.7	2.37
2020-06-09	3	0.85	8.46	3.91

Appendix D. MATLAB Scripts

Generate Gaussian curve – Initial attempt

```
j=1000;                            % number of point clouds
ne=1000;
np=2048;                           % number of points to be sampled
wor_dir=pwd;
dest_pt=[wor_dir '\points\'];
dest_seg=[wor_dir '\segs\'];
for j=1:nj
z = zeros(ne,ne);                  % initialize z matrix
n = 5;                             % number of bumps
sigma = 100;                       % std width of bell curve
max_ht = 50;                       % maximum height
[x,y] = meshgrid(1:size(z,1),1:size(z,2));
% vol_calc=0;                      % initialize for volume using formula
for i=1:n
   % random location of bumps
   xc = randi([size(z,1)/4 3*size(z,1)/4]);
   yc = randi([size(z,2)/4 3*size(z,2)/4]);
   x_c(i)=xc;
   y_c(i)=yc;
   % bell curve
   exponent = ((x-xc).^2 + (y-yc).^2)./(2*sigma^2);
   amplitude(i) = rand()*max_ht;
   z = z + amplitude(i)*exp(-exponent);
z1=reshape(z,ne*ne,1);
x1=reshape(x,ne*ne,1);
y1=reshape(y,ne*ne,1);
z2=[x1 y1 z1];
```

```
m1=randsample(size(z2,1),np);
z3=z2(m1,:);
writematrix(z3,[dest_pt 'ptfile_' num2str(j) '.txt'], 'Delimiter', ' ');
z4=[z3 ones(size(z3,1),1)];             %annotate all points as 1
z4(z4(:,3)>4,4)=2;                      %change annotation of scour points as 2
writematrix(z4(:,4),[dest_seg 'segfile_' num2str(j) '.txt']);
```

Generate Gaussian curve – Increased amplitude

```
nj=1000;                              % number of point clouds
ne=1000;
np=2048;                              % number of points to be sampled
wor_dir=pwd;
dest_pt=[wor_dir '\points\'];
dest_seg=[wor_dir '\segs\'];
for j=1:nj
z = zeros(ne,ne);                     % initialize z matrix
n = 5;                                % number of bumps
sigma = 100;                          % std width of bell curve
max_ht = 200;                         % maximum height
[x,y] = meshgrid(1:size(z,1),1:size(z,2));
for i=1:n
   % random location of bumps
   xc = randi([size(z,1)/4 3*size(z,1)/4]);
   yc = randi([size(z,2)/4 3*size(z,2)/4]);
   x_c(i)=xc;
   y_c(i)=yc;
   % bell curve
   exponent = ((x-xc).^2 + (y-yc).^2)./(2*sigma^2);
   amplitude(i) = rand()*max_ht;
   z = z + amplitude(i)*exp(-exponent);
end
z1=reshape(z,ne*ne,1);
x1=reshape(x,ne*ne,1);
y1=reshape(y,ne*ne,1);
z2=[x1 y1 z1];
m1=randsample(size(z2,1),np);
```

```
z3=z2(m1,:);
writematrix(z3,[dest_pt 'ptfile_' num2str(j) '.txt'], 'Delimiter', ' ');
z4=[z3 ones(size(z3,1),1)];                 %annotate all points as 1
z4(z4(:,3)>10,4)=2;                         %change annotattion of scour points as 2
writematrix(z4(:,4),[dest_seg 'ptfile_' num2str(j) '.txt']);
```

Subsample Point Cloud Data

```
function orient_subsamp(scour_file)
%read data
a = readmatrix(scour_file);
%get xyz data
b = a(:,1:3);
%get reference plane
[s1, s2, s3] = get_refpl(a);
%Translate and rotate data along with the reference plane
b_t=b-s1;
s1_t=s1-s1;
s2_t=s2-s1;
s3_t=s3-s1;
p1=cross(s2_t,s3_t);
p1_n=p1./norm(p1);
r_axis=cross(p1_n,[0 0 1]);
r_a_n=r_axis./norm(r_axis);
r_angle=acos(dot(p1_n,[0 0 -1]));
C=cos(r_angle);
S=sin(r_angle);
t=1-C;
u1=r_a_n(1);
u2=r_a_n(2);
u3=r_a_n(3);
T=[t*u1*u1+C t*u1*u2-S*u3 t*u1*u3+S*u2;
   t*u1*u2+S*u3 t*u2*u2+C t*u2*u3-S*u1;
   t*u1*u3-S*u2 t*u2*u3+S*u1 t*u3*u3+C];
b_new=b_t*T;
%ignore data if it has recurring points
```

```
[c,ia,~]=unique(b_new(:,1:2),'rows','stable');
b_new2=[c b_new(ia,3)];
%random sample data
random1k=randsample([1:size(b_new2,1)],1000);
random10k=randsample([1:size(b_new2,1)],10000);
b_rand1k=b_new2(random1k,:);
b_rand10k=b_new2(random10k,:);
%write data to text file
[~,file_name,~] = fileparts(scour_file);
writematrix(b_new2, ['point_output_' file_name '.txt']);
writematrix(b_rand1k,['point_output_' file_name '_1k.txt']);
writematrix(b_rand10k,['point_output_' file_name '_10k.txt']);
end
```

Appendix E. Python 3.6 Scripts

Tensorflow logger

```
# Code referenced from
https://gist.github.com/gyglim/1f8dfb1b5c82627ae3efcfbbadb9f514

import tensorflow as tf

import numpy as np

import scipy.misc

try:

    from StringIO import StringIO  # Python 2.7

except ImportError:

    from io import BytesIO         # Python 3.x

class Logger(object):

        def __init__(self, log_dir):

        """Create a summary writer logging to log_dir."""

        self.writer = tf.summary.FileWriter(log_dir)

    def scalar_summary(self, tag, value, step):

        """Log a scalar variable."""

        summary = tf.Summary(value=[tf.Summary.Value(tag=tag,
simple_value=value)])

        self.writer.add_summary(summary, step)

    def image_summary(self, tag, images, step):

        """Log a list of images."""

        img_summaries = []

        for i, img in enumerate(images):

            # Write the image to a string

            try:

                s = StringIO()

            except:

                s = BytesIO()

            scipy.misc.toimage(img).save(s, format="png")
            # Create an Image object
            img_sum = tf.Summary.Image(encoded_image_string=s.getvalue(),
                                   height=img.shape[0],
                                   width=img.shape[1])
            # Create a Summary value
            img_summaries.append(tf.Summary.Value(tag='%s/%d' % (tag, i),
image=img_sum))
        # Create and write Summary
        summary = tf.Summary(value=img_summaries)
        self.writer.add_summary(summary, step)
            def histo_summary(self, tag, values, step, bins=1000):
        """Log a histogram of the tensor of values."""
        # Create a histogram using numpy
        counts, bin_edges = np.histogram(values, bins=bins)
        # Fill the fields of the histogram proto
        hist = tf.HistogramProto()
        hist.min = float(np.min(values))
        hist.max = float(np.max(values))
        hist.num = int(np.prod(values.shape))
        hist.sum = float(np.sum(values))
        hist.sum_squares = float(np.sum(values**2))
        # Drop the start of the first bin
        bin_edges = bin_edges[1:]
        # Add bin edges and counts
        for edge in bin_edges:
            hist.bucket_limit.append(edge)
        for c in counts:
            hist.bucket.append(c)
        # Create and write Summary
        summary = tf.Summary(value=[tf.Summary.Value(tag=tag, histo=hist)])
```

```
self.writer.add_summary(summary, step)
self.writer.flush()
```

Auto-Encoder Testing

#Code referenced from https://github.com/yongheng1991/3D-point-capsule-networks

```
import argparse
import torch
import torch.nn.parallel
from torch.autograd import Variable
import torch.optim as optim
import sys
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../models')))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../dataloaders')))
import shapenet_part_loader
import shapenet_core13_loader
import shapenet_core55_loader
from pointcapsnet_ae import PointCapsNet
def main():
    USE_CUDA = True
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    capsule_net = PointCapsNet(opt.prim_caps_size, opt.prim_vec_size,
opt.latent_caps_size, opt.latent_caps_size, opt.num_points)
      if opt.model != '':
        capsule_net.load_state_dict(torch.load(opt.model))
    else:
     print ('pls set the model path')
         if USE_CUDA:
        print("Let's use", torch.cuda.device_count(), "GPUs!")
        capsule_net = torch.nn.DataParallel(capsule_net)
        capsule_net.to(device)

        if opt.dataset=='shapenet_part':

        test_dataset = shapenet_part_loader.PartDataset(classification=True,
npoints=opt.num_points, split='test')

        test_dataloader = torch.utils.data.DataLoader(test_dataset,
batch_size=opt.batch_size, shuffle=True, num_workers=4)

    elif opt.dataset=='shapenet_core13':

        test_dataset = shapenet_core13_loader.ShapeNet(normal=False,
npoints=opt.num_points, train=False)

        test_dataloader = torch.utils.data.DataLoader(test_dataset,
batch_size=opt.batch_size, shuffle=True, num_workers=4)

    elif opt.dataset=='shapenet_core55':

        test_dataset =
shapenet_core55_loader.Shapnet55Dataset(batch_size=opt.batch_size,npoints=op
t.num_points, shuffle=True, train=False)

# test process for 'shapenet_part' or 'shapenet_core13'

    capsule_net.eval()

    if 'test_dataloader' in locals().keys() :

        test_loss_sum = 0

        for batch_id, data in enumerate(test_dataloader):

            points, _= data

            if(points.size(0)<opt.batch_size):

                break

            points = Variable(points)

            points = points.transpose(2, 1)

            if USE_CUDA:

                points = points.cuda()

            latent_caps, reconstructions= capsule_net(points)

            test_loss = capsule_net.module.loss(points, reconstructions)

            test_loss_sum += test_loss.item()

            print('accumalate of batch %d loss is : %f' % (batch_id, test_loss.item()))

        test_loss_sum = test_loss_sum / float(len(test_dataloader))
        print('test loss is : %f' % (test_loss_sum))


# test process for 'shapenet_core55'
    else:
        test_loss_sum = 0
        while test_dataset.has_next_batch():
            batch_id, points_= test_dataset.next_batch()
            points = torch.from_numpy(points_)
            if(points.size(0)<opt.batch_size):
                break
            points = Variable(points)
            points = points.transpose(2, 1)
            if USE_CUDA:
                points = points.cuda()
            latent_caps, reconstructions= capsule_net(points)
            test_loss = capsule_net.module.loss(points, reconstructions)
            test_loss_sum += test_loss.item()
            print('accumalate of batch %d loss is : %f' % (batch_id, test_loss.item()))
        test_loss_sum = test_loss_sum / float(len(test_dataloader))
        print('test loss is : %f' % (test_loss_sum))
                        if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--batch_size', type=int, default=2, help='input batch size')
    parser.add_argument('--n_epochs', type=int, default=300, help='number of
epochs to train for')


    parser.add_argument('--prim_caps_size', type=int, default=1024, help='number
of primary point caps')
    parser.add_argument('--prim_vec_size', type=int, default=16, help='scale of
primary point caps')
```

```
    parser.add_argument('--latent_caps_size', type=int, default=64, help='number of latent caps')
    parser.add_argument('--latent_vec_size', type=int, default=64, help='scale of latent caps')
    parser.add_argument('--num_points', type=int, default=2048, help='input point set size')
    parser.add_argument('--model', type=str, default='tmp_checkpoints/shapenet_part_dataset__64caps_64vec_95.pth', help='model path')
    parser.add_argument('--dataset', type=str, default='shapenet_part', help='dataset: shapenet_part, shapenet_core13, shapenet_core55')
    opt = parser.parse_args()
    print(opt)
    main()
```

Auto-Encoder Training

```
import argparse
import torch
import torch.nn.parallel
from torch.autograd import Variable
import torch.optim as optim
import sys
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../models')))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../dataloaders')))
import shapenet_part_loader
import shapenet_core13_loader
import shapenet_core55_loader
from logger import Logger
from pointcapsnet_ae import PointCapsNet
def main():
   USE_CUDA = True
   device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
   capsule_net = PointCapsNet(opt.prim_caps_size, opt.prim_vec_size,
opt.latent_caps_size, opt.latent_caps_size, opt.num_points)
     if opt.model != '':
       capsule_net.load_state_dict(torch.load(opt.model))
    if USE_CUDA:
       print("Let's use", torch.cuda.device_count(), "GPUs!")
       capsule_net = torch.nn.DataParallel(capsule_net)
       capsule_net.to(device)
   #create folder to save logs
```

```
log_dir='./logs'+'/'+opt.dataset+'_dataset_'+str(opt.latent_caps_size)+'caps_'+str(o
pt.latent_vec_size)+'vec'+'_batch_size_'+str(opt.batch_size)
    if not os.path.exists(log_dir):
        os.makedirs(log_dir);
    logger = Logger(log_dir)
    #create folder to save trained models
    if not os.path.exists(opt.outf):
        os.makedirs(opt.outf);
        if opt.dataset=='shapenet_part':
        train_dataset = shapenet_part_loader.PartDataset(classification=True,
npoints=opt.num_points, split='train')
        train_dataloader = torch.utils.data.DataLoader(train_dataset,
batch_size=opt.batch_size, shuffle=True, num_workers=4)
    elif opt.dataset=='shapenet_core13':
        train_dataset = shapenet_core13_loader.ShapeNet(normal=False,
npoints=opt.num_points, train=True)
        train_dataloader = torch.utils.data.DataLoader(train_dataset,
batch_size=opt.batch_size, shuffle=True, num_workers=4)
    elif opt.dataset=='shapenet_core55':
        train_dataset =
shapenet_core55_loader.Shapnet55Dataset(batch_size=opt.batch_size,
npoints=opt.num_points, shuffle=True, train=True)
# training process for 'shapenet_part' or 'shapenet_core13'
    capsule_net.train()
    if 'train_dataloader' in locals().keys() :
        for epoch in range(opt.n_epochs):
            if epoch < 50:
                optimizer = optim.Adam(capsule_net.parameters(), lr=0.0001)
            elif epoch<150:
                optimizer = optim.Adam(capsule_net.parameters(), lr=0.00001)
            else:
                optimizer = optim.Adam(capsule_net.parameters(), lr=0.000001)
```

```
      train_loss_sum = 0
      for batch_id, data in enumerate(train_dataloader):
         points, _= data
         if(points.size(0)<opt.batch_size):
            break
         points = Variable(points)
         points = points.transpose(2, 1)
         if USE_CUDA:
            points = points.cuda()
            optimizer.zero_grad()
         latent_caps, reconstructions= capsule_net(points)
         train_loss = capsule_net.module.loss(points, reconstructions)
         train_loss.backward()
         optimizer.step()
         train_loss_sum += train_loss.item()

         info = {'train_loss': train_loss.item()}

         for tag, value in info.items():
            logger.scalar_summary(
               tag, value, (len(train_dataloader) * epoch) + batch_id + 1)
                    if batch_id % 50 == 0:
            print('bactch_no:%d/%d, train_loss: %f ' %  (batch_id,
len(train_dataloader), train_loss.item()))
         print('Average train loss of epoch %d : %f' %
           (epoch, (train_loss_sum / len(train_dataloader))))
         if epoch% 5 == 0:
         dict_name=opt.outf+'/'+opt.dataset+'_dataset_'+
'_'+str(opt.latent_caps_size)+'caps_'+str(opt.latent_caps_size)+'vec_'+str(epoch)+'
.pth'
         torch.save(capsule_net.module.state_dict(), dict_name)
```

```
# training process for 'shapenet_core55'
    else:
        for epoch in range(opt.n_epochs):
            if epoch < 20:
                optimizer = optim.Adam(capsule_net.parameters(), lr=0.001)
            elif epoch<50:
                optimizer = optim.Adam(capsule_net.parameters(), lr=0.0001)
            else:
                optimizer = optim.Adam(capsule_net.parameters(), lr=0.00001)
                    train_loss_sum = 0
            while train_dataset.has_next_batch():
                batch_id, points_= train_dataset.next_batch()
                points = torch.from_numpy(points_)
                if(points.size(0)<opt.batch_size):
                    break
                points = Variable(points)
                points = points.transpose(2, 1)
                if USE_CUDA:
                    points = points.cuda()
                optimizer.zero_grad()
                latent_caps, reconstructions= capsule_net(points)
                train_loss = capsule_net.module.loss(points, reconstructions)
                train_loss.backward()
                optimizer.step()
                train_loss_sum += train_loss.item()
                    info = {'train_loss': train_loss.item()}
                for tag, value in info.items():
                    logger.scalar_summary(
                        tag, value, (int(750/opt.batch_size) * epoch) + batch_id + 1)
                nfo = {'train_loss': train_loss.item()}
                for tag, value in info.items():
                    logger.scalar_summary(
                        tag, value, (int(750/opt.batch_size) * epoch) + batch_id + 1)
                            if batch_id % 50 == 0:
                    print('bactch_no:%d/%d at epoch %d train_loss: %f ' %  (batch_id,
int(750/opt.batch_size),epoch,train_loss.item() )) # the dataset size is 57448
            print('Average train loss of epoch %d : %f' % (epoch, (train_loss_sum /
int(750/opt.batch_size))))
            train_dataset.reset()
            if epoch% 5 == 0:
                dict_name=opt.outf+'/'+opt.dataset+'_dataset_'+
'_'+str(opt.latent_caps_size)+'caps_'+str(opt.latent_caps_size)+'vec_'+str(epoch)+'
.pth'
                torch.save(capsule_net.module.state_dict(), dict_name)
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--batch_size', type=int, default=2, help='input batch size')
    parser.add_argument('--n_epochs', type=int, default=2000, help='number of
epochs to train for')
    parser.add_argument('--prim_caps_size', type=int, default=1024, help='number
of primary point caps')
    parser.add_argument('--prim_vec_size', type=int, default=16, help='scale of
primary point caps')
    parser.add_argument('--latent_caps_size', type=int, default=64, help='number
of latent caps')
    parser.add_argument('--latent_vec_size', type=int, default=64, help='scale of
latent caps')
    parser.add_argument('--num_points', type=int, default=2048, help='input point
set size')
    parser.add_argument('--outf', type=str, default='tmp_checkpoints', help='output
folder')
    parser.add_argument('--model', type=str, default='', help='model path')
    parser.add_argument('--dataset', type=str, default='shapenet_part',
help='dataset: shapenet_part, shapenet_core13, shapenet_core55')
```

```
opt = parser.parse_args()
print(opt)
main()
```

Visualize Capsule Reconstruction

```
from open3d import *
import argparse
import torch
import torch.nn.parallel
from torch.autograd import Variable
import torch.optim as optim
import sys
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../models')))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../dataloaders')))
import shapenet_part_loader
import shapenet_core13_loader
import shapenet_core55_loader
from pointcapsnet_ae import PointCapsNet
def main():
        #create pcd object list to save the reconstructed patch per capsule
    pcd_list=[]
    for i in range(opt.latent_caps_size):
        pcd_ = open3d.geometry.PointCloud()
        pcd_list.append(pcd_)
    colors = plt.cm.tab20((np.arange(20)).astype(int))
    #random selected viz capsules
    hight_light_caps=[np.random.randint(0, opt.latent_caps_size) for r in
range(10)]
          USE_CUDA = True
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    capsule_net = PointCapsNet(opt.prim_caps_size, opt.prim_vec_size,
opt.latent_caps_size, opt.latent_caps_size, opt.num_points)
```

```
    if opt.model != '':
        capsule_net.load_state_dict(torch.load(opt.model))
    else:
        print ('pls set the model path')
            if USE_CUDA:
        print("Let's use", torch.cuda.device_count(), "GPUs!")
        capsule_net = torch.nn.DataParallel(capsule_net)
        capsule_net.to(device)
        if opt.dataset=='shapenet_part':
        test_dataset = shapenet_part_loader.PartDataset(classification=True,
npoints=opt.num_points, split='test')
        test_dataloader = torch.utils.data.DataLoader(test_dataset,
batch_size=opt.batch_size, shuffle=True, num_workers=4)
    elif opt.dataset=='shapenet_core13':
        test_dataset = shapenet_core13_loader.ShapeNet(normal=False,
npoints=opt.num_points, train=False)
        test_dataloader = torch.utils.data.DataLoader(test_dataset,
batch_size=opt.batch_size, shuffle=True, num_workers=4)
    elif opt.dataset=='shapenet_core55':
        test_dataset =
shapenet_core55_loader.Shapnet55Dataset(batch_size=opt.batch_size,npoints=op
t.num_points, shuffle=True, train=False)
    capsule_net.eval()
    if 'test_dataloader' in locals().keys() :
        test_loss_sum = 0
        for batch_id, data in enumerate(test_dataloader):
            points, _= data
            if(points.size(0)<opt.batch_size):
                break
            points = Variable(points)
            points = points.transpose(2, 1)
            if USE_CUDA:
                points = points.cuda()
            latent_caps, reconstructions= capsule_net(points)
                        for pointset_id in range(opt.batch_size):
                prc_r_all=reconstructions[pointset_id].transpose(1,
0).contiguous().data.cpu()
                prc_r_all_point=open3d.geometry.PointCloud()
                prc_r_all_point.points = open3d.utility.Vector3dVector(prc_r_all)
                colored_re_pointcloud= open3d.geometry.PointCloud()
                jc=0
                for j in range(opt.latent_caps_size):

current_patch=torch.zeros(int(opt.num_points/opt.latent_caps_size),3)
                    for m in range(int(opt.num_points/opt.latent_caps_size)):
                        current_patch[m,]=prc_r_all[opt.latent_caps_size*m+j,] # the
reconstructed patch of the capsule m is not saved continuesly in the output
reconstruction.
                    pcd_list[j].points = open3d.utility.Vector3dVector(current_patch)
                    if (j in hight_light_caps):
                        pcd_list[j].paint_uniform_color([colors[jc,0], colors[jc,1],
colors[jc,2]])
                        jc+=1
                    else:
                        pcd_list[j].paint_uniform_color([0.8,0.8,0.8])
                    colored_re_pointcloud+=pcd_list[j]
                open3d.visualization.draw_geometries([colored_re_pointcloud])
             # test process for 'shapenet_core55'
    else:
        test_loss_sum = 0
        while test_dataset.has_next_batch():
            batch_id, points_= test_dataset.next_batch()
            points = torch.from_numpy(points_)
            if(points.size(0)<opt.batch_size):
                break
            points = Variable(points)
            points = points.transpose(2, 1)
            if USE_CUDA:
                points = points.cuda()
            latent_caps, reconstructions= capsule_net(points)
            for pointset_id in range(opt.batch_size):
                prc_r_all=reconstructions[pointset_id].transpose(1,
0).contiguous().data.cpu()
                prc_r_all_point=open3d.geometry.PointCloud()
                prc_r_all_point.points = Vector3dVector(prc_r_all)
                colored_re_pointcloud= open3d.geometry.PointCloud()
                jc=0
                for j in range(opt.latent_caps_size):

current_patch=torch.zeros(int(opt.num_points/opt.latent_caps_size),3)
                    for m in range(int(opt.num_points/opt.latent_caps_size)):
                        current_patch[m,]=prc_r_all[opt.latent_caps_size*m+j,] # the
reconstructed patch of the capsule m is not saved continuesly in the output
reconstruction.
                    pcd_list[j].points = Vector3dVector(current_patch)
                    if (j in hight_light_caps):
                        pcd_list[j].paint_uniform_color([colors[jc,0], colors[jc,1],
colors[jc,2]])
                        jc+=1
                    else:
                        pcd_list[j].paint_uniform_color([0.8,0.8,0.8])
                    colored_re_pointcloud+=pcd_list[j]
        draw_geometries([colored_re_pointcloud])
if __name__ == "__main__":
    from open3d import *
    import matplotlib.pyplot as plt
    import numpy as np
    parser = argparse.ArgumentParser()
    parser.add_argument('--batch_size', type=int, default=2, help='input batch size')
    parser.add_argument('--n_epochs', type=int, default=100, help='number of
epochs to train for')
    parser.add_argument('--prim_caps_size', type=int, default=1024, help='number
of primary point caps')
    parser.add_argument('--prim_vec_size', type=int, default=16, help='scale of
primary point caps')
    parser.add_argument('--latent_caps_size', type=int, default=64, help='number
of latent caps')
    parser.add_argument('--latent_vec_size', type=int, default=64, help='scale of
latent caps')
    parser.add_argument('--num_points', type=int, default=2048, help='input point
set size')
    parser.add_argument('--model', type=str,
default='tmp_checkpoints/shapenet_part_dataset__64caps_64vec_95.pth',
help='model path')
    parser.add_argument('--dataset', type=str, default='shapenet_part',
help='dataset: shapenet_part, shapenet_core13, shapenet_core55')
    opt = parser.parse_args()
    print(opt)
    main()
```

Evaluate Segmentation

```
from open3d import *
import argparse
import torch
import torch.nn.parallel
from torch.autograd import Variable
import torch.optim as optim
import torch.nn.functional as F
import sys
import os
import numpy as np
import statistics
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../models')))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../dataloaders')))
import shapenet_part_loader
import matplotlib.pyplot as plt
from pointcapsnet_ae import PointCapsNet
from capsule_seg_net import CapsSegNet
#import h5py
from sklearn.svm import LinearSVC
import json
def main():
    blue = lambda x:'\033[94m' + x + '\033[0m'
    cat_no={'Scour':0}
    #generate part label one-hot correspondence from the catagory:
    dataset_main_path=os.path.abspath(os.path.join(BASE_DIR, '../../dataset'))
    oid2cpid_file_name=os.path.join(dataset_main_path, 
opt.dataset,'shapenetcore_partanno_segmentation_benchmark_v0/shapenet_part_
overallid_to_catid_partid.json')
```

```
    oid2cpid = json.load(open(oid2cpid_file_name, 'r'))
    object2setofoid = {}
    for idx in range(len(oid2cpid)):
        objid, pid = oid2cpid[idx]
        if not objid in object2setofoid.keys():
            object2setofoid[objid] = []
        object2setofoid[objid].append(idx)
        all_obj_cat_file = os.path.join(dataset_main_path, opt.dataset,
'shapenetcore_partanno_segmentation_benchmark_v0/synsetoffset2category.txt')
    fin = open(all_obj_cat_file, 'r')
    lines = [line.rstrip() for line in fin.readlines()]
    objcats = [line.split()[1] for line in lines]
#    objnames = [line.split()[0] for line in lines]
#    on2oid = {objcats[i]:i for i in range(len(objcats))}
    fin.close()
    colors = plt.cm.tab10((np.arange(10)).astype(int))
    blue = lambda x:'\033[94m' + x + '\033[0m'
# load the model for point cpas auto encoder
    capsule_net = PointCapsNet(opt.prim_caps_size, opt.prim_vec_size,
opt.latent_caps_size, opt.latent_vec_size, opt.num_points,)
    if opt.model != '':
        capsule_net.load_state_dict(torch.load(opt.model))
    if USE_CUDA:
        capsule_net = torch.nn.DataParallel(capsule_net).cuda()
    capsule_net=capsule_net.eval()
     # load the model for capsule wised part segmentation
    caps_seg_net = CapsSegNet(latent_caps_size=opt.latent_caps_size,
latent_vec_size=opt.latent_vec_size , num_classes=opt.n_classes)
    if opt.part_model != '':
        caps_seg_net.load_state_dict(torch.load(opt.part_model))
    if USE_CUDA:
        caps_seg_net = caps_seg_net.cuda()
    caps_seg_net = caps_seg_net.eval()
        train_dataset = shapenet_part_loader.PartDataset(classification=False,
class_choice=opt.class_choice, npoints=opt.num_points, split='test')
    train_dataloader = torch.utils.data.DataLoader(train_dataset,
batch_size=opt.batch_size, shuffle=False, num_workers=4)
    pcd_colored = open3d.geometry.PointCloud()
    pcd_ori_colored = open3d.geometry.PointCloud()
    rotation_angle=-np.pi/4
    cosval = np.cos(rotation_angle)
    sinval = np.sin(rotation_angle)
    flip_transforms  = [[cosval, 0, sinval,-1],[0, 1, 0,0],[-sinval, 0, cosval,0],[0, 0, 0,
1]]
    flip_transformt  = [[cosval, 0, sinval,1],[0, 1, 0,0],[-sinval, 0, cosval,0],[0, 0, 0,
1]]
            correct_sum=0
    for batch_id, data in enumerate(train_dataloader):
        points, part_label, cls_label= data
        if not (opt.class_choice==None ):
            cls_label[:]= cat_no[opt.class_choice]
            if(points.size(0)<opt.batch_size):
            break
                        # use the pre-trained AE to encode the point cloud into latent
capsules
        points_ = Variable(points)
        points_ = points_.transpose(2, 1)
        if USE_CUDA:
            points_ = points_.cuda()
        latent_caps, reconstructions= capsule_net(points_)
        reconstructions=reconstructions.transpose(1,2).data.cpu()
                        #concatanete the latent caps with one-hot part label
        cur_label_one_hot = np.zeros((opt.batch_size, 16), dtype=np.float32)
        for i in range(opt.batch_size):
            cur_label_one_hot[i, cls_label[i]] = 1
            iou_oids = object2setofoid[objcats[cls_label[i]]]
            for j in range(opt.num_points):
                part_label[i,j]=iou_oids[part_label[i,j]]
        cur_label_one_hot=torch.from_numpy(cur_label_one_hot).float()
        expand =cur_label_one_hot.unsqueeze(2).expand(opt.batch_size, 16,
opt.latent_caps_size).transpose(1,2)
        expand,latent_caps=Variable(expand),Variable(latent_caps)
        expand,latent_caps=expand.cuda(),latent_caps.cuda()
        latent_caps=torch.cat((latent_caps,expand),2)
              # predict the part class per capsule
        latent_caps=latent_caps.transpose(2, 1)
        output=caps_seg_net(latent_caps)
        for i in range (opt.batch_size):
            iou_oids = object2setofoid[objcats[cls_label[i]]]
            non_cat_labels = list(set(np.arange(2)).difference(set(iou_oids))) #
            mini = torch.min(output[i,:,:])
            output[i,:, non_cat_labels] = mini - 1000
        pred_choice = output.data.cpu().max(2)[1]
               # assign predicted the capsule part label to its reconstructed point patch

reconstructions_part_label=torch.zeros([opt.batch_size,opt.num_points],dtype=tor
ch.int64)
        for i in range(opt.batch_size):
            for j in range(opt.latent_caps_size):
                for m in range(int(opt.num_points/opt.latent_caps_size)):

reconstructions_part_label[i,opt.latent_caps_size*m+j]=pred_choice[i,j]
```

```
                # assign the part label from the reconstructed point cloud to the input
point set with NN
        pcd=pcd = open3d.geometry.PointCloud()

pred_ori_pointcloud_part_label=torch.zeros([opt.batch_size,opt.num_points],dtyp
e=torch.int64)
        for point_set_no in range (opt.batch_size):
            pcd.points =
open3d.utility.Vector3dVector(reconstructions[point_set_no,])
            pcd_tree = open3d.geometry.KDTreeFlann(pcd)
            for point_id in range (opt.num_points):
                [k, idx, _] =
pcd_tree.search_knn_vector_3d(points[point_set_no,point_id,:], 10)
                local_patch_labels=reconstructions_part_label[point_set_no,idx]

pred_ori_pointcloud_part_label[point_set_no,point_id]=statistics.median(local_p
atch_labels)
                       # calculate the accuracy with the GT
        correct =
pred_ori_pointcloud_part_label.eq(part_label.data.cpu()).cpu().sum()
        correct_sum=correct_sum+correct.item()
        print(' accuracy is: %f'
%(correct_sum/float(opt.batch_size*(batch_id+1)*opt.num_points)))


                        # viz the part segmentation
        point_color=torch.zeros([opt.batch_size,opt.num_points,3])
        point_ori_color=torch.zeros([opt.batch_size,opt.num_points,3])
        for point_set_no in range (opt.batch_size):
            iou_oids = object2setofoid[objcats[cls_label[point_set_no ]]]
            for point_id in range (opt.num_points):
                part_no=pred_ori_pointcloud_part_label[point_set_no,point_id]-
iou_oids[0]
                point_color[point_set_no,point_id,0]=colors[part_no,0]
                point_color[point_set_no,point_id,1]=colors[part_no,1]
                point_color[point_set_no,point_id,2]=colors[part_no,2]

            pcd_colored.points=open3d.utility.Vector3dVector(points[point_set_no,])

pcd_colored.colors=open3d.utility.Vector3dVector(point_color[point_set_no,])
                    for point_id in range (opt.num_points):
                part_no=part_label[point_set_no,point_id]-iou_oids[0]
                point_ori_color[point_set_no,point_id,0]=colors[part_no,0]
                point_ori_color[point_set_no,point_id,1]=colors[part_no,1]
                point_ori_color[point_set_no,point_id,2]=colors[part_no,2]


pcd_ori_colored.points=open3d.utility.Vector3dVector(points[point_set_no,])

pcd_ori_colored.colors=open3d.utility.Vector3dVector(point_ori_color[point_set
_no,])
                  pcd_ori_colored.transform(flip_transforms)# tansform the pcd in
order to viz both point cloud
            pcd_colored.transform(flip_transformt)
            open3d.visualization.draw_geometries([pcd_ori_colored, pcd_colored])
                  if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument('--batch_size', type=int, default=2, help='input batch size')
    parser.add_argument('--prim_caps_size', type=int, default=1024, help='number
of primary point caps')
    parser.add_argument('--prim_vec_size', type=int, default=16, help='scale of
primary point caps')
    parser.add_argument('--latent_caps_size', type=int, default=64, help='number
of latent caps')
    parser.add_argument('--latent_vec_size', type=int, default=64, help='scale of
latent caps')
```

```
    parser.add_argument('--num_points', type=int, default=2048, help='input point
set size')

    parser.add_argument('--part_model', type=str,
default='./tmp_checkpoints/64caps_64vec_100%
of_training_data_at_epoch100.pth', help='model path for the pre-trained part
segmentation network')

    parser.add_argument('--model', type=str,
default='../AE/tmp_checkpoints/shapenet_part_dataset__64caps_64vec_95.pth',
help='model path')

    parser.add_argument('--dataset', type=str, default='shapenet_part',
help='dataset: shapenet_part, shapenet_core13, shapenet_core55, modelent40')

    parser.add_argument('--n_classes', type=int, default=2, help='part classes in all
the catagories')

    parser.add_argument('--class_choice', type=str, default='Scour', help='choose
the class to eva')

    opt = parser.parse_args()

    print(opt)

    USE_CUDA = True

    main()
```

Save Results

```
from open3d import *
import argparse
import torch
import torch.nn.parallel
from torch.autograd import Variable
import torch.optim as optim
import sys
import os
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../models')))
sys.path.append(os.path.abspath(os.path.join(BASE_DIR, '../../dataloaders')))
import shapenet_part_loader
from pointcapsnet_ae import PointCapsNet
def main():
    USE_CUDA = True
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
    capsule_net = PointCapsNet(opt.prim_caps_size, opt.prim_vec_size,
opt.latent_caps_size, opt.latent_caps_size, opt.num_points)
     if opt.model != '':
       capsule_net.load_state_dict(torch.load(opt.model))
     if USE_CUDA:
       print("Let's use", torch.cuda.device_count(), "GPUs!")
       capsule_net = torch.nn.DataParallel(capsule_net)
       capsule_net.to(device)
       if opt.dataset=='shapenet_part':
       if opt.save_training:
           split='train'
       else :
           split='test'
```

```
        dataset = shapenet_part_loader.PartDataset(classification=False,
npoints=opt.num_points, split=split)

        dataloader = torch.utils.data.DataLoader(dataset, batch_size=opt.batch_size,
shuffle=True, num_workers=4)

        # init saving process

    pcd = open3d.geometry.PointCloud()

    data_size=0

    dataset_main_path=os.path.abspath(os.path.join(BASE_DIR, '../../dataset'))

    out_file_path=os.path.join(dataset_main_path, opt.dataset,'latent_caps')

    if not os.path.exists(out_file_path):

        os.makedirs(out_file_path);

    if opt.save_training:

        out_file_name=out_file_path+"/saved_train_with_part_label.h5"

    else:

        out_file_name=out_file_path+"/saved_test_with_part_label.h5"

    if os.path.exists(out_file_name):

        os.remove(out_file_name)

    fw = h5py.File(out_file_name, 'w', libver='latest')

    dset = fw.create_dataset("data",
(1,opt.latent_caps_size,opt.latent_vec_size,),maxshape=(None,opt.latent_caps_siz
e,opt.latent_vec_size), dtype='<f4')

    dset_s =
fw.create_dataset("part_label",(1,opt.latent_caps_size,),maxshape=(None,opt.late
nt_caps_size,),dtype='uint8')

    dset_c = fw.create_dataset("cls_label",(1,),maxshape=(None,),dtype='uint8')

    fw.swmr_mode = True

#   process for 'shapenet_part' or 'shapenet_core13'

    capsule_net.eval()

        for batch_id, data in enumerate(dataloader):

        #print(list(enumerate(dataloader)))

        points, part_label, cls_label= data

        if(points.size(0)<opt.batch_size):
            break
        points = Variable(points)
        points = points.transpose(2, 1)
        if USE_CUDA:
            points = points.cuda()
        latent_caps, reconstructions= capsule_net(points)
              # For each resonstructed point, find the nearest point in the input
pointset,
        # use their part label to annotate the resonstructed point,
        # Then after checking which capsule reconstructed this point, use the part
label to annotate this capsule
        reconstructions=reconstructions.transpose(1,2).data.cpu()
        points=points.transpose(1,2).data.cpu()
        cap_part_count=torch.zeros([opt.batch_size, opt.latent_caps_size,
opt.n_classes],dtype=torch.int64)
        for batch_no in range (points.size(0)):
            pcd.points = open3d.utility.Vector3dVector(points[batch_no,])
            pcd_tree = open3d.geometry.KDTreeFlann(pcd)
            for point_id in range (opt.num_points):
                [k, idx, _] =
pcd_tree.search_knn_vector_3d(reconstructions[batch_no,point_id,:], 1)
                point_part_label=part_label[batch_no, idx]
                caps_no=point_id % opt.latent_caps_size
                #print(batch_no)
                #print(caps_no)
                #print(point_part_label.numpy())
                cap_part_count[batch_no,caps_no,point_part_label]+=1
        _,cap_part_label=torch.max(cap_part_count,2) # if the reconstucted points
have multiple part labels, use the majority as the capsule part label
             # write the output latent caps and cls into file
        data_size=data_size+points.size(0)
        new_shape = (data_size,opt.latent_caps_size,opt.latent_vec_size, )
        dset.resize(new_shape)
        dset_s.resize((data_size,opt.latent_caps_size,))
        dset_c.resize((data_size,))
                latent_caps_=latent_caps.cpu().detach().numpy()
        target_=cap_part_label.numpy()
        dset[data_size-points.size(0):data_size,:,:] = latent_caps_
        dset_s[data_size-points.size(0):data_size] = target_
        dset_c[data_size-points.size(0):data_size] = cls_label.squeeze().numpy()
            dset.flush()
        dset_s.flush()
        dset_c.flush()
        print('accumalate of batch %d, and datasize is %d ' % ((batch_id),
(dset.shape[0])))
               fw.close()
             if __name__ == "__main__":
    import h5py
    parser = argparse.ArgumentParser()
    parser.add_argument('--batch_size', type=int, default=2, help='input batch size')
    parser.add_argument('--n_epochs', type=int, default=100, help='number of
epochs to train for')
    parser.add_argument('--prim_caps_size', type=int, default=1024, help='number
of primary point caps')
    parser.add_argument('--prim_vec_size', type=int, default=16, help='scale of
primary point caps')
    parser.add_argument('--latent_caps_size', type=int, default=64, help='number
of latent caps')
    parser.add_argument('--latent_vec_size', type=int, default=64, help='scale of
latent caps')
    parser.add_argument('--num_points', type=int, default=2048, help='input point
set size')
```

```
    parser.add_argument('--model', type=str,
default='../AE/tmp_checkpoints/shapenet_part_dataset__64caps_64vec_95.pth',
help='model path')

    parser.add_argument('--dataset', type=str, default='shapenet_part', help='It has
to be shapenet part')

#    parser.add_argument('--save_training', type=bool, default=True, help='save
the output latent caps of training data or test data')

    parser.add_argument('--save_training', help='save the output latent caps of
training data or test data', action='store_true')

    parser.add_argument('--n_classes', type=int, default=2, help='catagories of
current dataset')

    opt = parser.parse_args()

    print(opt)

    main()
```

ShapeNet Loader

```
#from __future__ import print_function
import torch.utils.data as data
import os
import os.path
import torch
import json
import numpy as np
import sys
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
dataset_path=os.path.abspath(os.path.join(BASE_DIR,
'../dataset/shapenet_part/shapenetcore_partanno_segmentation_benchmark_v0/'))


class PartDataset(data.Dataset):
    def __init__(self, root=dataset_path, npoints=2048, classification=False,
class_choice=None, split='train', normalize=True):
        self.npoints = npoints
        self.root = root
        self.catfile = os.path.join(self.root, 'synsetoffset2category.txt')
        self.cat = {}
        self.classification = classification
        self.normalize = normalize
        with open(self.catfile, 'r') as f:
            for line in f:
                ls = line.strip().split()
                self.cat[ls[0]] = ls[1]
        # print(self.cat)
        if not class_choice is None:
            self.cat = {k: v for k, v in self.cat.items() if k in class_choice}
            print(self.cat)
        self.meta = {}
        with open(os.path.join(self.root, 'train_test_split',
'shuffled_train_file_list.json'), 'r') as f:
            train_ids = set([str(d.split('/')[2]) for d in json.load(f)])
        with open(os.path.join(self.root, 'train_test_split',
'shuffled_val_file_list.json'), 'r') as f:
            val_ids = set([str(d.split('/')[2]) for d in json.load(f)])
        with open(os.path.join(self.root, 'train_test_split',
'shuffled_test_file_list.json'), 'r') as f:
            test_ids = set([str(d.split('/')[2]) for d in json.load(f)])
        for item in self.cat:
            # print('category', item)
            self.meta[item] = []
            dir_point = os.path.join(self.root, self.cat[item], 'points')
            dir_seg = os.path.join(self.root, self.cat[item], 'points_label')
            # print(dir_point, dir_seg)
            fns = sorted(os.listdir(dir_point))
            if split == 'trainval':
                fns = [fn for fn in fns if ((fn[0:-4] in train_ids) or (fn[0:-4] in val_ids))]
            elif split == 'train':
                fns = [fn for fn in fns if fn[0:-4] in train_ids]
            elif split == 'val':
                fns = [fn for fn in fns if fn[0:-4] in val_ids]
            elif split == 'test':
                fns = [fn for fn in fns if fn[0:-4] in test_ids]
            else:
                print('Unknown split: %s. Exiting..' % (split))
                exit(-1)
            for fn in fns:
                token = (os.path.splitext(os.path.basename(fn))[0])
                self.meta[item].append((os.path.join(dir_point, token + '.txt'),
os.path.join(dir_seg, token + '.txt'),self.cat[item], token))
        self.datapath = []
        for item in self.cat:
            for fn in self.meta[item]:
                self.datapath.append((item, fn[0], fn[1], fn[2], fn[3]))
        self.classes = dict(zip(sorted(self.cat), range(len(self.cat))))
        print(self.classes)
        self.num_seg_classes = 0
        if not self.classification:
            for i in range(len(self.datapath)//50):
                l = len(np.unique(np.loadtxt(self.datapath[i][2]).astype(np.uint8)))
                if l > self.num_seg_classes:
                    self.num_seg_classes = l
        # print(self.num_seg_classes)
        self.cache = {}  # from index to (point_set, cls, seg) tuple
        self.cache_size = 18000
    def __getitem__(self, index):
        if index in self.cache:
#              point_set, seg, cls= self.cache[index]
            point_set, seg, cls, foldername, filename = self.cache[index]
        else:
            fn = self.datapath[index]
            cls = self.classes[self.datapath[index][0]]
#              cls = np.array([cls]).astype(np.int32)
            point_set = np.loadtxt(fn[1]).astype(np.float32)
            if self.normalize:
                point_set = self.pc_normalize(point_set)
            seg = np.loadtxt(fn[2]).astype(np.int64) - 1
            foldername = fn[3]
            filename = fn[4]
            if len(self.cache) < self.cache_size:
                self.cache[index] = (point_set, seg, cls, foldername, filename)
        #print(point_set.shape, seg.shape)
        choice = np.random.choice(len(seg), self.npoints, replace=True)
        # resample
        point_set = point_set[choice, :]
        seg = seg[choice]

        # To Pytorch
        point_set = torch.from_numpy(point_set)
        seg = torch.from_numpy(seg)
        cls = torch.from_numpy(np.array([cls]).astype(np.int64))
        if self.classification:
            return point_set, cls
        else:
            return point_set, seg , cls
            def __len__(self):
        return len(self.datapath)

    def pc_normalize(self, pc):
        """ pc: NxC, return NxC """
        l = pc.shape[0]
        centroid = np.mean(pc, axis=0)
        pc = pc - centroid
        m = np.max(np.sqrt(np.sum(pc**2, axis=1)))
        pc = pc / m
        return pc
if __name__ == '__main__':
    print('test')
```

```
d = PartDataset(
root='../dataset/shapenetcore_partanno_segmentation_benchmark_v0/',classification=True, class_choice='Scour', npoints=2048, split='test')

ps, cls = d[0]

print(ps.size(), ps.type(), cls.size(), cls.type())
```