SELF-SIMILAR SPIN IMAGES FOR POINT CLOUD MATCHING

by

Daniel Pulido

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computational Sciences and Informatics

Committee:

_____	Dr. Anthony Stefanidis, Dissertation Director
_____	Dr. Estela Blaisten-Barojas, Committee Member
_____	Dr. Arie Croitoru, Committee Member
_____	Dr. Juan Cebral, Committee Member
_____	Dr. Jason Kinser, Acting Chair, Department of Computational and Data Sciences
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
_____	Dr. Peggy Agouris, Dean, College of Science
Date: _____	Fall Semester 2017 George Mason University Fairfax, VA

Self-Similar Spin Images for Point Cloud Matching

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Daniel Pulido
Master of Science
Worcester Polytechnic Institute, 2003
Bachelor of Science
Boston University, 1998

Director: Dr. Anthony Stefanidis, Professor
Department of School of Physics, Astronomy, and Computational Sciences

Fall Semester 2017
George Mason University
Fairfax, VA

Dedication

This dissertation is dedicated to my parents Alfonso and Ines Pulido, my wife Caroline, my sister Claudia, brother-in-law Ted, and nephew Aidan ... also Kali, Jackie, Sushi, and Leia.

Acknowledgments

My special thanks to my dissertation director Dr. Anthony Stefanidis for providing helpful insights during the proposal and development of this work. He provided both the freedom to pursue my own interests while ensuring the research stayed focused. His editorial comments on this dissertation and the related publication provided incredibly helpful in keeping both works on track.

My additional thanks go to the members of the committee: Dr. Estela Blaisten for taking on the duties of the committee chair and keeping this dissertation work organized; Dr. Juan Cebral and Dr. Arie Croitoru for their reviews and insightful comments during the proposal and defense phase.

Table of Contents

Page

List of Tables vii
List of Figures viii
List of Algorithms xiii
Abstract xiv
1 Introduction 1
1.1 Motivation and Problem Statement 1
1.2 Statement of Research Objectives 4
1.3 Hypothesis 5
1.4 Intended Audience 5
1.5 Organization of the Thesis 5
2 Background and Related Works 6
2.1 Overview 6
2.2 Single Image-to-Point Cloud Registration 6
2.3 Point Cloud-to-Point Cloud Registration 11
2.3.1 Feature Detection 13
2.3.2 Feature Matching 15
2.3.3 Registration 16
2.4 Scale Matching 17
2.5 Change Detection 20
3 Local Self-Similarity based Matching 23
3.1 Overview 23
3.2 Spin Images 24
3.3 Self-Similar Spin Images 26
3.4 Self-Similar Keyscale 28
3.5 Point Cloud Registration 32
4 Point Cloud Change Detection 35
4.1 Overview of Change Detection 35
4.2 Nearest Neighbor Order Statistics 35

5 Experiments 40
5.1 Test Data 41
5.2 Scale Matching 42
5.3 Feature Matching 46
5.4 Change Detection 54
6 Conclusions and Future Work 57
6.1 Overview 57
6.2 Scale Matching 58
6.3 Feature Matching 59
6.4 Change Detection 60
6.5 Future Work 60
A Robustness Test Set 61
B Scale Matching Results 64
C Feature Matching Results 68
D Change Detection Results 76
E Publication 79
Bibliography 87

List of Tables

Table		Page
2.1	List of feature detectors, their properties, and reference document.	14
5.1	Break down of experiment algorithms tested	40
5.2	Model point clouds for testing feature and scale matching.	41
5.3	Lidar point clouds for testing feature and scale matching.	41
5.4	Point clouds for testing change detection.	44
5.5	List of transformations applied to test point clouds	44
5.6	Results of performing scale matching on the robustness dataset.	48
5.7	Transformation parameters estimated using the baseline and proposed methods. The proposed method correctly estimated the RST parameters. The baseline method did not.	50
5.8	Transformation parameters estimated using the baseline and proposed methods. The proposed method provides a better estimate for the RST parameters than does the baseline method.	51
5.9	Results of performing feature matching on the robustness dataset.	51

List of Figures

Figure		Page
1.1	Fusion of Point Cloud and Web 2.0 data.	2
2.1	Alignment of Point Cloud to Overhead Image. © 2009.	7
2.2	Lines extracted from Lidar and Imagery are processed to form 2D orthogonal corners then matched. © 2008.	7
2.3	Alignment of SfM to high resolution DSM shows detailed features like steel wires perfectly aligned. © 2011.	8
2.4	Overview of SfM to GIS registration: (a) unordered Internet photos (b) SfM reconstruction of scene (c) geographic data such as Google Earth 3D models. (d) Alignment of SfM point cloud with geographic data. © 2013.	9
2.5	Alignment of Aerial Image to Lidar Depth Image. © 2011.	10
2.6	Self-similarity based matching of figure sketches with images of people striking poses similar to those in the sketch figures. © 2007.	11
2.7	Descriptor creation process. (a) Image based matching computes a local correlation surface at detected feature points and then converts it to a log-polar representation. (b) Video based matching similar to the image case except the correlation and descriptors are volumes. © 2007.	12
2.8	An example of MoPC feature extraction. The size and color of each feature point indicates the various scales at which the feature points were detected. © 2012.	15
2.9	Feature matching for a human point cloud using local self-similarity. © 2012.	16
2.10	Point Cloud coarse adjustment based on extracted feature points. © 2012.	17
2.11	Coarse rotation adjustment estimated using EGI. Despite the little overlap between the two halves of the point cloud the method was able to align them. © 2006.	18
2.12	Matching low and high resolution images with scale differences. © 2004.	19

2.13 Matching point clouds using conventional descriptor like spin images, which is clearly susceptible to false matches, versus the the more robust LD-SIFT method. © 2012 20
2.14 Point cloud scale matching using their keyscales - which is the optimal spin image bin width at the minimum of the their PCA contribution rate curves. © 2010. 21
2.15 Example of detecting of changes in a target (c) image against a database of reference images, (a) and (b) using a probabilistic inference model that searches for regions in a test image that can not be composed from regions of reference images. © 2005. 22
2.16 Example change detection between a reference (a) and target (b) point cloud using a distance metric. The highlighted changes (c) spotlight the two missing buildings. 22
3.1 Self-similarity surface of normals. The brighter a point is, the more similar its normal is to the normal at the center point. © 2012. 23
3.2 Descriptor local coordinate system and quantization 26
3.3 Example Self-Similar Spin Image descriptor displayed as six radial slices. . . 28
3.4 Example of point clouds with extracted feature points (a) and (b) and their keyscales computed from their PCA scores minimum (c) and (d). The ratio of their keyscales is 10 which correctly matches their relative scale. 30
3.5 Example of self-similar keyscales computed from their PCA scores maximum (a) and (b). The ratio of their keyscales is 10 which correctly matches their relative scale. 32
4.1 Buildings in reference that are missing in target are highlighted red. 36
4.2 The histogram thresholded (in red) is computed using the triangle method. Right mode corresponds to missing buildings and left mode is the persistent background. Two clusters of changes are detected between reference and target point clouds using min-cut segmentation. 37
5.1 Model point clouds. 42
5.2 Lidar point clouds. 43
5.3 Lidar change detection point clouds. 45
5.4 The scale difference between the Ant point cloud and its scaled version was detected by both the baseline and proposed method. The ratio of keyscales for both the baseline and proposed methods yields 10. 46

5.5 The scale difference between the Ant point cloud and its scaled version was detected by the proposed method but not the baseline method. The ratio of the keyscale for proposed method curves yields the correct scale difference of 10. The baseline method did not have any local minimums and therefore did not yield a keyscale. 47
5.6 The feature points identified using the MoPC metric for the point cloud in (a) and its transformed version (b). 48
5.7 The Beethoven point clouds matched using the feature points for the baseline method (a) and proposed method (b). 49
5.8 The feature points identified using the MoPC metric for the point cloud in (a) and its transformed version (b). 49
5.9 The PC-SMALL point clouds matched using the feature points for the baseline method (a) and proposed method (b). 50
5.10 Example Lidar (PC-LARGE) being reduced in height for a range of scales. 52
5.11 Minimum, average, and maximum alignment error over all 25 Lidar datasets for the proposed method over a range of height-only reductions. The errors were stable and trivially small up to 75% and began to grow at 90%. 53
5.12 Minimum, average, and maximum alignment error over all 25 Lidar datasets for the proposed method over a range of height reductions combined with the RST transformation. The errors were stable up to 75% and began to grow at 90%. 54
5.13 2D projections of the reference and first target point clouds. There are two missing building in the upper left of the target point cloud. 55
5.14 Clusters found for each target point cloud's outliers using the graph min-cut algorithm. The extra cluster found in (c) and (d) is from a sharp change in altitude from the building. 56
A.1 Set of Lidar point clouds used for robustness testing. The data sets were selected from an urban area in Denver that covered a variety of building shapes, heights, as well as flat terrains. 63
B.1 The scale difference between the Beethoven point cloud and its scaled version was detected by the baseline method and approximately by the proposed method. The ratio of keyscale for the baseline method yielded 10 and for the proposed method was 9.41. 64

B.2 The scale difference between the Chopper point cloud and its scaled version was detected by both the baseline and proposed methods. The ratio of keyscales for both yields 10. 65
B.3 The scale difference between the Turbine point cloud and its scaled version was detected by the baseline method and approximately by the proposed method. The ratio of keyscales for the baseline method yielded 10 and the proposed method was 9.41. 66
B.4 The scale difference between the PC #4 point cloud and its scaled version was detected only by the proposed method which yielded a ratio of keyscales of 2. The baseline method did not have any local minimums. 67
C.1 Feature points detected using MoPC for Ant (a) and its RST version (b). . 68
C.2 Initial matched features are shown for Ant (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated. 69
C.3 Feature points detected using MoPC for Chopper (a) and its RST version (b). 70
C.4 Initial matched features are shown for Chopper (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated. 71
C.5 Feature points detected using MoPC for Turbine (a) and its RST version (b). 72
C.6 Initial matched features are shown for Turbine (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated. 73
C.7 Feature points detected using MoPC for PC-LARGE (a) and its RST version (b). 74
C.8 Initial matched features are shown for PC-LARGE (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated. 75
D.1 2D projections of the PC-TARGET-2 point cloud, the expected changes, and the detected clusters of outliers. There are three missing buildings along the left side of the point cloud. 76
D.2 2D projections of the PC-TARGET-3 point cloud, the expected changes, and the detected clusters of outliers. There are four missing buildings along the left and bottom side of the point cloud. 77

D.3 2D projections of the PC-TARGET-4 point cloud, the expected changes, and the detected clusters of outliers. There are five missing buildings along the left, bottom, and upper right side of the point cloud. 78

List of Algorithms

3.1 Spin Image 25
3.2 Self-Similar Spin Image 27
3.3 Keyscale 29
3.4 Self-Similar Keyscale 31
3.5 Self-Similarity based Registration 33
4.1 Nearest Neighbor Order Statistics Change Detection 39

Abstract

SELF-SIMILAR SPIN IMAGES FOR POINT CLOUD MATCHING

Daniel Pulido, PhD

George Mason University, 2017

Dissertation Director: Dr. Anthony Stefanidis

The rapid growth of Light Detection And Ranging (Lidar) technologies that collect, process, and disseminate 3D point clouds have allowed for increasingly accurate spatial modeling and analysis of the real world. Lidar sensors can generate massive 3D point clouds of a collection area that provide highly detailed spatial and radiometric information. However, a Lidar collection can be expensive and time consuming. Simultaneously, the growth of crowdsourced Web 2.0 data (e.g., Flickr, OpenStreetMap) have provided researchers with a wealth of freely available data sources that cover a variety of geographic areas. Crowdsourced data can be of varying quality and density. In addition, since it is typically not collected as part of a dedicated experiment but rather volunteered, when and where the data is collected is arbitrary. The integration of these two sources of geoinformation can provide researchers the ability to generate products and derive intelligence that mitigate their respective disadvantages and combine their advantages.

Therefore, this research will address the problem of fusing two point clouds from potentially different sources. Specifically, we will consider two problems: scale matching and feature matching. Scale matching consists of computing feature metrics of each point cloud and analyzing their distributions to determine scale differences. Feature matching consists of defining local descriptors that are invariant to common dataset distortions (e.g., rotation

and translation). Additionally, after matching the point clouds they can be registered and processed further (e.g., change detection).

The objective of this research is to develop novel methods to fuse and enhance two point clouds from potentially disparate sources (e.g., Lidar and crowdsourced Web 2.0 datasets). The scope of this research is to investigate both scale and feature matching between two point clouds. The specific focus of this research will be in developing a novel local descriptor based on the concept of self-similarity to aid in the scale and feature matching steps.

An open problem in fusion is how best to extract features from two point clouds and then perform feature-based matching. The proposed approach for this matching step is the use of local self-similarity as an invariant measure to match features. In particular, the proposed approach is to combine the concept of local self-similarity with a well-known feature descriptor, Spin Images, and thereby define "**Self-Similar Spin Images**". This approach is then extended to the case of matching two points clouds in very different coordinate systems (e.g., a geo-referenced Lidar point cloud and stereo-image derived point cloud without geo-referencing). The use of Self-Similar Spin Images is again applied to address this problem by introducing a "**Self-Similar Keyscale**" that matches the spatial scales of two point clouds.

Another open problem is how best to detect changes in content between two point clouds. A method is proposed to find changes between two point clouds by analyzing the order statistics of the nearest neighbors between the two clouds, and thereby define the "**Nearest Neighbor Order Statistic**" method. Note that the well-known Hausdorff distance is a special case as being just the maximum order statistic. Therefore, by studying the entire histogram of these nearest neighbors it is expected to yield a more robust method to detect points that are present in one cloud but not the other. This approach is applied at multiple resolutions. Therefore, changes detected at the coarsest level will yield large missing targets and at finer levels will yield smaller targets.

Chapter 1: Introduction

1.1 Motivation and Problem Statement

The fusion of multi-source data sets is a problem of fundamental importance in the geospatial sciences. Its central role in data processing is due to a number of reasons.

1. The rapid growth of available data from a variety of new technologies provides researchers with a wealth of new intelligence to exploit.

2. The fusion of disparate data sets is also a necessity since no single sensor's data can capture all of the relevant characteristics of a given scene.

3. The varying availability of data sources requires combining those that are available.

4. Fusing data sets is often an *a priori* design decision and not just *a posteriori* necessity. For instance, the fusion of satellite images via pansharpening is part of the design of sensors like QuickBird and IKONOS. Otherwise, developing high resolution multispectral sensors would be cost and mission prohibited.

Two areas that have experienced considerable growth recently in the amount of data available to the public are 3D Lidar point clouds and crowdsourced Web 2.0 data. The former is usually the product of a collection organized by a government, commercial, or academic entity. The latter is usually the result of volunteers or amateurs posting data to an online site for the purposes of solving a specific problem (e.g., OpenStreetMap) or for entertainment (e.g., Flickr). The fusion of these two data sources can help solve numerous scientific problems. As with all fusion problems, each data source comes with its own advantages and disadvantages. The goal of data fusion is to mitigate their respective disadvantages.

Lidar data commonly comes in the form of a 3D point cloud. It has a number of advantages including high resolution, spatial accuracy, and point density. However, it has a number of disadvantages such as having limited availability in some areas, it requires specialized and expensive equipment, and planning and conducting a collection can be time consuming.

Alternatively, crowdsourced Web 2.0 data comes in a variety of formats such as images, GIS, or text. It also has a number of advantages. For instance, for a user it is generally free. In addition, it is becoming increasingly available. With much of this data being collected and delivered with mobile devices, online crowdsourced content is often updated at a more frequent pace. As a result, it lends itself to extraction of the newer content via change detection and fusing the new content to other sources (e.g., out-dated or legacy Lidar point clouds). However, as a consequence of crowdsourced data being provided by volunteers and amateurs it can be inaccurate. Also, by virtue of being volunteered it may not always be reliable in both quality and availability.

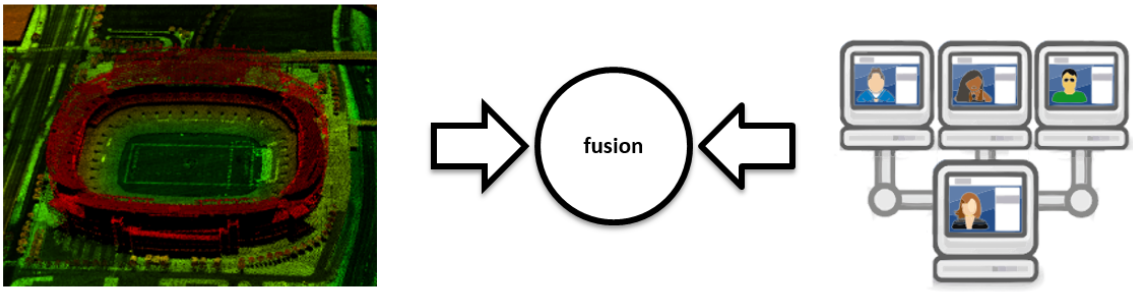


Figure 1.1: Fusion of Point Cloud and Web 2.0 data.

These two disparate sources are ideal candidates for data fusion. Combining the precision and accuracy of Lidar data with the ubiquity of crowdsourced imagery can provide a fused product with the best of both sources. However, the fusion of these two sources provides a number of technical challenges. The following summarize the challenges of fusing

Lidar 3D point clouds and unstructured images:

- **Differing modalities.** Lidar point clouds and imagery are two vastly different modalities. Selecting an appropriate local descriptor to detect corresponding features between a set of unstructured 3D spatial points and a correspond 2D raster image is a difficult task that can greatly impact any registration or fusion tasks.

- **Scale Mismatch.** Crowdsourced imagery often have limited associated geoinformation. If no geographic information is available then fusing will require aligning two data sets at very different scales. One source will be in a geographic coordinate system (the Lidar point cloud) and the other can be in a relative coordinate system (the imagery).

- **Scene Changes.** Differing collection times between the Lidar and imagery can lead to changes in scene content. While this provides an opportunity for extracting new content it also introduces errors during registration.

- **Point Density.** Lidar point clouds and stereo-image derived point clouds can have very different spatial point densities. This can have profound effects on the local descriptors used to characterize features.

- **Computational Load.** A single Lidar point cloud can be very large consisting of millions to billions of points. Therefore, processing a file that large can present a significant computational load.

- **User Interaction.** Many approaches that address the above issues often require some user interaction. It is desirable to limit user intervention in the fusion process as this interaction will often be dependent on the user's training and experience.

These challenges are still unresolved and active areas of research. This dissertation is motivated by finding robust techniques to address the first three of these technical challenges. The issues of computational load and user interaction are not addressed in this work.

Using a self-similarity based metric that is less sensitive to modality changes will address the differing modalities challenge presented. The development of a Self-Similar Keyscale scale matching approach will address the scale mismatch challenge presented. Finally, the proposed Nearest Neighbor Order Statistics change detection algorithm will address the scene change challenge discussed above.

1.2 Statement of Research Objectives

This research will develop novel methods to robustly and automatically detect scale differences between two point clouds and match their corresponding features in order to spatially register them. To mitigate the various sources of misalignment errors between point clouds - such as differing modalities, scale mismatch, and scene changes - this dissertation focuses on exploiting the property of local self-similarity in developing its matching feature descriptor. Therefore, this work has the following research objectives:

1. Develop a feature-driven, point cloud registration approach using a local self-similarity based local descriptor referred to as "**Self-Similar Spin Images**". This approach combines the robustness of local self-similarity based metrics with the descriptive power of spin images.

2. Develop a feature-driven, point cloud scale matching approach that exploits Self-Similar Spin Images by introducing the "**Self-Similar Keyscale**". This approach combines the robustness of local self-similarity based metrics introduced by using Self-Similar Spin Images with the automated exploratory power of the Principal Components based method of keyscales to detect changes in scale between two 3D point clouds.

3. Develop an approach to detect changes between two point clouds by introducing the method of "**Nearest Neighbor Order Statistics**". This approach combines the robustness of a dataset's order statistics with the power of graph based clustering methods.

1.3 Hypothesis

The statement of research objectives described in the previous section leads to the following hypothesis for this dissertation:

> *By combining the invariant properties of local self-similarity based metrics with the descriptive power of spin images and incorporating them into a single feature descriptor, this will lead to a robust spatial and scale matching process.*

This approach can lead to an improved state-of-the-art in point cloud matching while mitigating many of the sources of point cloud misalignment.

1.4 Intended Audience

This intended audience of this dissertation are researchers from the geospatial, remote sensing, geographic information systems, and computer vision communities. In addition, the results obtained from this research will be of particular value to scientists interested in 3D point cloud matching and change detection.

1.5 Organization of the Thesis

This thesis is organized as follows. Chapter 2 presents a technical background to the concepts presented in this dissertation as well as a review of the relevant related work in the scientific literature. Chapter 3 introduces the concept of local self-similarity and its application to point cloud registration and scale matching. Chapter 4 presents the detection of changes between two point clouds. Chapter 5 provides a summary of the experiments performed to validate the proposed algorithms and their results. Chapter 6 presents the conclusions drawn from the performed experiments and a summary of potential future areas of research to build upon this work. Finally, the appendices contain the details of all the experimental results obtained.

Chapter 2: Background and Related Works

2.1 Overview

This section summarizes the relevant research in the related areas addressed in this dissertation. The problem of geospatial data fusion is a varied and challenging area. Fusing data sources requires taking two or more products that are, usually, in very different formats and come from very different sensors. Consequently, before combining them requires some preprocessing that aligns the two data into a common frame. This frame varies from problem to problem but it can be a common coordinate system, spectral space, or some other abstract reference frame from which the data's primitive elements can be compared. The relevant scientific research reviewed for this dissertation is divided into the three main problems addressed - point cloud registration, point cloud scale matching, and change detection.

2.2 Single Image-to-Point Cloud Registration

A precursor to registering a 3D point cloud to another 3D point cloud is the problem of registering a single 2D image to a 3D Lidar point cloud [1–5]. The problem usually consists of projecting the 2D raster pixels and the 3D Lidar points into a common reference frame and then extracting and identifying matching features. These matching features define a registration transformation between the two data sets. For instance, registration of a point cloud to overhead imagery was demonstrated by [6] as depicted in Figure 2.1. In this approach the authors compute the optimal alignment using an objective function that matches 3D points to image edges and imposes free space constraints based on the visibility of points in each camera.

Similarly, in [7] aerial imagery was registered with untextured Lidar. Their approach

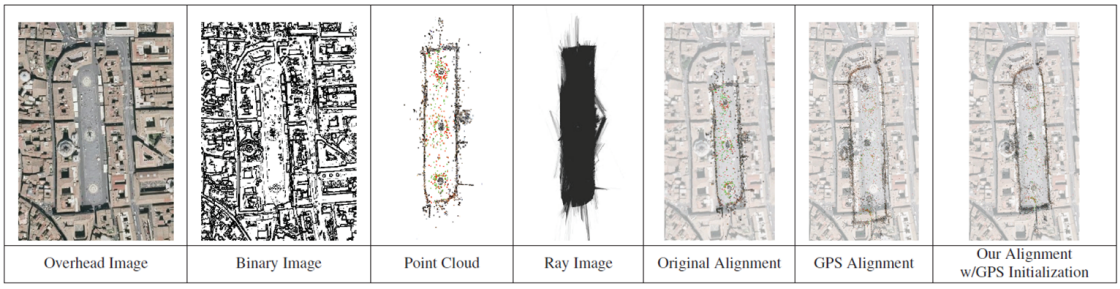
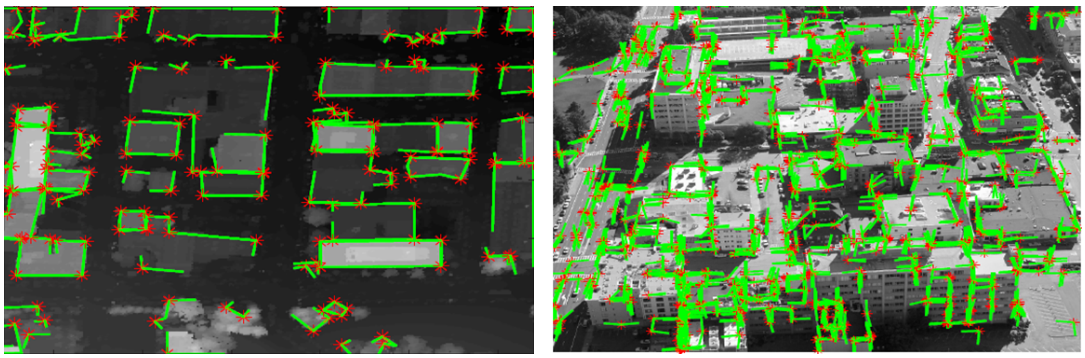


Figure 2.1: Alignment of Point Cloud to Overhead Image. © 2009.

involved starting with a course registration using Global Positioning System (GPS) parameters and the vanishing points of vertical lines to estimate camera properties. Then their registration is refined (see Figure 2.2) by finding 2D orthogonal corners from lines extracted in the Aerial Image and DSM. These corners are matched and a registration refinement is computed.

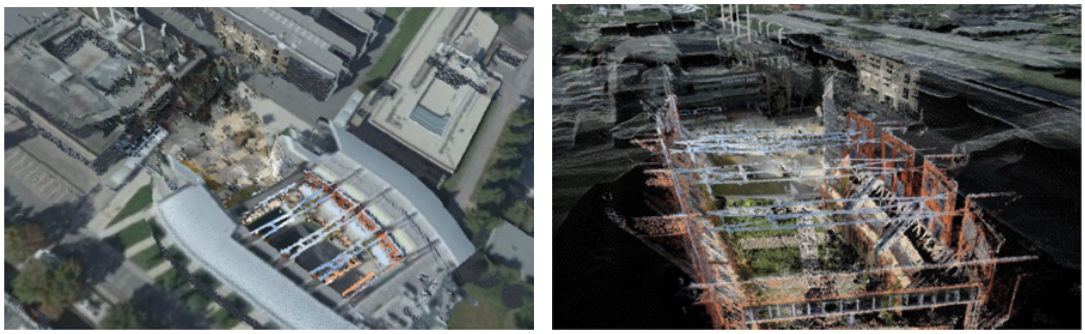


(a) Extracted lines from Lidar DSM (b) Extracted lines from Aerial Image

Figure 2.2: Lines extracted from Lidar and Imagery are processed to form 2D orthogonal corners then matched. © 2008.

In a related problem, the authors in [8] detailed an approach to perform automatic alignment of a point cloud to a Digital Surface Model (DSM) raster image (see Figure 2.3).

Their approach involved three steps where, first, GPS information of the DSM is used to generate an approximate geo-referencing. Then, a course registration is performed between the SfM model and the DSM by exploiting available GPS information. Lastly, a refined registration is performed by projecting the point cloud points onto the DSM and computing a best height map alignment using a normalized cross-correlation.



(a) Overlay of SfM model to DSM (b) Oblique view of SfM model to DSM overlay

Figure 2.3: Alignment of SfM to high resolution DSM shows detailed features like steel wires perfectly aligned. © 2011.

Accurate point cloud registration using geographic data was presented in [9] and depicted in Figure 2.4. This approach required matching data from very different modalities - a 3D point cloud derived from a Lidar sensor and a vector data. Their approach involved building a SfM point cloud from unstructured Internet photos (e.g., Flickr geotagged images); then estimating the gravity vector of the scene using vanishing points; estimating the current position, heading, and scale using geotags; then they perform a fine registration using ICP with the SfM point cloud and nearby Google Earth 3D buildings or Google Street View images.

The authors in [10] presented another method to register a panoramic sequence of images to a point cloud. Their approach involved integrating the SfM bundle adjustment with an ICP registration by sequentially performing ICP on portions of the SfM trajectories then updating camera properties using a bundle adjustment. This improved system performs an

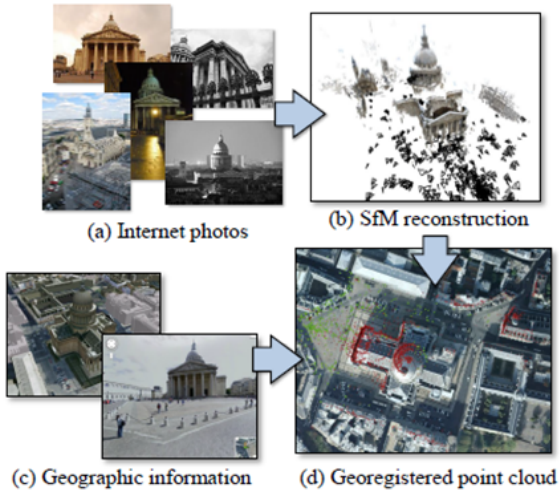


Figure 2.4: Overview of SfM to GIS registration: (a) unordered Internet photos (b) SfM reconstruction of scene (c) geographic data such as Google Earth 3D models. (d) Alignment of SfM point cloud with geographic data. © 2013.

accurate coarse registration and then SIFT interest point matches then can easily be added to update and refine the registration.

The above works focus on extracting feature components such as lines, corners, and facets and then matching them to similar components in the target point cloud. Using primitives like lines, corners, and facets has the drawback that they are ubiquitous elements found through out many scenes and matching them can because ambiguous and lead to very sensitive algorithms. In addition, they all assume either exact or approximate knowledge of the dataset scale differences. It is for this reason that a more robust method that analyzes local self-similarity patterns in data sets that can match complex objects with simple representations of the same objects was chosen for this research.

While the concept of self-similarity is prominent in fractal theory, and has been applied to computing spatial dimensionality and radar sensor designs, its application to image and point cloud matching is relatively recent. The use of self-similarity has been exploited as a feature descriptor in a number of applications such as imagery and video registration [11], detecting deformable shapes [12], and interest point detection [13]. The authors in [14]

exploit self-similarity based feature descriptors to match multi-modal images as shown in Figure 2.5.

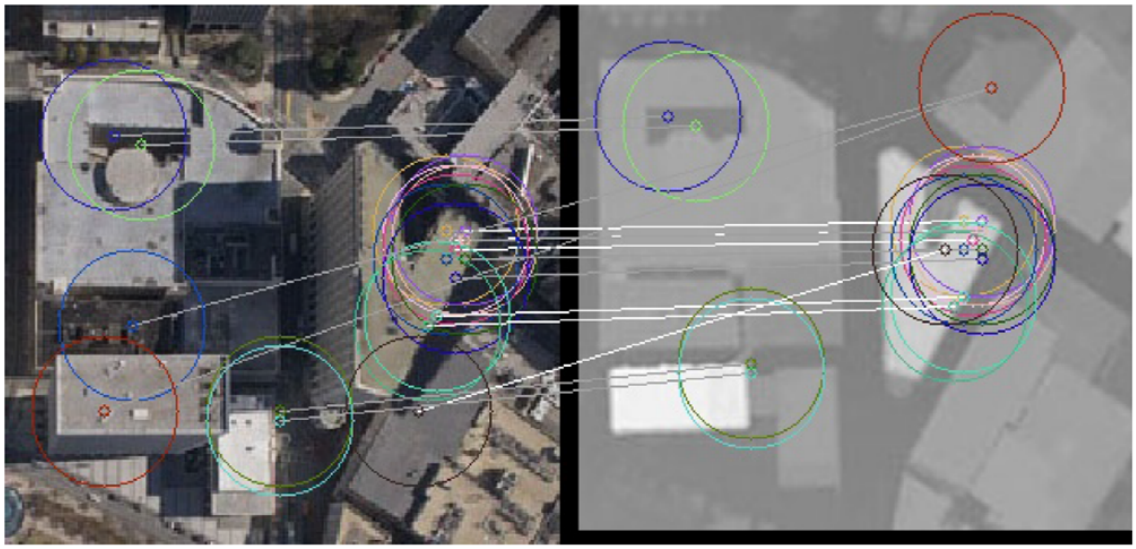


Figure 2.5: Alignment of Aerial Image to Lidar Depth Image. © 2011.

The notion of local self-similarity is based on the idea of identifying local geometric patterns across data sources regardless of modality. Finding similar features across data sets is described in [11] as follows, "*the local internal layouts of self-similarities are shared by these images, even though the patterns generating those self-similarities are not shared by those images*". That is, by examining how the structure near a feature is self-similar, that self-similarity will also appear in similar features in other data sources. To demonstrate the technique, the authors were able to match objects in an image (e.g. a human figure) to hand-drawn templates (e.g., a stick figure drawing of a human pose) as shown in Figure 2.6.

Specifically, the local self-similarity is computed as a correlation surface using a particular local descriptor (e.g., pixel intensity) for a given neighborhood of pixels. This correlation surface is then transformed into a binned log-polar representation to account for local spatial affine deformations as shown in Figure 2.7. This descriptor is constructed in a manner

Figure 2.6: Self-similarity based matching of figure sketches with images of people striking poses similar to those in the sketch figures. © 2007.

to ensure it provides a compact representation, mitigates against spatial distortions as well as local non-rigid deformations.

The application of local self-similarity to match features in images is used to register a 2D image to a Lidar point cloud. The approach projects the Lidar point cloud onto the image plane to generate a range depth map. Features are then extracted and matched from the 2D image and the corresponding range depth map using the Local Self-Similarity approach described in [15] where the similarity function measures similarity in the pixel values and their spatial gradients whereas previous applications ignored spatial gradients to match features. Therefore, this approach assumes that the image to be registered is geo-referenced but its geoinformation needs to be adjusted.

2.3 Point Cloud-to-Point Cloud Registration

To better understand the process of matching and registering two point clouds, and better organize the existing literature contributions, this process is broken down into the following basic steps.

1. **Feature Detection.** Due to the large number of points in most Lidar point clouds computing any local descriptor to perform feature matching will be computationally prohibitive. Therefore, feature detection is performed so that descriptors are only

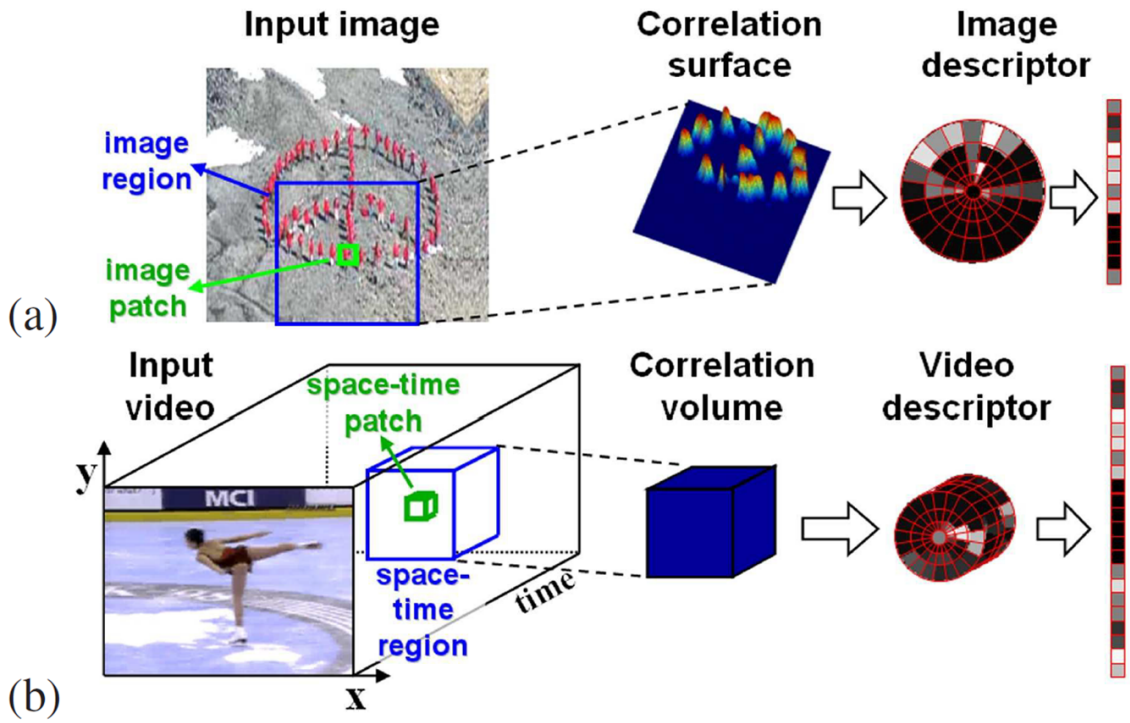


Figure 2.7: Descriptor creation process. (a) Image based matching computes a local correlation surface at detected feature points and then converts it to a log-polar representation. (b) Video based matching similar to the image case except the correlation and descriptors are volumes. © 2007.

computed at a subset of points.

2. **Scale Matching.** The extracted features are then used to determine the characteristic scale of each point cloud and then the scale of the target point cloud is scaled to match the reference point cloud. Matching the scale between the two point clouds is addressed in Section 2.4.

3. **Feature Matching.** The extracted features are then matched by computing local descriptors and comparing them to find corresponding points between the two point clouds. This step assumes that the feature descriptor are at the same scale and therefore comparable.

4. **Registration.** Once corresponding features have been identified, the two point clouds can be registered using the matched features to compute a coarse registration followed by a fine registration.

Each of these steps are themselves challenging areas of research, however, only the **Scale Matching** and **Feature Matching** aspects will be addressed in this proposed research. The relevant related work in the literature of all of these areas is discussed in the following sections.

2.3.1 Feature Detection

Matching point clouds can be computationally infeasible if the matching is performed for every point in the cloud since even modestly sized point clouds can contain millions of points, and often contain billions. Point clouds must typically be preprocessed to identify points that represent unique features in the scene. Therefore, a feature detector is needed to identify these feature points. This has two advantages. One, the computational load in computing local descriptors is decreased. Two, processing points that are not deemed to be unique likely do not contribute additional descriptor information. For instance, points on the corner of a building are typically identified as feature points by most feature detectors because they identify the geometric extents of the underlying building. However, points on a flat surface of a building will likely provide redundant information that add little descriptive information - as all the nearby points would have similar descriptors. Therefore, selecting a robust and computationally efficient feature extractor is crucial in pre-processing a point cloud. For example, extraction of local features in a dataset has a wide range of applications including stereo image matching [16], [17]; object recognition [18], [19]; texture recognition [20], [21]; image retrieval [21], [22]; and category recognition [23], [24], [25], [26].

Table 2.1 provides a list of the feature detectors reviewed and a brief summary of their properties and reference document. The descriptors can be broadly divided into three categories 1) differential operator based descriptors (Harris points, Harris-Laplace regions, Hessian-Laplace regions, Harris-Affine regions, Hessian-Affine regions, Harris-3D) which are

contrasted in [27], 2) Kernel based descriptors (Heat Kernel) which are compared in [28], and 3) Mesh based descriptors (Salient Points, Mesh-DoG, Mesh SIFT, and Mesh-Scale DoG) which are tested in [29].

Table 2.1: List of feature detectors, their properties, and reference document.

Descriptor	**Property**	**Reference**
Maximum of Principle Curvature	Rotation and scale invariance	[15]
Harris points	Rotation invariance	[30]
Harris-Laplace regions	Rotation and scale invariance	[21]
Hessian-Laplace regions	Rotation and scale invariance	[19]
Harris-Affine regions	Affine transformation invariance	[31]
Hessian-Affine regions	Affine transformation invariance	[32]
Harris-3D	Rotation and scale invariance	[33]
Heat Kernel	Isometric transformation invariance	[34]
Salient Point	Perceptually-inspired	[35]
Mesh-DoG	Rotation and scale invariance	[36]
Mesh-Scale DoG	Rotation and scale invariance	[36]
Mesh SIFT	Rotation and scale invariance	[37]

The feature detector selected for this dissertation is the "Maximum of Principle Curvature" (MoPC) method as it provides a robust and stable locator of interest regions as described by [15]. Figure 2.8 provides an example of feature points extracted. The example shows many of the locally flat areas not being extracted and only extracting points where considerable curvature occur. The MoPC method generates a minimum and maximum scale based on a characteristic size of the point cloud (i.e., its diameter). Intermediate scales are computed such that the ratio between sequential scales are constant. Then, a point in the cloud is a considered a feature point if its local principal curvature is a local maxima within a local neighborhood in space and adjacent scales.

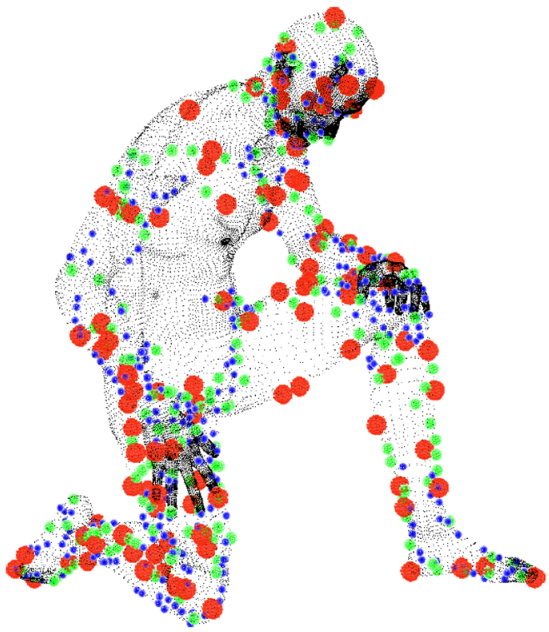
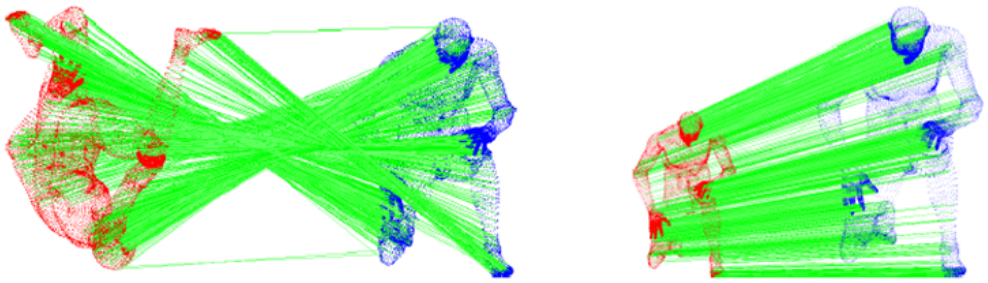


Figure 2.8: An example of MoPC feature extraction. The size and color of each feature point indicates the various scales at which the feature points were detected. © 2012.

2.3.2 Feature Matching

Local self-similarity has recently been applied as a local feature descriptor for 3D point cloud matching by Huang [15]. Their approach exploits the local self-similarity of geometric properties of point clouds, principally the local surface normal. Figure 2.9 shows an example of matching features of a human point cloud subjected to a rotation and scale transformation. The process of building the self-similarity surface and matching the features is similar to the process used in this dissertation and described in Section 3.3 and Section 3.5 except the authors in [15] used the local normal as the metric used to build the correlation surface.

In addition, the authors include other point cloud attributes such as intensity in defining their self-similarity descriptor. This approach is adopted and extended in this research by incorporating a more descriptive point cloud descriptor, Spin Images [38], into the local self-similarity framework.



(a) Matching with a rotation transformation. (b) Matching with a scale transformation.

Figure 2.9: Feature matching for a human point cloud using local self-similarity. © 2012.

2.3.3 Registration

While point cloud registration is not specifically researched in this dissertation, there are many aspects in choosing a registration method that can greatly affect the estimated solution. Often, registration techniques have two phases - a coarse adjustment followed by a refinement. The coarse adjustment can follow a standard approach, as in [39], where feature points are extracted and matched between two points and then the coordinates of the matched points are used within a data fitting process to estimate the parameters of some model (e.g., affine transformation) as shown in Figure 2.10.

Alternatively, a global approach can be used where local features are not extracted and instead a global model is used to estimate parameters of some spatial model. For instance, in [40] the authors exploited a global representation of range scans known as Extended Gaussian Images (EGI) to find the global rotation between two point clouds. Figure 2.11 shows an example of globally aligning two rabbit point clouds that have very little overlap.

The standard approach to a refinement registering of two point clouds is the Iterative Closest Point (ICP) method [41–46]. This approach seeks to iteratively minimize the distance between nearest neighbors of two point clouds. There are many variations on the basic ICP algorithm. The basic difference between these different implementations is what primitive feature to use when computing nearest neighbors (i.e., the point cloud points,

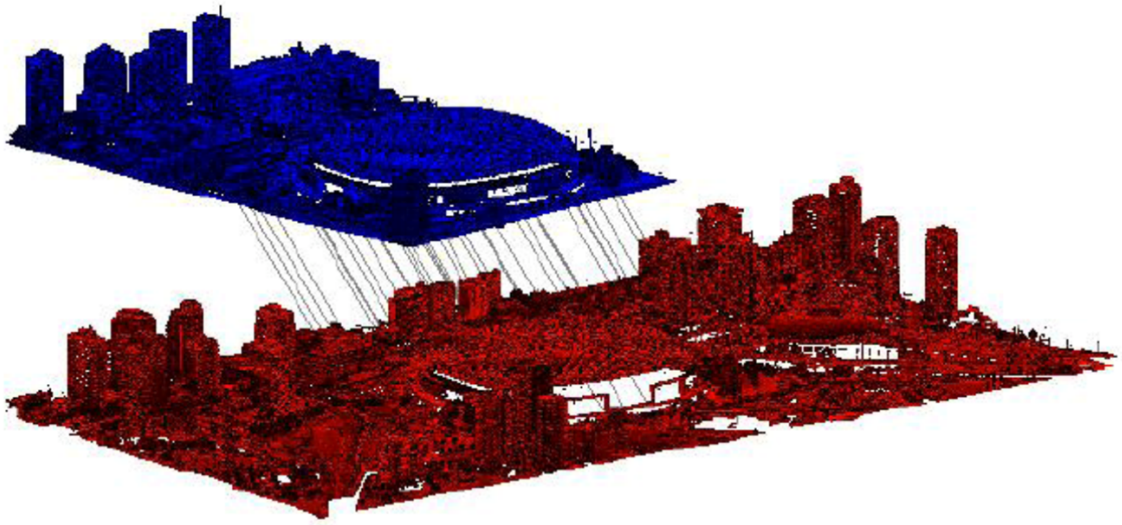


Figure 2.10: Point Cloud coarse adjustment based on extracted feature points. © 2012.

planes, or some other object). The ICP technique has also been extended to incorporate scale invariance in [47].

However, the solution provided by ICP can frequently correspond to a local minimum if the initial coarse adjustment is not sufficiently close to the true solution. As such, in this dissertation only a coarse adjustment is performed by computing the affine transformation parameters based on the matched features and the refinement step is not considered. This was done because the goal of this research is to evaluate the effectiveness of the matching process and not the complexities introduced by the refinement.

2.4 Scale Matching

A central processing task before two datasets are matched is to ensure that they are comparable. In the context of 3D point clouds this usually means that the two point clouds are spatially at the same scale. The process of scale matching usually takes one of two forms, either the scale is detected and one of the datasets are scaled to match the other or features

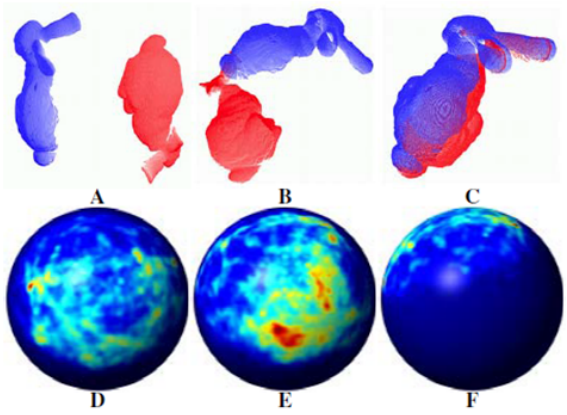


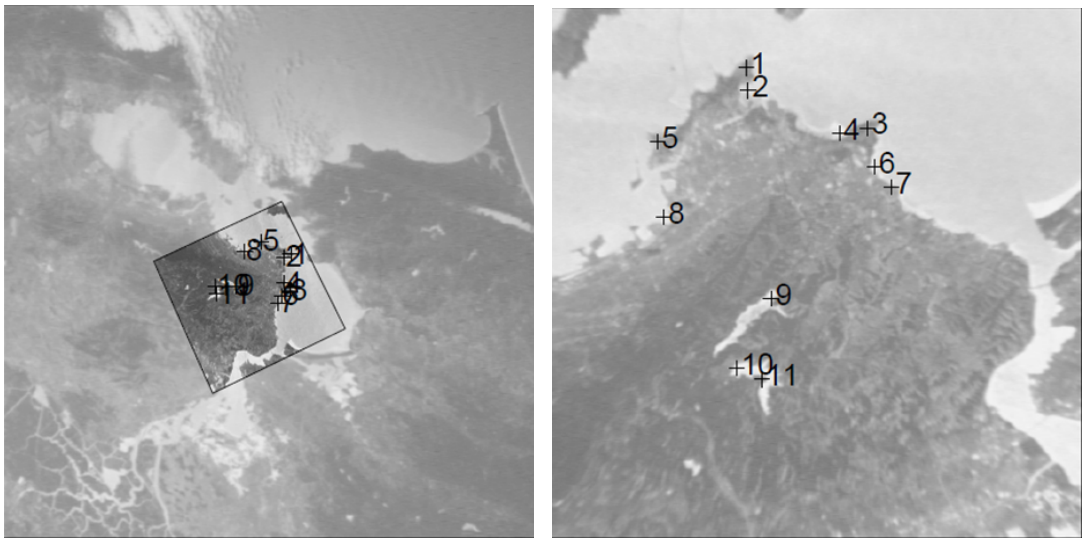
Figure 2.11: Coarse rotation adjustment estimated using EGI. Despite the little overlap between the two halves of the point cloud the method was able to align them. © 2006.

are extracted from the datasets in a scale-invariant manner.

In the 2D case the author in [48] examines the source images in scale space to correct for scale variations and matching the images by analyzing their scale pyramids. In [49] the author uses a scale-space representations of interest points and their descriptors to match and adjust the images to the same scale. This method relied on generating a scale representation of the high-resolution image and performing a one-to-many image matching solution. Figure 2.12 shows an example of the feature points identified between a low resolution to high resolution image using this method.

Another 2D approach used in [50] is to detect lines in a scale invariant manner by detecting them as local extrema in the scale-space while the author in [51] developed the well-known SIFT method that detects local features in a scale-invariant manner.

For the 3D case of matching point clouds similar approaches have been suggested. In [52] a 3D extension of the SIFT descriptor was proposed known as LD-SIFT. Figure 2.13 provides an example of matching a point cloud with this descriptor as compared to the standard Spin Image descriptor and clearly shows the susceptibility of the Spin Image descriptor to have false matches versus the more robust LD-SIFT approach. The LD-SIFT

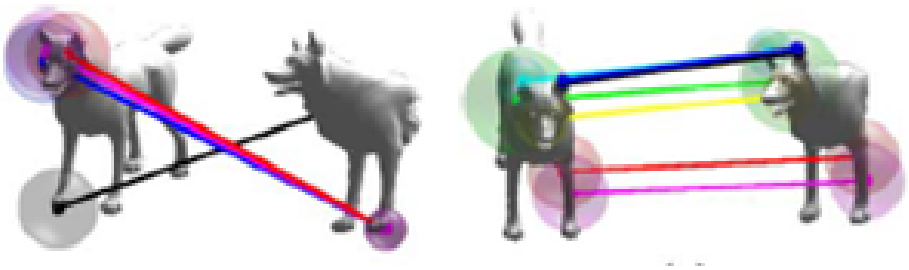


(a) Low resolution image (b) High resolution image

Figure 2.12: Matching low and high resolution images with scale differences. © 2004.

descriptor "*utilizes the DoG detector ... to detect the interest vertices and estimate the local scale, whereas the descriptor is computed by representing the vicinity of the interest vertex ... as a depth map.*" Similarly, in [53] a rotational projection statistics (RoPS) multi-scale representation of features is used to perform object recognition in a scale invariant manner.

A different approach is to directly detect the scales between two point clouds. In [54] the authors compute the *keyscale* of a point cloud that is an optimal scale that best captures point cloud feature descriptors as shown in Figure 2.14. In this approach the authors exploit the fact that for spin images with small bin widths all objects look like a plane and for large bin widths all objects look like a point. Therefore, there must be an optimal bin width size which is its keyscale. It is then hypothesized that the ratio between two point clouds' keyscales is the scale mismatch between them. Similarly, the authors in [47] indirectly use the keyscale concept by directly incorporating it into the standard ICP registration algorithm. It is this keyscale method that is adopted and extended in this dissertation.



(a) Matching point cloud with spin images (b) Matching point cloud with LD-SIFT

Figure 2.13: Matching point clouds using conventional descriptor like spin images, which is clearly susceptible to false matches, versus the the more robust LD-SIFT method. © 2012

2.5 Change Detection

While co-registering two datasets is itself a challenging task, the goal of co-registering is usually to then perform some subsequent task, such as fusion, to combine the corresponding data points and their attributes in some manner. For example, fusing oblique imagery with augmented aerial Lidar was a presented in [55]. Fusing is not limited to combining data sources from a single sensor modality or capture time. The authors in [56] address the problem of multi-modal and multi-temporal data fusion. One precursor to fusion is often change detection. That is, in order to inject one data source with information from another one must first identify those parts of the data set that are new. The authors in [57] provide a method to detect changes in images and video by using a probabilistic inference model that searches for regions in a test image that can not be composed from regions of reference images. Figure 2.15 provides an example of a region that cannot be composed from the reference image regions and is therefore flagged as unlikely.

Change detection in a 3D environment can also be performed from 2D imagery as detailed in [58]. Performing change detection between two point clouds can be done in a variety of manners but most approach the problem as depicted in Figure 2.16 where there is a reference point cloud, a target point cloud, and a distance or similarity metric is computed

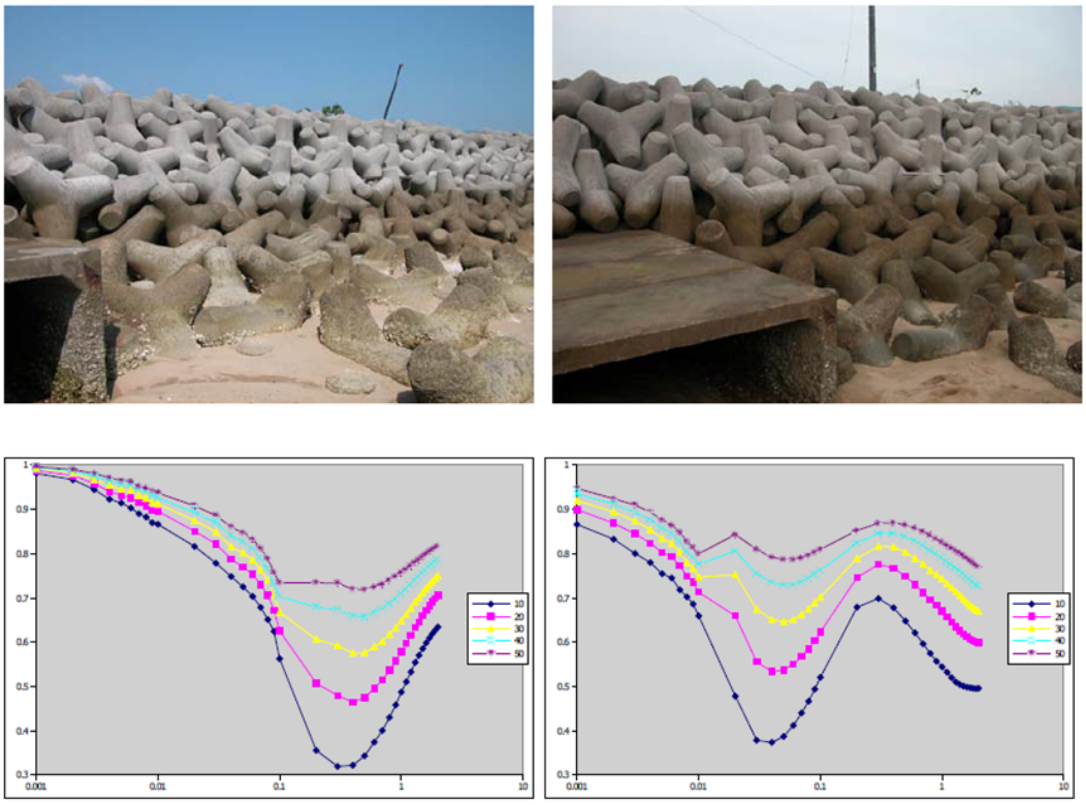


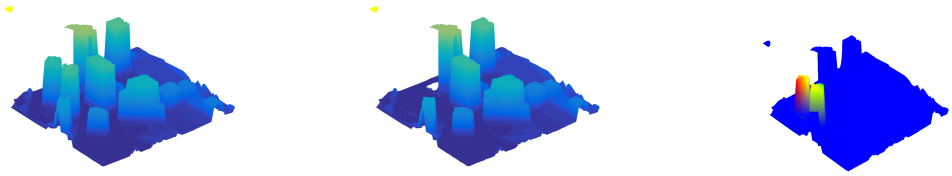
Figure 2.14: Point cloud scale matching using their keyscales - which is the optimal spin image bin width at the minimum of the their PCA contribution rate curves. © 2010.

between them that highlights points in the target that are farthest from any point in the reference.

For example, detecting changes from a ground Lidar sensor is presented in [59]. This approach seeks to create a spatial representation of the point cloud and then computes simple distance metrics through this representation. In particular, the authors use a octree representation of the input points clouds and then compute simple distance metrics between the points in corresponding octree cells. One of the distance metrics the authors use is the Hausdorff distance which computes the largest distance between any two points between the two point clouds. The approach to change detection utilized in this dissertation, as

(a) Person walking. (b) Person running. (c) Person crawling.

Figure 2.15: Example of detecting of changes in a target (c) image against a database of reference images, (a) and (b) using a probabilistic inference model that searches for regions in a test image that can not be composed from regions of reference images. © 2005.



(a) Reference (b) Target (c) Highlighted changes

Figure 2.16: Example change detection between a reference (a) and target (b) point cloud using a distance metric. The highlighted changes (c) spotlight the two missing buildings.

described in Chapter 4, is similar to this approach but exploits a more robust method to highlight changes and then extends it to actually merge the detected changed points and clusters them into objects.

Chapter 3: Local Self-Similarity based Matching

3.1 Overview

The local self-similarity approach was extended from 2D images to 3D point clouds in [15] by selecting a simple geometric descriptor, the surface normal, to generate the needed correlation surfaces for a given neighborhood of points. The authors also incorporated multiple descriptors in building the self-similarity correlation surface such as curvature and intensity. Figure 3.1 depicts a sample point cloud with a local self-similarity surface of its normals [15].

Figure 3.1: Self-similarity surface of normals. The brighter a point is, the more similar its normal is to the normal at the center point. © 2012.

In this dissertation, the application of local self-similarity to 3D point features is used to match their respective point clouds in both scale and space and then register the clouds. The proposed approach extracts Self-Similar Spin Image descriptors, as defined in 3.3 and

presented in [60], at feature locations for both points clouds. The extracted Self-Similar Spin Images are then analyzed using a PCA analysis to compute the Self-Similar Keyscale metric, as described in Section 3.4 and presented in [60], of each point cloud in order to determine their relative scale differences. Once the two point clouds are scale matched, Self-Similar Spin Images that are deemed to be matches define a transformation between the two point clouds. This transformation is used to register them, as defined in 3.5. This research will only consider affine transformations.

3.2 Spin Images

A central issue in computer vision how to recognize 3D objects. This problem is complicated by the fact that 3D objects are often unstructured (e.g., point clouds). That is, unlike other datasets like 2D raster imagery there is no inherent structure to exploit. This problem is addressed by Spin Images [38] which provide a local representation of an object. Spin Images have proven to be a powerful and robust method to match two point clouds. It is this robustness that is utilized in this research. By incorporating this method of object recognition within the self-similarity framework it is expected that an even more powerful point cloud matching method will be created.

A Spin Image is just what it sounds like, a local image that is generated by spinning about a point's local normal. The spinning builds a histogram-like image that measure's the local distribution of points. The collection of all (or enough) Spin Images for a 3D object can be used to match it with another. Because this approach requires generating an image for each point this can be computationally intensive for large point clouds. Therefore, we limit computing Spin Images to only points that are deemed to be of interest by a feature detector. Otherwise, most spin images will be redundant and not provide any additional useful information. For example, all the Spin Images on a flat roof will look very similar and will likely match all flat surfaces. Therefore, a Spin Image representation of a point cloud is computed as described in Algorithm 3.1:

Algorithm 3.1 Spin Image

1: Define C = Point Cloud, F = Feature Points, w = Spin Image Width, b = Bin Size.

2: **for** $f \in F$ **do**

3: Define a local coordinate system such that:

Origin is at the feature point f.

$\hat{z}$ is the local normal unit vector.

$\hat{x}$ and $\hat{y}$ define the local tangent plane.

4: Let $N = \{p | \, \|p - f\|_C \leq w\}$ be all points within a box of width w

$\|\cdot\|_C$ is the Chebyshev metric

5: Initialize the Spin Image $S_f[i, j]$ as a 2D array of dimensions $\frac{w}{b}$ by $\frac{w}{b}$.

6: **for** $n \in N$ **do**

7: Let $\alpha = \sqrt{\|p - f\|^2 + |\hat{z} \cdot (n - f)|^2}$

8: Let $\beta = \hat{z} \cdot (n - f)$

9: Let $i = \lfloor \frac{\frac{w}{2} - \beta}{b} \rfloor$

10: Let $j = \lfloor \frac{\alpha}{b} \rfloor$

11: Increment $S_f[i, j] = S_f[i, j] + 1$

3.3 Self-Similar Spin Images

As described in Section 2.2, the self-similarity framework introduced in [11] and summarized in Figure 2.7 computes an attribute (e.g., local normal) for a data point (e.g., 2D pixel or 3D point) and its neighbors. This attribute is then compared against those neighboring attributes using some comparator metric (e.g., correlation) and then normalized. This collection of comparison values defines the self-similarity descriptor.

It is expected that by using a more powerful descriptor within the self-similarity framework that in turn a more robust method of feature matching can be achieved. In particular, using a well established robust descriptor such as Spin Images, instead of just the local normal, can provide a more descriptive correlation surface. Therefore, computing a "**Self-Similar Spin Image**" proceeds as described in Algorithm 3.2:

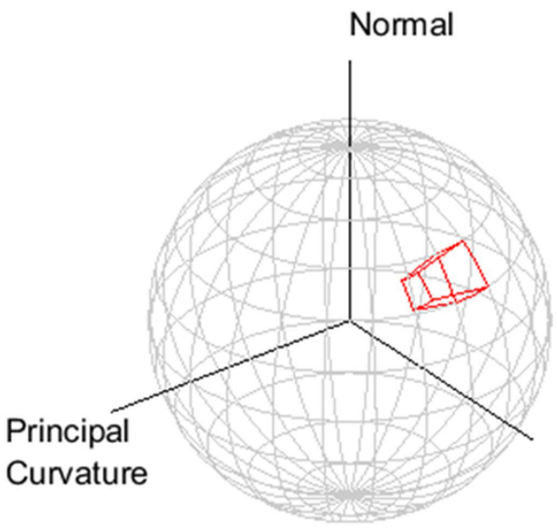


Figure 3.2: Descriptor local coordinate system and quantization

This spherical correlation surface is the desired feature descriptor that can be compared against other features in other points clouds to find a match. Figure 3.3 presents six radial slices of the descriptor. For the testing used in this dissertation the number of bins in the

Algorithm 3.2 Self-Similar Spin Image

1: Define C = Point Cloud, F = Feature Points, R = Neighborhood Size.

2: **for** $f \in F$ **do**

3: Let $N = \{p | \, \|p - f\| \leq R\}$ be all points within a radius of R (Figure 3.2)

4: Compute a Spin Image S_n for every point within this neighborhood $n \in N$.

5: Compute the similarity metric, $M_{f,n} = \text{correlation}(S_f, S_n)$

This metric is the correlation between the vectorized Spin Images.

6: Define a spherical coordinate system as defined in Figure 3.2 such that:

Origin is at the feature point f.

X axis is the local principal direction.

Z axis is the local normal.

Y axis is the cross product of Z and X to define a right-handed coordinate system.

7: Spherically bin this neighborhood with a given latitude/longitude/range resolution.

8: Assign each bin the average similarity metric $M_{f,n}$ within that bin.

9: The values in this correlation surface are then normalized to the range $[0, 1]$.

radial, longitude, and latitude directions are six, eight, and six, respectively.

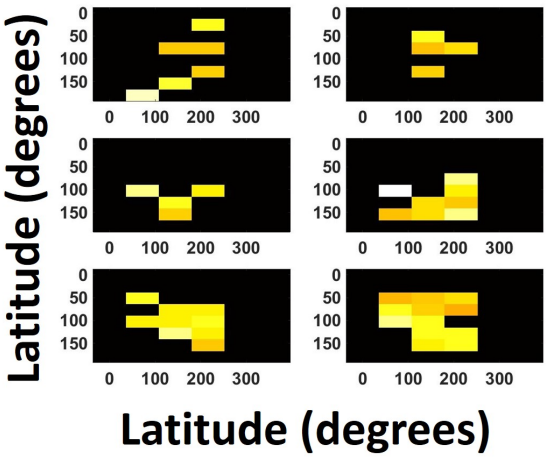


Figure 3.3: Example Self-Similar Spin Image descriptor displayed as six radial slices.

3.4 Self-Similar Keyscale

In order for the above descriptor to be truly scale-invariant the size of the spherical neighborhood must be determined. In the approach described in [15] the size of this neighborhood was empirically chosen to be four times "the detected scale at which the principal curvature reaches its local maxima". However, for this method to be scale-invariant and independent of user-interaction a method must be devised to determined the scale at which the point cloud descriptors should be computed. To determine this characteristic size of the point cloud we extend the technique introduced by Tamaki (et al) [54].

The technique relies on computing a "**keyscale**", which is a characteristic scale by which if two point clouds are scaled by the ratio of their respective keyscales then the point clouds can be matched. A point cloud's keyscale is computed by performing a Principal Component Analysis (PCA) of its Spin Images over a range of scales and finding the scale that yields the minimum cumulative contribution rate. The motivation behind this technique is that for both very small scales and very large scales all Spin Images will tend to look the same,

either all representing a plane or a point, respectively. Therefore, it is conjectured that there is an optimal scale in between. Computing the "**Keyscale**" of a point cloud proceeds as described in Algorithm 3.3:

Algorithm 3.3 Keyscale

1: Define F = Feature Points.

2: Let R_{min} and R_{max} be the minimum and maximum spherical neighborhood sizes.

3: Let T be the set of neighborhood sizes to be tested in the range $[R_{min}, R_{max}]$ such that the ratio of sequential scales are equal.

4: **for** $R \in T$ **do**

5: **for** $f \in F$ **do**

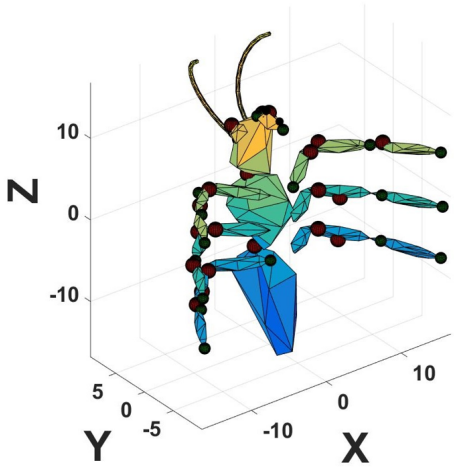
6: Compute Spin Image S_f with a neighborhood size R

7: Vectorize each S_f and compute the PCA decomposition of all S_f

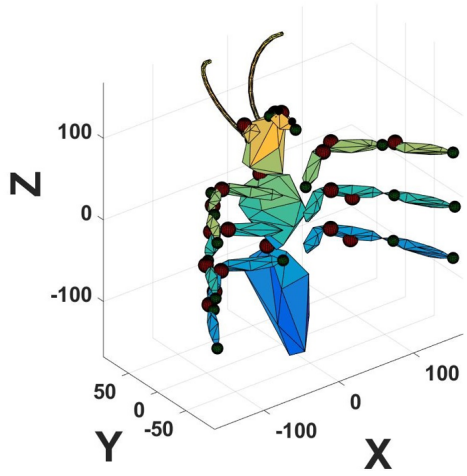
8: Compute the cumulative contribution rates of the PCA bands

9: Let $R_{keyscale}$ be the value of R that minimizes the cumulative contribution rate of the first principal component.

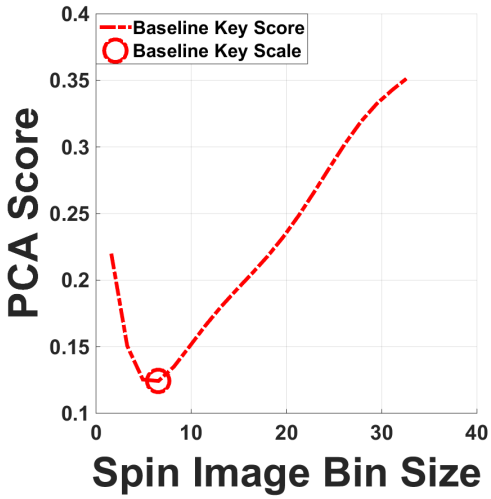
For example, Figure 3.4a and Figure 3.4b show a sample point cloud and the same cloud scaled by a factor of 10, respectively. The PCA scores for both point clouds are displayed in 3.4c and Figure 3.4d, respectively. The keyscale for the original ant point cloud is found to be 6.55 and for the scaled point cloud is 65.5 which yields a keyscale ratio of 10.



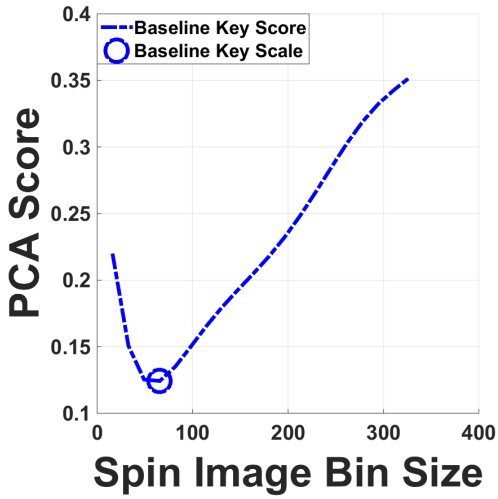
(a) Ant point cloud.



(b) Ant point cloud scaled 10x.



(c) Keyscale of (a) computed at 6.55.



(d) Keyscale of (b) computed at 65.5.

Figure 3.4: Example of point clouds with extracted feature points (a) and (b) and their keyscales computed from their PCA scores minimum (c) and (d). The ratio of their keyscales is 10 which correctly matches their relative scale.

In this research we extend this approach to compute the keyscale of a point cloud by performing this multi-scale PCA analysis on the Self-Similar Spin Images and not just

the Spin Images. By applying this scale matching technique to a local descriptor that captures the self-similarity of local geometric patterns instead of simply the distribution of neighboring points it is expected that this will provide a more robust scale detector. Therefore, computing the "**Self-Similar Keyscale**" of a point cloud proceeds as described in Algorithm 3.4:

Algorithm 3.4 Self-Similar Keyscale

1: Define F = Feature Points.

2: Let R_{min} and R_{max} be the minimum and maximum spherical neighborhood sizes.

3: Let T be the set of neighborhood sizes to be tested in the range $[R_{min}, R_{max}]$ such that the ratio of sequential scales are equal.

4: **for** $R \in T$ **do**

5: **for** $f \in F$ **do**

6: Compute Self-Similar Spin Image S_f with a neighborhood size R

7: Vectorize each S_f and compute the PCA decomposition of all S_f

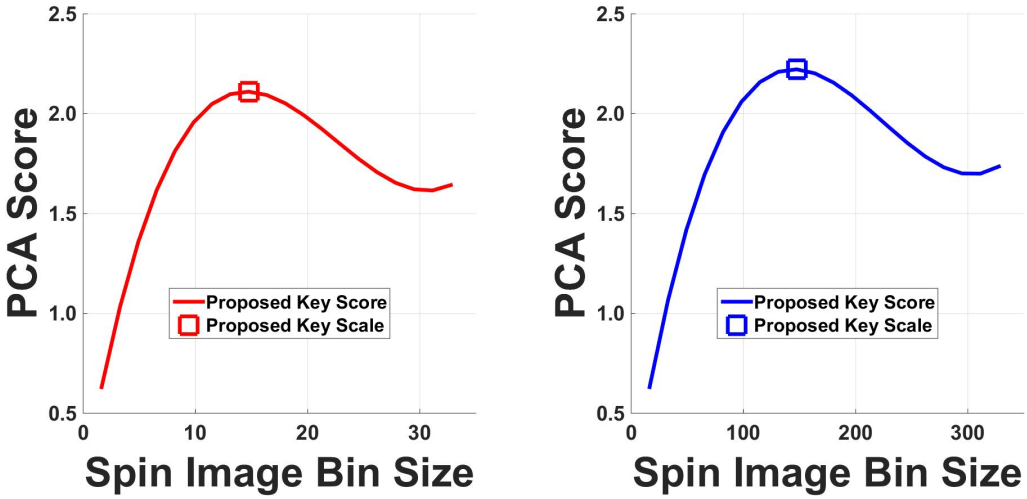
8: Compute the cumulative contribution rates of the PCA bands

9: Let $R_{keyscale}$ be the value of R that maximizes the cumulative contribution rate of the first principal component.

The values for R_{min} and R_{max} must be selected. For this dissertation they are selected as multiples of a characteristic size of the point cloud. Specifically, we use the ϵ, the average nearest neighbor distance of all points in the point cloud, as the characteristic size and we

use $R_{min} = \epsilon$ and $R_{max} = 20 * \epsilon$.

Figure 3.5a shows an example of the PCA scores and the highlighted keyscale found at the curve maximum. It should be noted that unlike the standard keyscale approach introduced in [54] the keyscale is not the minimum of the PCA rate curve but for Self-Similar Keyscale it is the maximum. The motivation behind this approach is similar to that of the keyscale approach in [54], whereas values of R that are either too small or too large will yield similar flat spin images and so an optimal value in between is expected.



(a) Keyscale of Figure 3.4a computed at 14.74.

(b) Keyscale of 3.4b computed at 147.4.

Figure 3.5: Example of self-similar keyscales computed from their PCA scores maximum (a) and (b). The ratio of their keyscales is 10 which correctly matches their relative scale.

3.5 Point Cloud Registration

In order to register two point clouds and find the corresponding transformation that will align them, we first detect features in both clouds as described in 2.3.1. Then for each detected feature we compute their Self-Similar Spin Images described in 3.3. These descriptors are then analyzed to compute their Self-Similar Keyscale described in 3.4. The point

clouds are then scaled to match their relative keyscales.

The scaled point clouds feature descriptors are compared by computing the correlation coefficient between the two surfaces. Descriptors are considered a match if they satisfy the Nearest Neighbor Distance Ratio (NNDR) where a feature x in point cloud X matches a point y_1 in point cloud Y if it satisfies the criteria

$$\frac{correlation(x, y_1)}{correlation(x, y_2)} < threshold \tag{3.1}$$

where y_1 and y_2 are the first and second nearest neighbors to x. The threshold was found to be fairly insensitive and was empirically found to be 0.75.

Algorithm 3.5 Self-Similarity based Registration

1: **for** $i \in \{1, 2\}$ **do**

2: Let $P_i = i^{th}$ point cloud

3: Compute the feature point set F_i of P_i using MoPC

4: Compute the Self-Similar Spin Images S_i at each point in F_i

5: Compute the Self-Similar Keyscale k_i of P_i

6: Match scales by rescaling P_2 to match the scale of P_1 using their keyscales, $P_2 \leftarrow \frac{k_1}{k_2} P_2$

7: Compute $E_{n,m} = ||s_{1,n} - s_{2,m}||$ the Euclidean distance between all feature descriptors in $s_{1,n} \in S_1$ and $s_{2,m} \in S_2$

8: Compute feature matches by keeping those (n, m) pairs that match the NNDR criteria

9: Compute the affine transformation between matched features using RANSAC method.

Finally, to mitigate against false alarms distorting the rigid body and scale transformation computed we use the RANSAC method to detect and remove outliers from the match feature points. The non-outliers matched points are used to compute the final affine transformation between the point clouds. This matching process will yield a coarse registration and get the two clouds close to each other. The entire self-similarity based registration process is outlined in Algorithm 3.5.

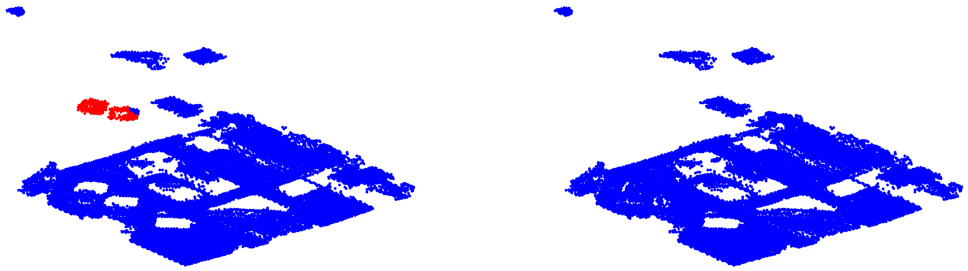
Chapter 4: Point Cloud Change Detection

4.1 Overview of Change Detection

A variety of techniques have been proposed to perform change detection between point clouds. Using a point cloud derived Depth Map has been applied to change detection computing a DEM for each point cloud and then finding significant changes in depth between the maps [61]. Change detection has also been performed directly on the 3D point cloud points themselves [59] by using an octree representation of the point cloud and then computing distance metrics between the point in each octree cell. Distance metrics considered were average distance and the Hausdorff distance.

4.2 Nearest Neighbor Order Statistics

Human analysts are able to visually detect changes between points clouds with relative ease, see Figure 4.1a and Figure 4.1b. Therefore, it seems appropriate to develop an approach to detecting changes between two point clouds by modeling it after the visual cues that humans use when noticing changes. When detecting a change between two co-registered point clouds a human is observing that there are points in one point cloud that are far from any points in the other cloud. However, they are also making a relative judgment about the distribution of distances between those points. If there are many points at the same relative distance from points in the other cloud then they are less likely to classify those points as a change. Instead, they will dismiss that as a translation bias in the data and not a change.



(a) Reference Point Cloud (b) Target Point Cloud

Figure 4.1: Buildings in reference that are missing in target are highlighted red.

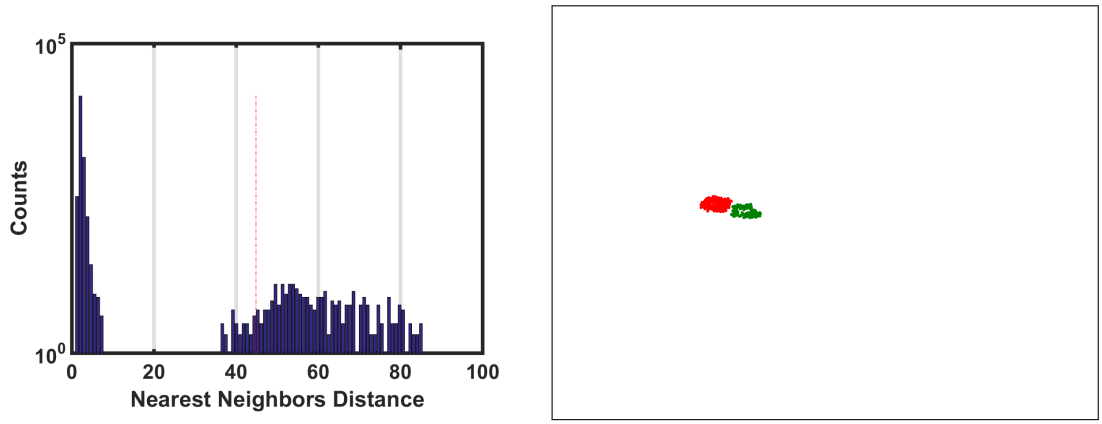
Therefore, any method that detects changes should also account for this spatial distribution of points. Lastly, humans are also capable of making such evaluations at different scales in the data. For example, the number of points needed to classify a change as being a new building in the scene need not be the same as to classify them as a new car. Therefore, applying algorithm parameters and thresholds that detect changes at one scale can fail to detect changes in another.

The approach developed in this dissertation is similar in spirit to that presented in [59] but exploits a more robust metric to detect changes and a different spatial representation of the point clouds. In particular, the point clouds are spatially represented by a voxel representation at multiple resolutions and then at each resolution level a histogram is computed of all nearest neighbor distances between the two clouds. These histograms are then analyzed to detect changes. It is expected that since non-changes are the norm in a scene then these histograms will be heavily unimodal and changes will be found in the tails of the histogram.

With these motivations in mind we propose the "**Nearest Neighbor Order Statistics**" method for change detection. The method takes as input a pair of point clouds,

computes a multi-level voxelization of the space enclosing the clouds, the distances between point clouds in each voxel are analyzed and thresholded, the thresholded points are then segmented to form clusters of objects. These clustered objects are the detected changes between the clouds. This process computes these changes as described in Algorithm 4.1.

Therefore, at the largest voxel there is one histogram. Thresholded points will correspond to the largest possible new targets that are in P_2 but not in P_1. Thresholded points at smaller voxels correspond to smaller and smaller new targets. It should be noted that the Hausdorff distance [59] used to detect changes is just a special case of this approach. The Hausdorff distance is the maximum of the nearest neighbor distances computed.



(a) Reference-to-Target distance histogram.

(b) Two clusters detected using NNOS approach.

Figure 4.2: The histogram thresholded (in red) is computed using the triangle method. Right mode corresponds to missing buildings and left mode is the persistent background. Two clusters of changes are detected between reference and target point clouds using min-cut segmentation.

Finally, the threshold value applied to the nearest neighbor histograms must still be determined. Since change detection by definition assumes that most of the data under consideration is part of the background and only a small fraction of the scene changes, the changes to be detected will correspond to small perturbations in the histogram tail. This research uses the **triangle method**, where a line is formed between the histogram peak

and the maximum bin (which forms a triangle with the x and y axis) and the perpendicular distance is computed between each bin and the formed line. The histogram bin with the largest distance is the threshold (see Figure 4.2a). Using this threshold leads to the desired clusters (see Figure 4.2b).

Algorithm 4.1 Nearest Neighbor Order Statistics Change Detection

1: Let P_1 and P_2 be two co-registered point clouds

2: The largest possible voxel is constructed (i.e., the bounding box of the entire data set).

3: Point clouds are voxelized at the next resolution level such that each voxel from the previous level is subdivided into 8 equal voxels with their sides reduced in half.

4: This process of producing smaller voxels by powers of 2 is continued for N levels.

5: At each level, every voxel is processed to compute the nearest neighbor in P_2 for every point in P_1.

6: For each voxel a histogram is computed of these nearest neighbor distances.

7: The nearest neighbor histograms are then thresholded (see Figure 4.2a). Any points with a nearest neighbor distance greater than the threshold are considered changes.

8: To mitigate against false alarms, the smallest allowed voxel should be much larger (i.e., an order of magnitude) than the smallest allowable object size.

9: Segment detected points into connected sets using a graph min-cut segmentation

10: Discard connected sets of points that are smaller than some smallest allowable object size

Chapter 5: Experiments

To evaluate the performance of the proposed algorithms three categories of experiments are performed as listed in Table 5.1. The purpose of the Scale Matching experiments in Section 5.2 is to evaluate the ability of detecting the scale difference between a reference and target point cloud. Specifically, the reference point cloud has been manually modified by a Rotation/Scale/Translation (RST) transformation that contains a scale component that is to be computed using the Self-Similar Keyscale approach defined in Section 3.4. The purpose of the Feature Matching experiments in Section 5.3 is to evaluate the ability to spatially match the features of a reference and target point cloud. As in the Scale Matching experiments, the reference point cloud has been manually modified by a RST transformation that must be computed using the Self-Similar Spin Image approach defined in 3.3. The purpose of the Change Detection experiments in Section 5.4 is to evaluate the ability to find objects missing in a point cloud that are present in another. The reference point cloud has been manually modified to remove a series of objects to create the target point cloud. The missing objects are located using the Nearest Neighbors Order Statistics approach defined in 4.2.

Table 5.1: Break down of experiment algorithms tested

Category	**Algorithm**
Scale Matching	Self-Similar Keyscale
Feature Matching	Self-Similar Spin Images
Change Detection	Nearest Neighbor Order Statistics

5.1 Test Data

The feature matching and scale matching methods developed in Section 3.3 and Section 3.4, respectively, will be tested against model point cloud datasets described in Table 5.2 and shown in Figure 5.1a through 5.1d and actual Lidar described in Table 5.3 and shown in Figure 5.2a through 5.2d. In addition, to test the proposed and baseline methods for robustness against a wide range of terrains, a set of twenty five Lidar point clouds were also tested. This robustness test set are presented in the Appendix.

Table 5.2: Model point clouds for testing feature and scale matching.

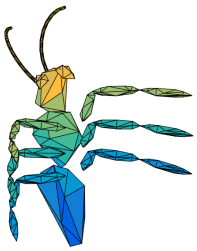
Name	**Type**	**Description**
Ant	Model	Small sample
Beethoven	Model	Large sample
Chopper	Model	Complex sample
Turbine	Model	Medium sample

Table 5.3: Lidar point clouds for testing feature and scale matching.

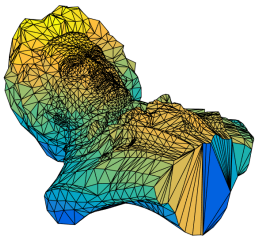
Name	**Type**	**Description**
PC-SMALL	Lidar	Small-sized features
PC-MEDIUM	Lidar	Medium-sized features
PC-LARGE	Lidar	Large-sized features
PC-MIX	Lidar	Mix-sized features

The change detection method developed in Section 4.2 will be tested against actual Lidar datasets described in Table 5.4 and shown in Figure 5.3a through Figure 5.3e.

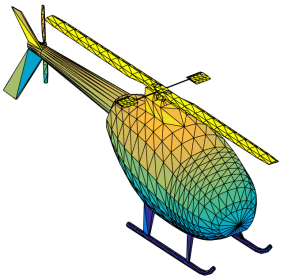
For the experiments, the test point clouds are transformed using a variety of transformations to test the limits of the proposed algorithms. The transformations and the labels used to identify them are described in Table 5.5.



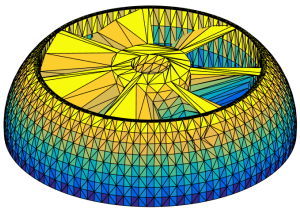
(a) Ant



(b) Beethoven



(c) Chopper



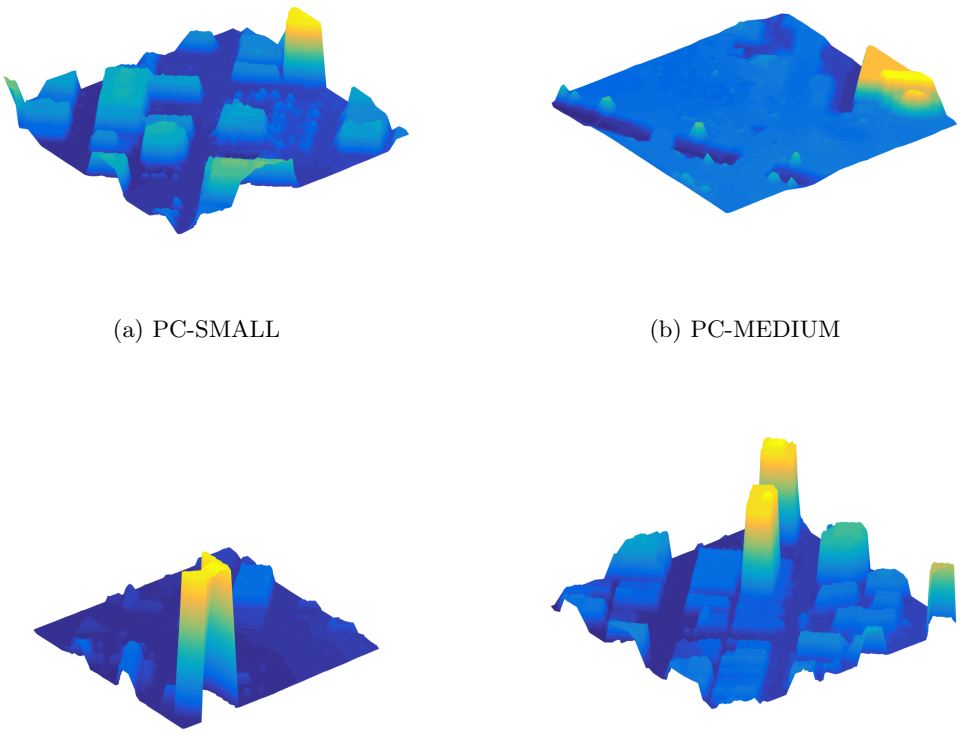
(d) Turbine

Figure 5.1: Model point clouds.

5.2 Scale Matching

For the scale matching experiments the test point clouds were tested using the S10 transformation. The figures presented in this section represent the nominal results observed. For the full set of results see the Appendix.

The scale matching step described in Section 3.4 requires computing a PC analysis of the local descriptors (Self-Similar Spin Images for the proposed method and Spin Images for the baseline) for all extracted features over multiple scales. This yields a point cloud's keyscale which is defined by the authors in [54] as "the scale that gives the minimum of



(a) PC-SMALL (b) PC-MEDIUM

(c) PC-LARGE (d) PC-MIX

Figure 5.2: Lidar point clouds.

cumulative contribution rate of PCA at a specific dimension of eigenspace".

However, initial experiments using the model point clouds (Figure 5.4a and 5.4b) showed that using the minimum to define the keyscale proved valid only for the baseline algorithm using Spin Images. For the proposed algorithm using Self-Similar Spin Images the maximum needed to be computed. Therefore, a more general definition for a point cloud's keyscale would be "the scale that gives the **extremum** of cumulative contribution rate of PCA at a specific dimension of eigenspace".

The experiments for the Lidar point clouds showed similar results for the proposed

Table 5.4: Point clouds for testing change detection.

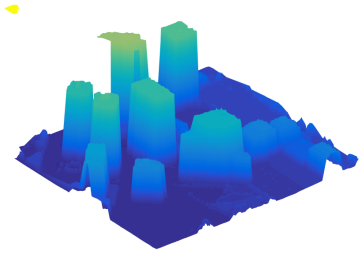
Name	Type	Description
PC-REFERENCE	Lidar	Reference used for change detection
PC-TARGET-1	Lidar	Reference with 2 buildings manually removed
PC-TARGET-2	Lidar	Reference with 3 buildings manually removed
PC-TARGET-3	Lidar	Reference with 4 buildings manually removed
PC-TARGET-4	Lidar	Reference with 5 buildings manually removed

Table 5.5: List of transformations applied to test point clouds

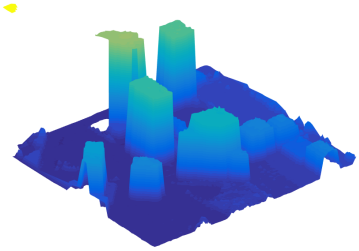
Label	Rotation	Scale	Translation
Y180	180 degree about Y	None	None
S10	None	10	None
RST	-45 degree rotation about Y	2	Maximum point cloud dimension

algorithm as shown in Figure 5.5a and Figure 5.5b. However, for the baseline algorithm the PC analysis curves did not have any local minimums. Therefore, a keyscale could not be found. The curves clearly have a global minimum but this is an unreliable metric since it is arbitrarily dependent on the minimum scale chosen for the computation.

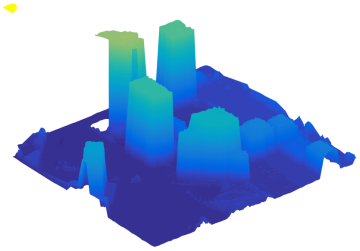
Finally, in order to evaluate and compare the performance of the proposed and baseline methods for a wide range of terrains and misalignment transformations they were tested against the robustness test set. The results presented in Table 5.6 show both the baseline and proposed did a good job of estimating the scale for the simplest transformations S10 and Y180 with errors ranging from 0.21% to 1.64%. However, in both cases the proposed method still reduced the error over that of the baseline by 67.42% for S10 and 68.83% for Y180. For the more complex RST transformation the proposed method, with an error of 1.86%, outperformed the baseline method, with an error of 14.41%, again reducing the error by 87.10%. Finally, the proposed method not only reduced the average error in estimating scale differences over the baseline method for all test cases but it also had a smaller standard



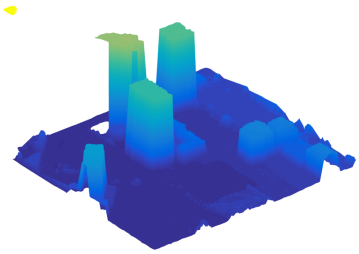
(a) PC-REFERENCE



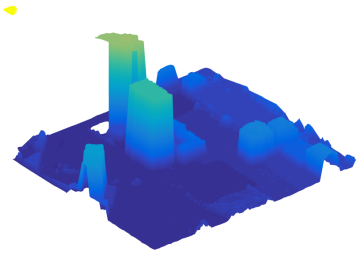
(b) PC-TARGET-1



(c) PC-TARGET-2

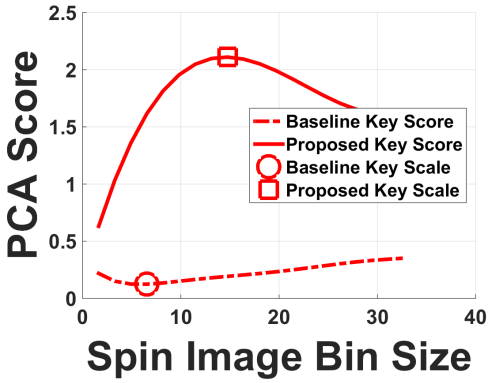


(d) PC-TARGET-3

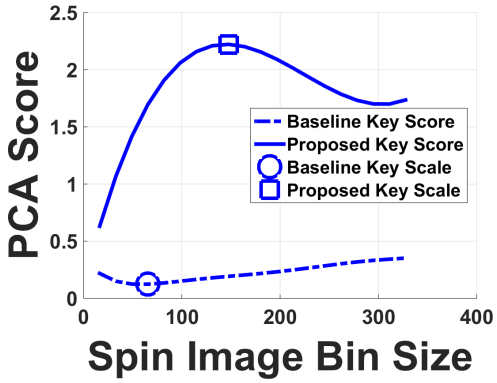


(e) PC-TARGET-4

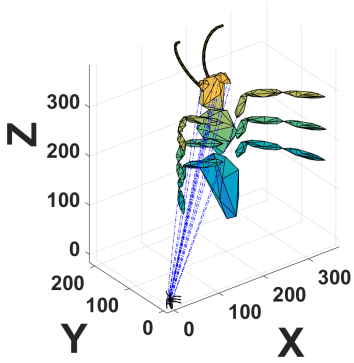
Figure 5.3: Lidar change detection point clouds.



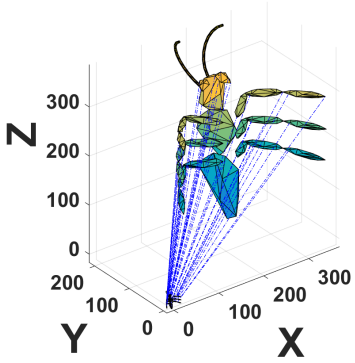
(a) Ant PCA scores. Baseline keyscale found at 6.55 and proposed keyscale found at 14.74.



(b) Ant (S10) PC scores. Baseline keyscale found at 65.5 and proposed keyscale found at 147.4.



(c) Ant matched to Ant (S10) with baseline method



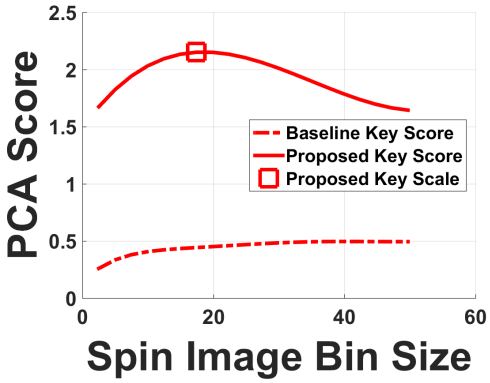
(d) Ant matched to Ant (S10) with proposed method

Figure 5.4: The scale difference between the Ant point cloud and its scaled version was detected by both the baseline and proposed method. The ratio of keyscales for both the baseline and proposed methods yields 10.

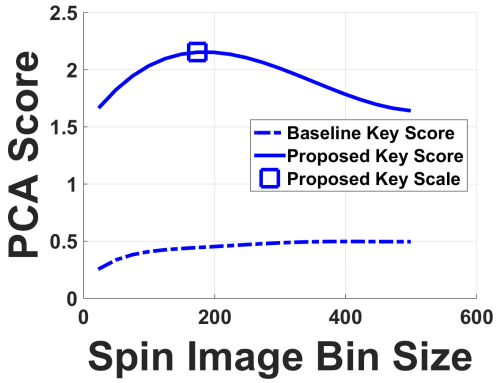
deviation over the entire robustness test cases.

5.3 Feature Matching

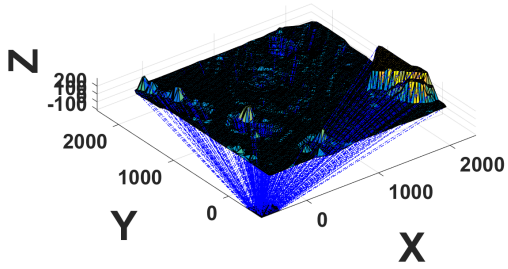
For the feature matching experiments the test point clouds were tested using the RST transformation. The figures presented in this section represent the nominal results observed. For the full set of results see the Appendix.



(a) PC #3 PC scores. Baseline keyscale not found and proposed keyscale found at 17.38.



(b) PC #3 (S10) PC scores. Baseline keyscale not found and proposed keyscale found at 173.8.



(c) PC #3 matched to PC #3 (S10) with self-similar spin images.

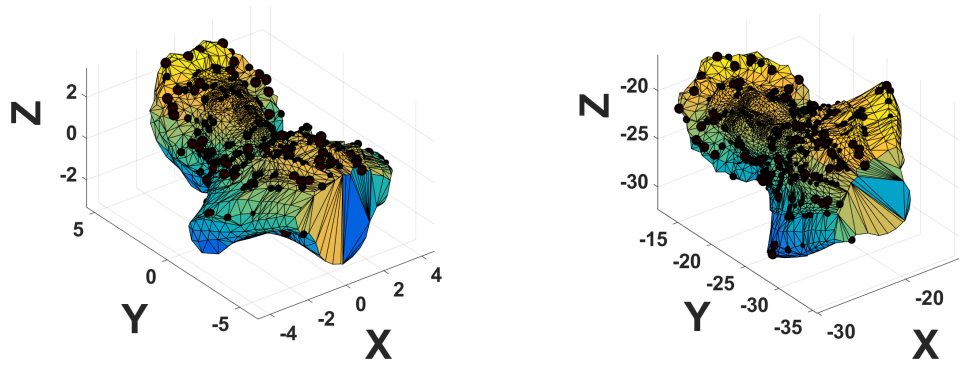
Figure 5.5: The scale difference between the Ant point cloud and its scaled version was detected by the proposed method but not the baseline method. The ratio of the keyscale for proposed method curves yields the correct scale difference of 10. The baseline method did not have any local minimums and therefore did not yield a keyscale.

The feature matching and registration step described in Section 3.5 requires computing a local descriptor (Self-Similar Spin Images for the proposed method and Spin Images for the baseline) for all extracted features, matching them, and then estimating the transformation. Figure 5.6a and Figure 5.6b show the extracted feature points for the Beethoven point cloud and its transformed point cloud, respectively.

The resulting matched features are shown in Figure 5.7a and 5.7b. Table 5.7 provides

Table 5.6: Results of performing scale matching on the robustness dataset.

Case	Algorithm	Truth	Avg	Std	Error (%)	Improvement (%)
S10	Proposed	10	9.98	0.11	0.21	67.42
S10	Baseline	10	10.06	0.25	0.64	-
Y180	Proposed	1	1.01	0.03	0.51	68.83
Y180	Baseline	1	0.98	0.07	1.64	-
RST	Proposed	2	1.96	0.24	1.86	87.10
RST	Baseline	2	1.71	0.41	14.41	-



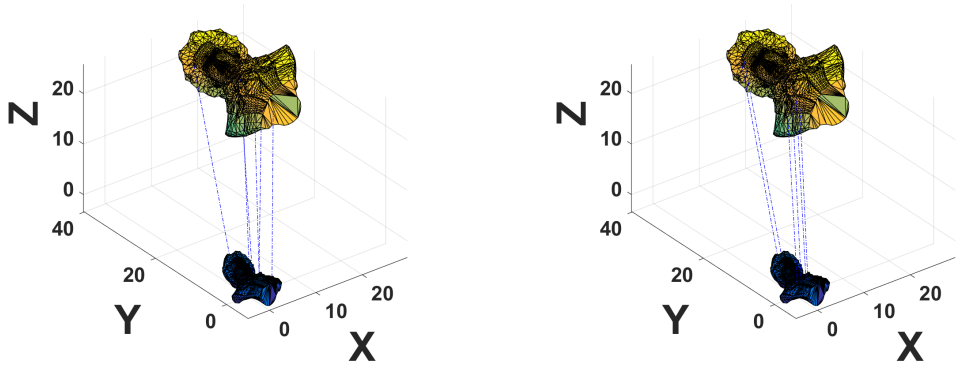
(a) Beethoven feature points

(b) Beethoven (RST) feature points

Figure 5.6: The feature points identified using the MoPC metric for the point cloud in (a) and its transformed version (b).

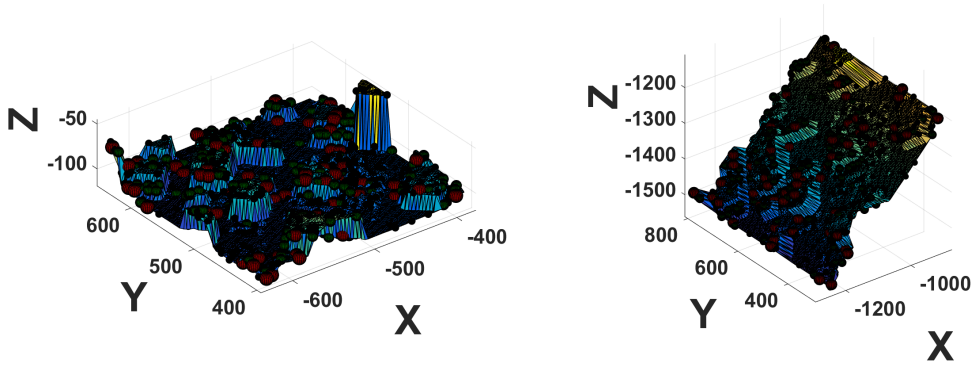
the transformation parameters estimated from these matched feature points as well as the true parameter values. The results show that the proposed method was able to correctly estimate the rotation, scale, and translation parameters (total error of 7.7831e-07%) while the baseline correctly estimated the rotation angle only but not the scale or translation parameters (total error of 15.33%). Figure 5.8a and Figure 5.8b show the extracted feature points for the one of the Lidar point clouds and its transformed point cloud, respectively.

The resulting matched features are shown in Figure 5.9a and 5.9b. Table 5.8 provides the transformation parameters estimated from these matched feature points as well as the



(a) Beethoven matches with baseline method (b) Beethoven matches with proposed method

Figure 5.7: The Beethoven point clouds matched using the feature points for the baseline method (a) and proposed method (b).



(a) PC-SMALL Features (b) PC-SMALL (RST) Features

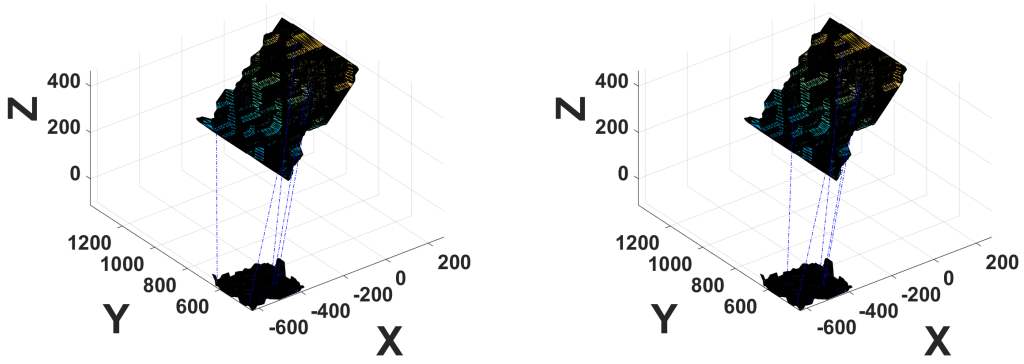
Figure 5.8: The feature points identified using the MoPC metric for the point cloud in (a) and its transformed version (b).

true parameter values. The results show that the proposed method was able to provide a better estimate (total error of 3.29%) for the rotation, scale, and translation parameters than the baseline method (total error of 27.94%).

Finally, in order to evaluate and compare the performance of the proposed and baseline

Table 5.7: Transformation parameters estimated using the baseline and proposed methods. The proposed method correctly estimated the RST parameters. The baseline method did not.

Parameter	**Truth**	**Baseline**	Error (%)	**Proposed**	Error (%)
Rotation Angle	-45	-45	0	-45	0
Rotation Axis X	0	0	0	0	0
Rotation Axis Y	1	1	0	1	0
Rotation Axis Z	0	0	0	0	0
Scale	2.0	1.75	-12.50	2.0	0
Translation X	-11.5751	-13.1363	13.49	-11.5750	-0.00086
Translation Y	-11.5751	-13.1981	14.02	-11.5750	-0.00086
Translation Z	-11.5751	-13.0410	12.66	-11.5750	-0.00086
Total Error	-	-	15.33	-	7.7831e-07



(a) PC-SMALL matches with baseline method (b) PC-SMALL matches with proposed method

Figure 5.9: The PC-SMALL point clouds matched using the feature points for the baseline method (a) and proposed method (b).

methods for a wide range of terrains and misalignment transformations they were tested against the robustness test set. The results presented in Table 5.9 show both the baseline and proposed did an very good job of estimating the misalignment transformation for the simplest transformations S10 and Y180 with trivial errors ranging from 2.60E-11% to 1.72E-10%. For the more complex RST transformation the proposed method, with an error of

Table 5.8: Transformation parameters estimated using the baseline and proposed methods. The proposed method provides a better estimate for the RST parameters than does the baseline method.

Parameter	**Truth**	**Baseline**	Error (%)	**Proposed**	Error (%)
Rotation Angle	-45	-44.8346	-0.3675	-44.8147	-0.4119
Rotation Axis X	0	-0.0040	-	-0.0053	-
Rotation Axis Y	1	0.9998	-0.0230	0.9997	-0.0287
Rotation Axis Z	0	0.0211	-	0.0234	-
Scale	2.0	1.7778	-11.1111	2.0	0
Translation X	-249	-322.8787	29.6702	-255.8789	2.7626
Translation Y	-249	-224.4231	-9.8703	-259.3456	4.1549
Translation Z	-249	-340.3450	36.6847	-255.7538	2.7124
Total Error	-	-	27.9366	-	3.2885

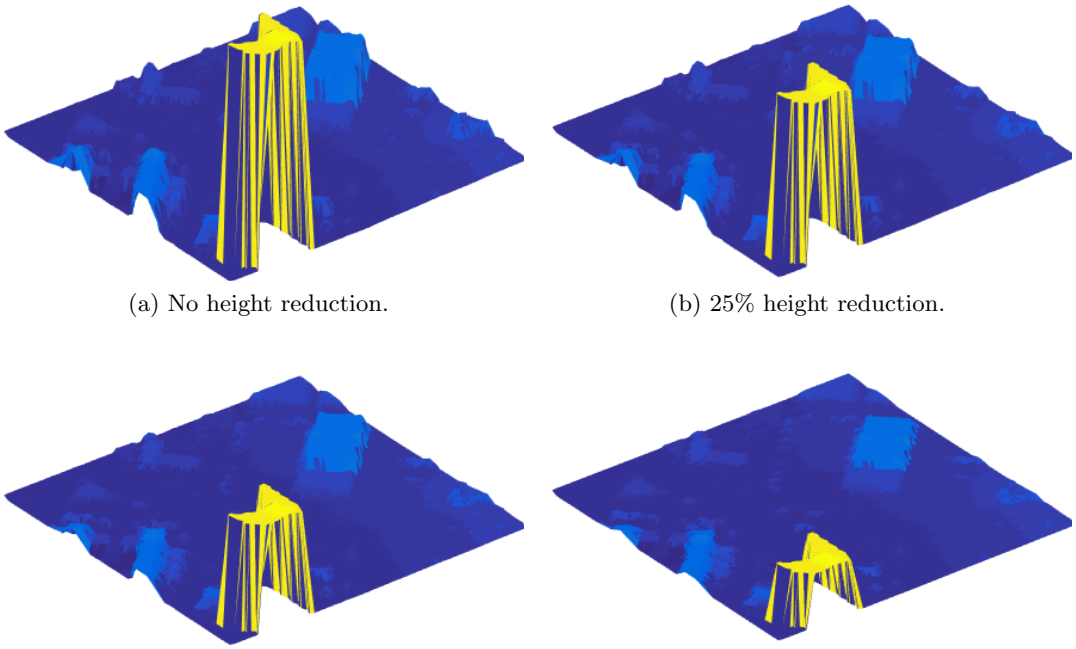
18.92%, outperformed the baseline method, with an error of 33.23%, reducing the error by 43.08%. Finally, the proposed method not only reduced the average error in estimating scale differences over the baseline method but it also had a smaller standard deviation over the all the test cases.

Table 5.9: Results of performing feature matching on the robustness dataset.

Case	**Algorithm**	**Error (%)**	**Std (%)**	**Improvement (%)**
S10	Proposed	1.72E-10	5.47E-11	-4.39
S10	Baseline	1.64E-10	5.37E-11	-
Y180	Proposed	2.60E-11	2.84E-11	9.74
Y180	Baseline	2.88E-11	3.10E-11	-
RST	Proposed	18.92	17.59	43.08
RST	Baseline	33.23	29.94	-

Lastly, a sensitivity study was performed in order to assess the performance of the proposed method when subjected to non-homogeneous transformations, like a distortion in the height of the data. To perform this study all LIDAR datasets were reduced in height for a range of scales and the alignment error computed. The first study reduced only the

heights of the buildings by scale factors of 10%, 25%, 50%, 75%, and 90% and left all other dimensions unaltered (see Figure 5.10).



(a) No height reduction.

(b) 25% height reduction.

(c) 50% height reduction.

(d) 75% height reduction.

Figure 5.10: Example Lidar (PC-LARGE) being reduced in height for a range of scales.

The proposed method proved to be very stable for large distortions up to a 75% reduction (see Figure 5.11). The method's sensitivity begins to show at a 90% reduction, which is quite large and a rather unrealistically large distortion to be found in any real dataset. However, the overall alignment errors are trivially small, on the order of 10^{-8}.

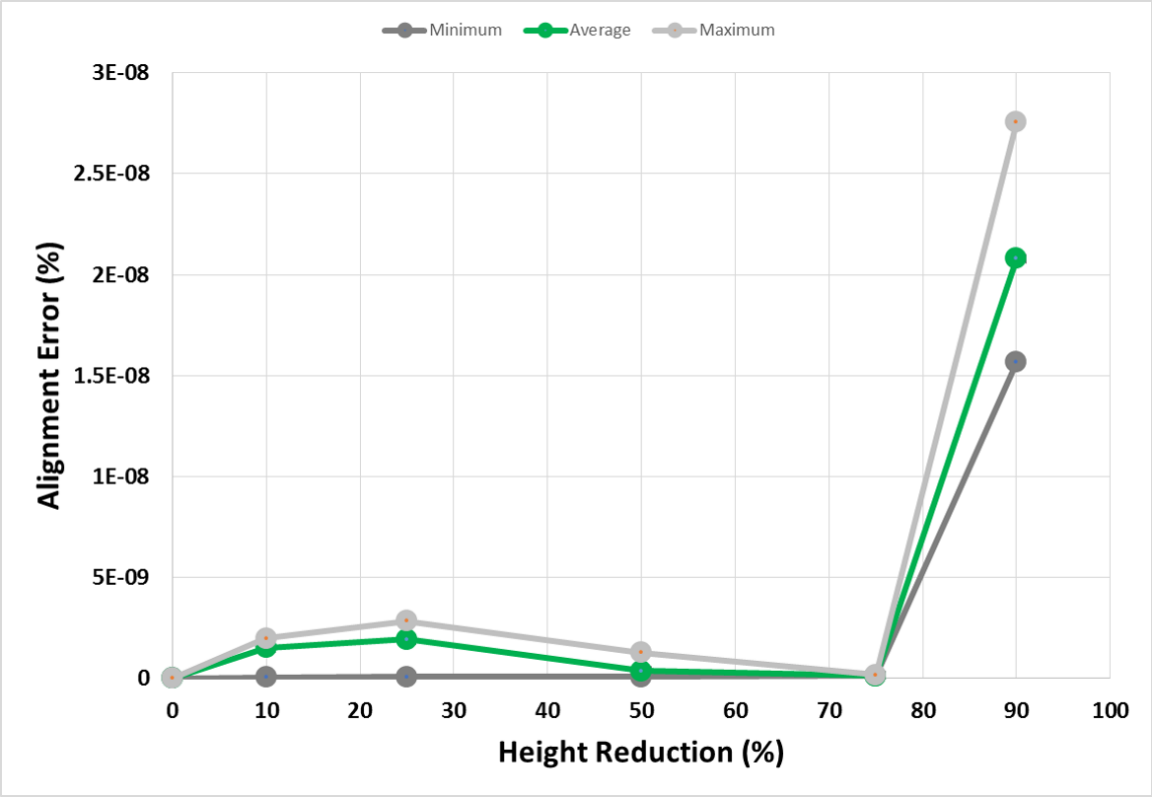


Figure 5.11: Minimum, average, and maximum alignment error over all 25 Lidar datasets for the proposed method over a range of height-only reductions. The errors were stable and trivially small up to 75% and began to grow at 90%.

Therefore, the same sensitivity was performed using the same reductions in height but the RST transformation was also applied (see Figure 5.12). Similarly, the proposed method was very stable for large distortions up to a 75% reduction and grew unstable at a 90% reduction level which, again, is quite large. These sensitivity errors only begin to grow when buildings are shrunk so much that the terrain and the buildings become ambiguous.

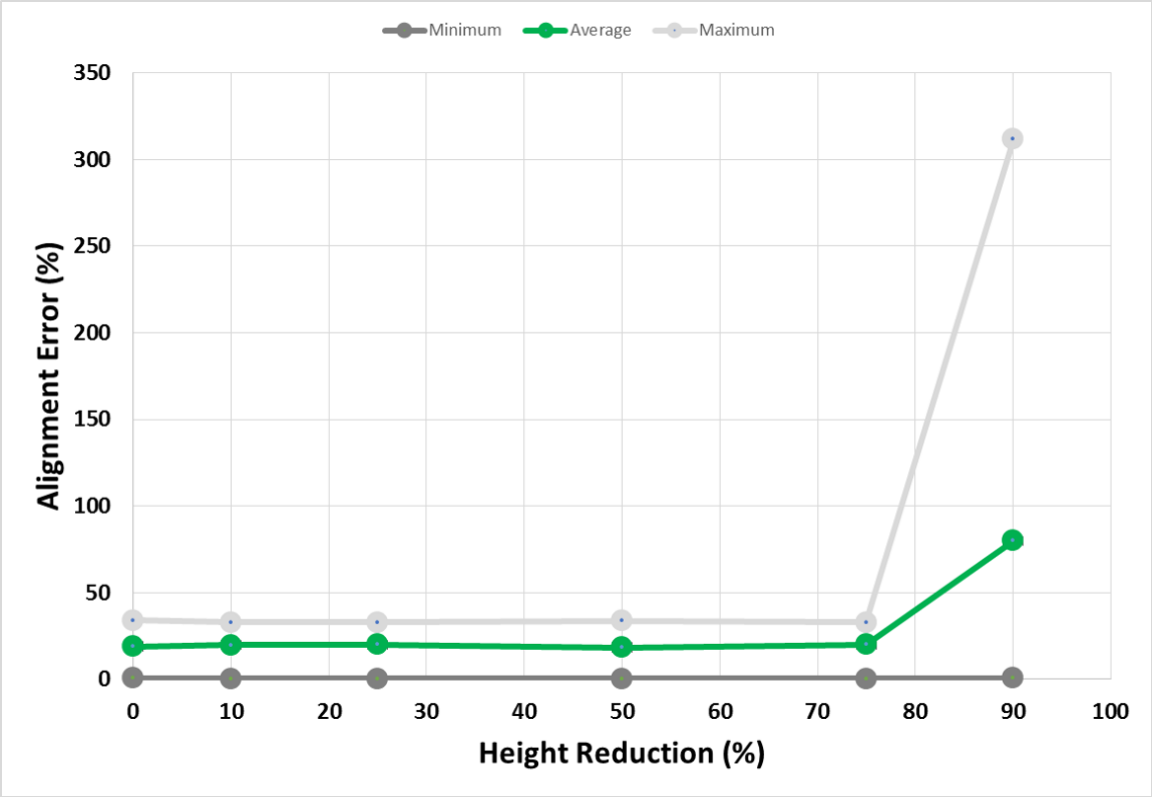
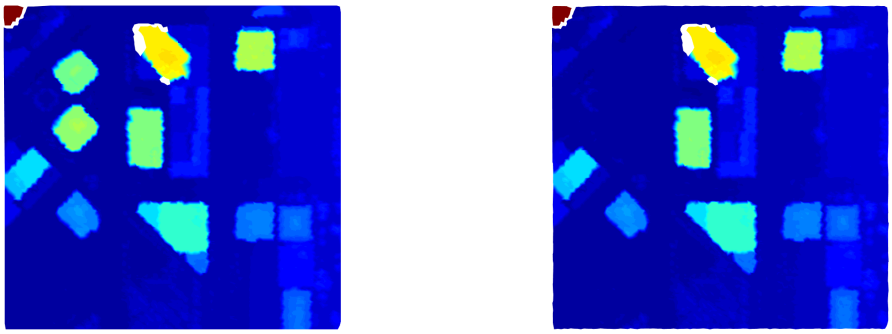


Figure 5.12: Minimum, average, and maximum alignment error over all 25 Lidar datasets for the proposed method over a range of height reductions combined with the RST transformation. The errors were stable up to 75% and began to grow at 90%.

5.4 Change Detection

For the change detection experiments the reference point cloud, see Figure 5.3a, was tested against a series of similar point clouds of the same location but various buildings missing. The figures presented in this section represent the nominal results observed. For the full set of results see the Appendix.

The change detection step described in Section 4.2 requires building voxels of two registered point clouds, computing the nearest neighbor distance between the two point clouds for each point in each voxel, building histograms of these distances, thresholding them, and

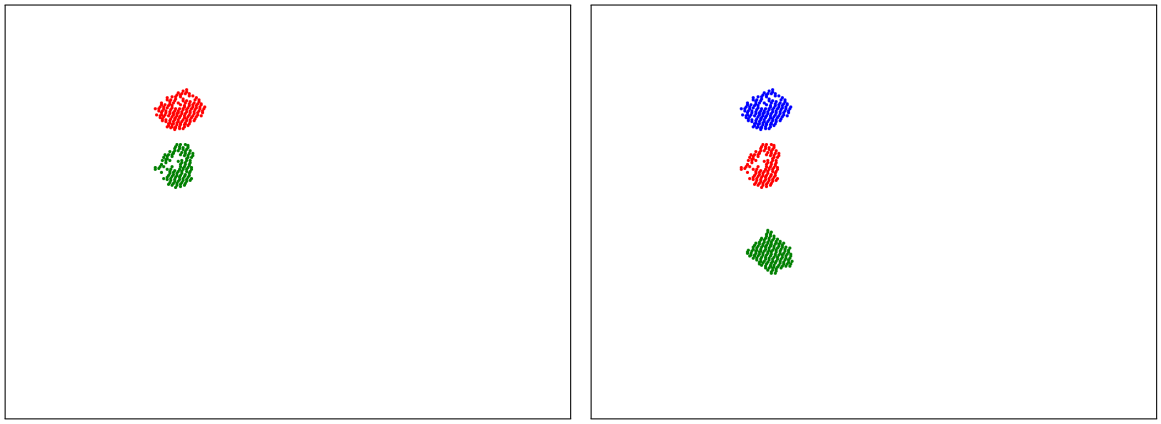


(a) PC-REFERENCE (b) PC-TARGET-1

Figure 5.13: 2D projections of the reference and first target point clouds. There are two missing building in the upper left of the target point cloud.

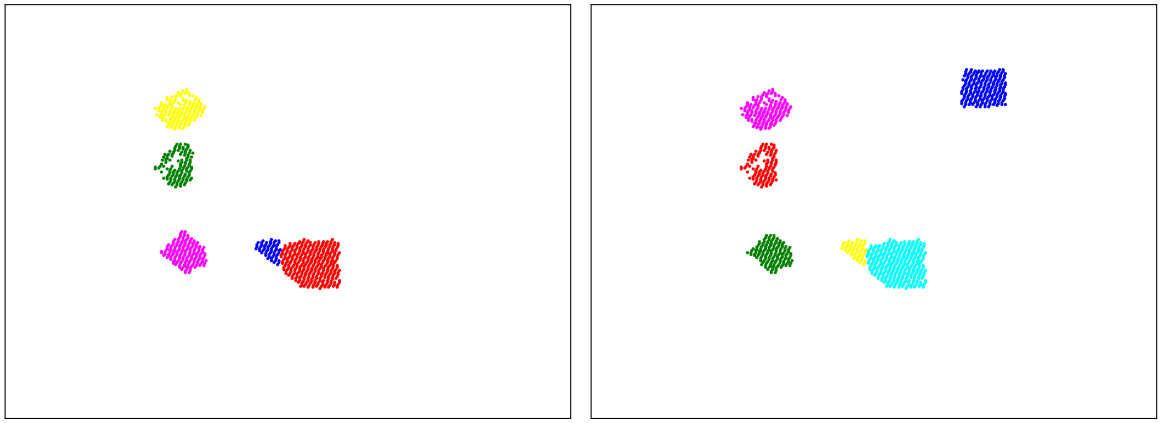
the segmenting the outliers into clusters. Figure 5.13a and Figure 5.13b show 2D projections of the reference point cloud and the first target point cloud. Figure 5.14a through Figure 5.14d present the clustering results for all the target point clouds after the outliers are clustered using a graph min-cut algorithm. The results show the correct number of buildings identified for each test case.

Note, initial inspection for the cases presented in Figure 5.14c and Figure 5.14d would appear that the clustering generated an extra building. However, for both of these cases the building on the lower right has a portion of its rooftop with a significant drop in altitude. A subsequent cluster merging step may be needed to address these issues.



(a) PC-TARGET-1

(b) PC-TARGET-2



(c) PC-TARGET-3

(d) PC-TARGET-4

Figure 5.14: Clusters found for each target point cloud's outliers using the graph min-cut algorithm. The extra cluster found in (c) and (d) is from a sharp change in altitude from the building.

Chapter 6: Conclusions and Future Work

6.1 Overview

Given the need to match misaligned 3D point clouds in a variety of fields, this dissertation developed an automated method to align two point clouds with both spatial and scale misalignments. The misalignment between point clouds comes from a variety of sources such as differing modalities, coordinate mismatches, variations in scenery, and changes in point densities. Using different modalities (e.g., Lidar and stereo-imagery) to extract the point clouds can lead to challenges in defining a feature descriptor that can be meaningfully compared despite the underlying data coming from very different datasets. Mismatches between the coordinate systems can lead to both spatial and scale differences if, for example, a Lidar point cloud is geo-registered but a stereo-imagery derived point cloud is not then one cloud will be in physical units while the other is not. Variations in scenery will clearly cause problems when content in one cloud is not be present in the other and can therefore lead to false alarms and missed features. Changes in point density can lead to differences in the feature descriptor used to match features and can lead to them not being similar enough to match.

Therefore, to overcome many of these problems, this dissertation proposed using a self-similarity based feature descriptor as the basis of its matching process because it can identify local patterns regardless of how they are generated. Combining this property with a robust local metric like spin images leads to a robust feature descriptor introduced Self-Similar Spin Images. Moreover, the distribution of these new descriptors was then analyzed in order to match any scale difference between point clouds by defining a Self-Similar Keyscale for each cloud. Finally, once two point clouds have been matched in space and scale they can be compared. As an example of processing matched clouds, this dissertation introduced a

change detection solution that compares the distances between nearest neighbors between the two clouds referred to as the Nearest Neighbor Order Statistic method. The proposed solutions were tested in three sets of experiments - Scale Matching, Feature Matching, and Change Detection.

6.2 Scale Matching

The Scale Matching experiments evaluated the ability to detect scale differences between a reference point cloud and a target point cloud that was manually modified by magnification in scale. The scale component was computed using the Self-Similar Keyscale approach. The robustness test set of Lidar point clouds were tested by applying the S10 transformation (magnifying them by a scale of 10x) and detecting the change using both the proposed and baseline methods.

This initial test was performed to focus on the scale matching and keep other distortions constant. The mean scale change detected using the proposed method was 9.98 and using the baseline method was 10.06. While both methods did a good job detecting the correct scale difference the proposed method reduced the error in detecting the scale difference by a factor of 67%. In addition, the same data set was modified using the RST transformation (that had a scale change of 2). In this case, the mean scale change for the proposed and baseline methods were 1.96 and 1.71, respectively. In this case of a more complex misalignment the proposed did a superior job of detecting the scale difference and reduced the detection error by 87%. Lastly, the simplest transformation Y180 (a rotation about the y-axis by 180 degrees) had a scale change of 1. In this case, the mean scale change for the proposed and baseline methods were 1.005 and 0.983, respectively. In this case of a simple misalignment the proposed did a superior job of detecting the scale difference and reduced the detection error by 69%. In all cases, the proposed method was able to detect the scale change between two point clouds and outperformed the baseline method.

Therefore, for the problem of scale matching the proposed method Self-Similar Keyscales was able to accurately match the scales of the test cases over a variety of point cloud types.

In particular, the proposed method was able to outperform its baseline counterpart by exploiting a more powerful spin image descriptor within the self-similar framework.

6.3 Feature Matching

The Feature Matching experiments evaluated the ability to match corresponding features between point clouds and use them to compute the complete misalignment transformation. The features were matched using the Self-Similar Spin Image approach. The same robustness test set of Lidar point clouds were tested by applying the same three transformations - S10, RST, and Y180. The methods were evaluated by computing the position error resulting from the estimated transformation for a randoms sample of points within the reference point cloud.

The simplest transformations Y180 and S10, where only a single parameter was distorted, yielded similar results with both methods able to recover the transformation parameter. The mean percentage error for the Y180 case were 2.60E-11% and 2.88E-11% for the proposed and baseline method, respectively. The mean percentage error for the S10 case were 1.72E-10% and 1.64E-10% for the proposed and baseline method, respectively. Lastly, for the more complex RST transformation, the mean percentage error for the RST case were 18.92% and 33.23% for the proposed and baseline method, respectively. In this case of a more complex misalignment the proposed did a superior job of detecting the transformation and reduced the alignment error by 43%. In all cases, the proposed method was able to detect the transformation change between two point clouds and performed as well or better than the baseline method.

Therefore, for the problem of feature matching the proposed method of Self-Similar Spin Images was able to accurately match the features of the test cases over a variety of point cloud types. In particular, the proposed method was able to outperform its baseline counterpart by exploiting a more powerful spin image descriptor within the self-similar framework.

6.4 Change Detection

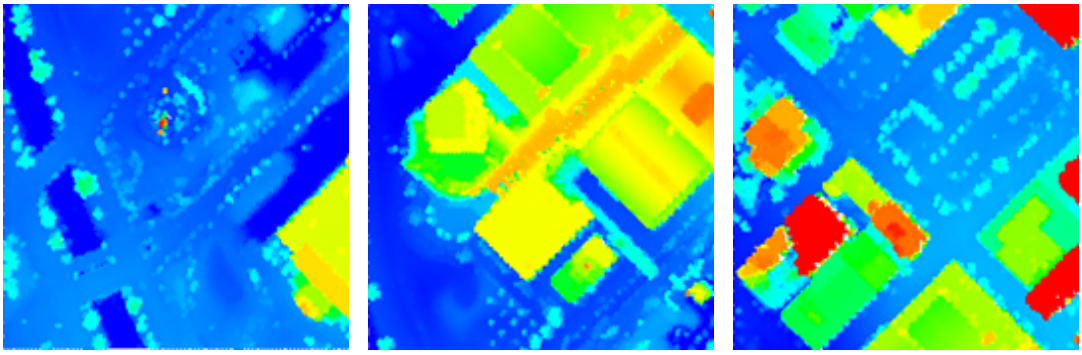
The Change Detection experiments were performed to provide an example application of processing two point clouds aligned using the self-similar based feature matching and registration described in Algorithm 3.5. The experiments evaluated the proposed NNOS method's ability to detect points (present in one cloud but not in the other) and then cluster them together. A reference point cloud was compared with four target point clouds which had two, three, four, then five buildings missing. The proposed method was able to detect all missing buildings in all cases. In addition, it was able to detect a significant roof height change in one building and clustered them separately.

6.5 Future Work

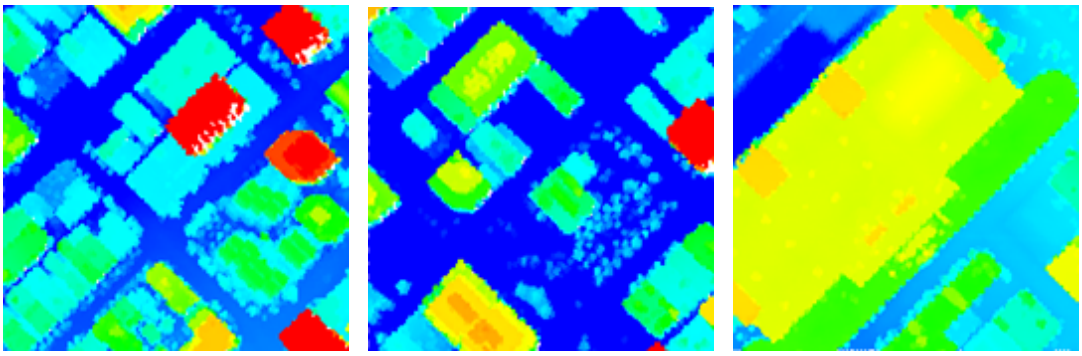
Future work on the Self-Similar Spin Image can incorporate additional metrics into the similarity descriptor in addition to spin images. Potential candidates for additional metrics are the local normal, the local curvature, and photometric intensity. These additional metrics can be combined using a linear combination with prescribed weights to yield a unified similarity [15]. The construction of the similarity surface and its application to feature matching would then follow as detailed in this dissertation.

Another area of new research can be a follow on to the previous proposal of defining a new descriptor by adding new similarity metrics to spin images. Once new metrics have been added and a new descriptor defined then this will also yield a new Self-Similar Keyscale. The same PCA analysis can then be applied to this new descriptor to detect scale differences. In addition, future work can include studying alternative methods of exploratory data analysis other than PCA to analyze the distribution of feature descriptors such as Factor Analysis, Canonical Correlation Analysis, or Independent Component Analysis.

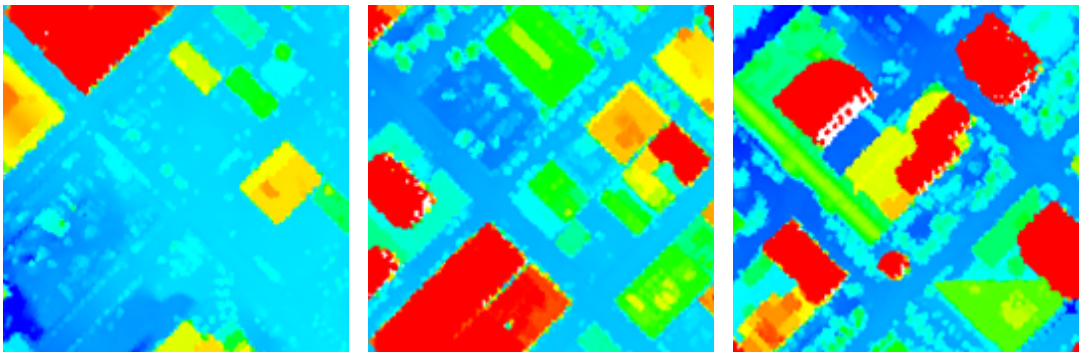
Appendix A: Robustness Test Set



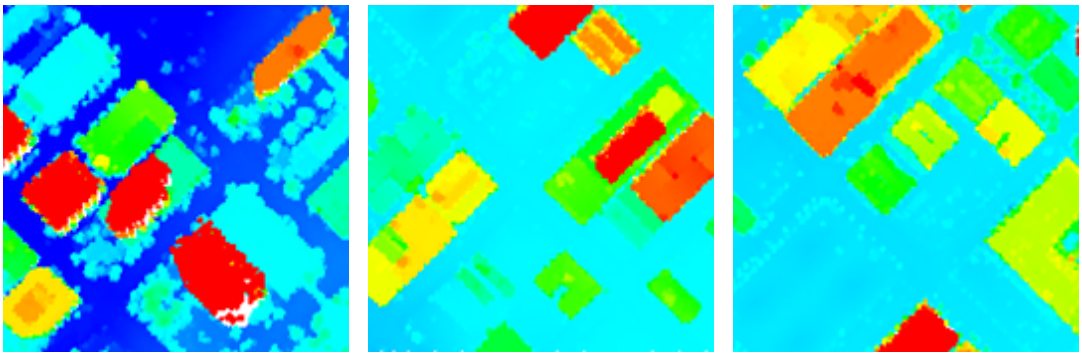
(a) (b) (c)



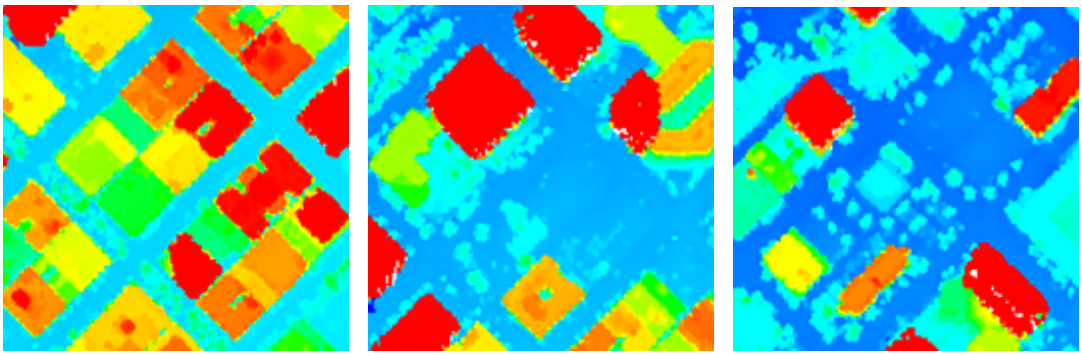
(d) (e) (f)



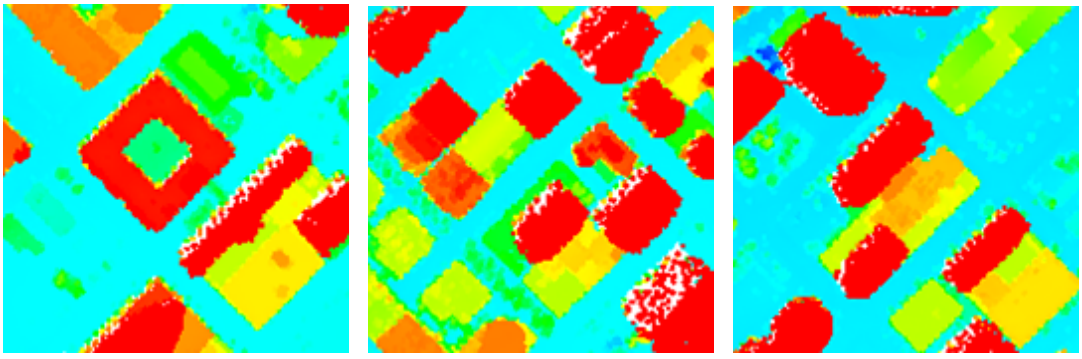
(g) (h) (i)



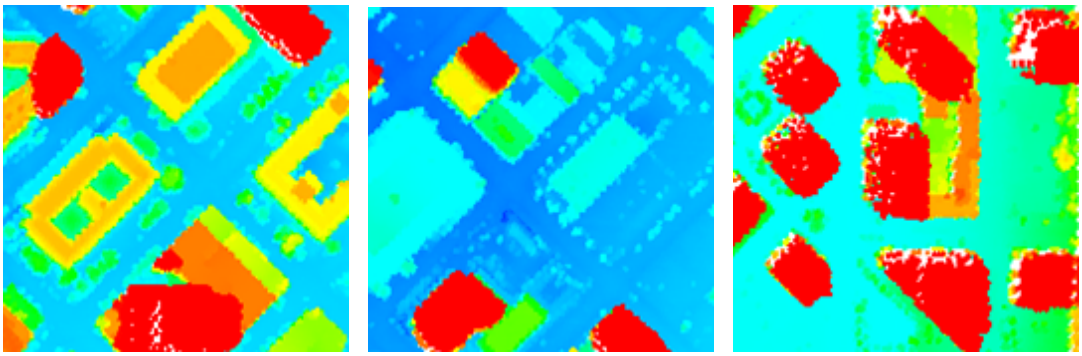
(j) (k) (l)



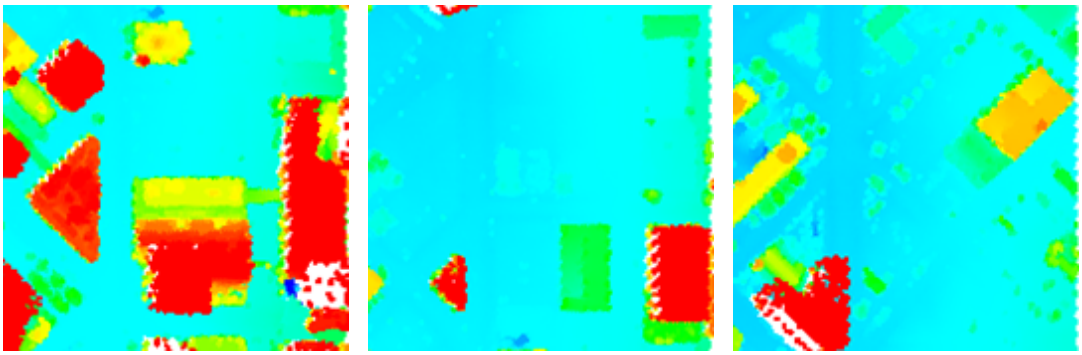
(m) (n) (o)



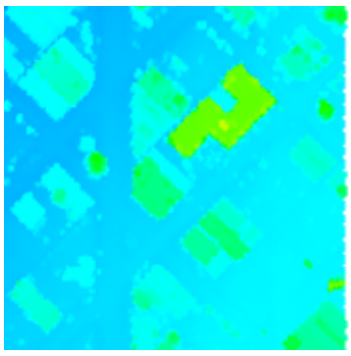
(p) (q) (r)



(s) (t) (u)



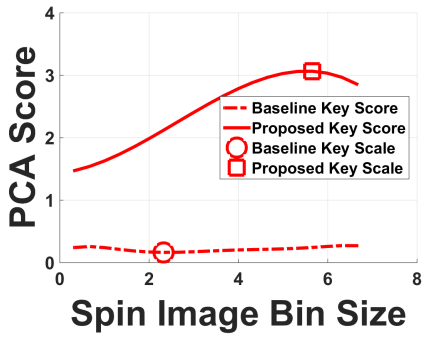
(v) (w) (x)



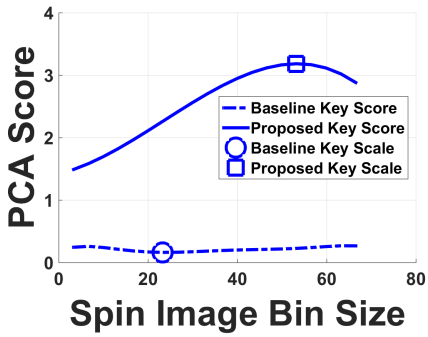
(y)

Figure A.1: Set of Lidar point clouds used for robustness testing. The data sets were selected from an urban area in Denver that covered a variety of building shapes, heights, as well as flat terrains.

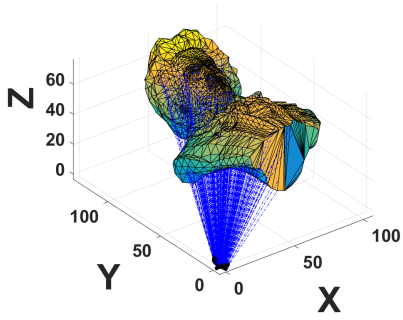
Appendix B: Scale Matching Results



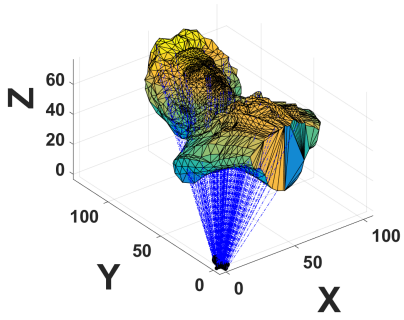
(a) Beethoven PCA scores. Baseline keyscale found at 2.33 and proposed keyscale found at 5.65.



(b) Beethoven (S10) PC scores. Baseline keyscale found at 23.3 and proposed keyscale found at 53.19.

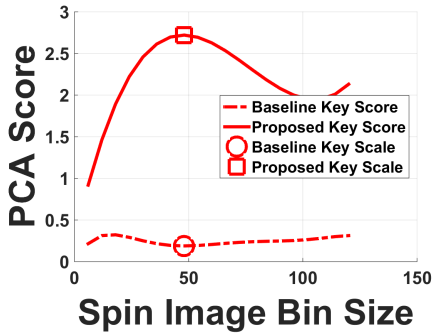


(c) Beethoven matched to Beethoven (S10) with baseline method

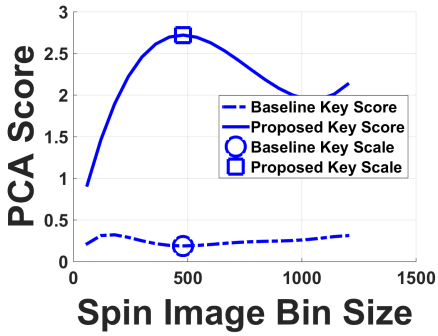


(d) Beethoven matched to Beethoven (S10) with proposed method

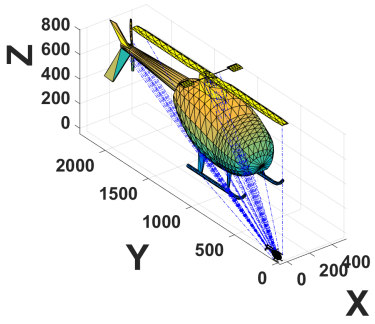
Figure B.1: The scale difference between the Beethoven point cloud and its scaled version was detected by the baseline method and approximately by the proposed method. The ratio of keyscale for the baseline method yielded 10 and for the proposed method was 9.41.



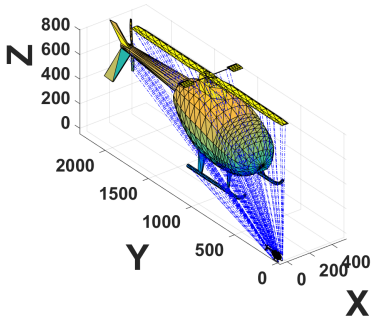
(a) Chopper PCA scores. Both the baseline and proposed keyscale found at 48.01.



(b) Chopper (S10) PC scores. Both the baseline and proposed keyscale found at 480.1.

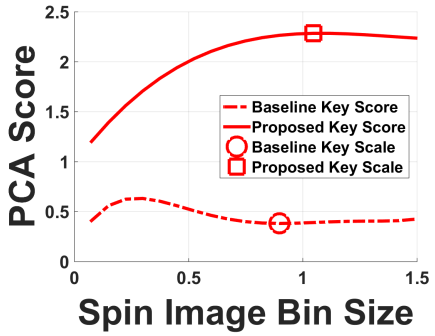


(c) Chopper matched to Chopper (S10) with baseline method

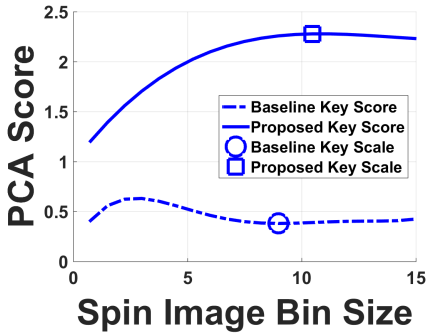


(d) Chopper matched to Chopper (S10) with proposed method

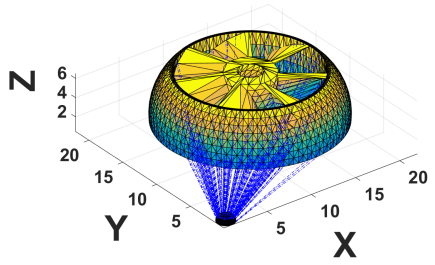
Figure B.2: The scale difference between the Chopper point cloud and its scaled version was detected by both the baseline and proposed methods. The ratio of keyscales for both yields 10.



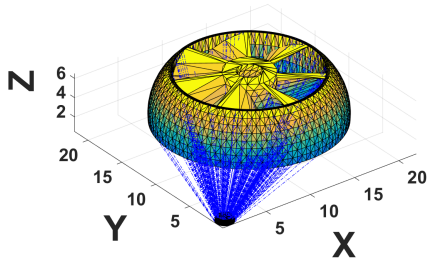
(a) Turbine PCA scores. Baseline keyscale found at 0.897 and proposed keyscale found at 1.046090.



(b) Turbine (S10) PC scores. Baseline keyscale found at 8.97 and proposed keyscale found at 10.460897.

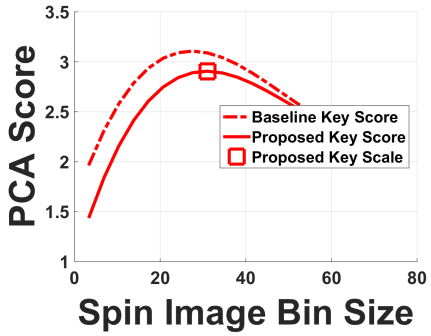


(c) Turbine matched to Beethoven (S10) with baseline method

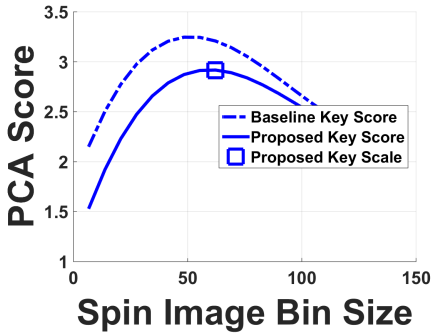


(d) Turbine matched to Beethoven (S10) with proposed method

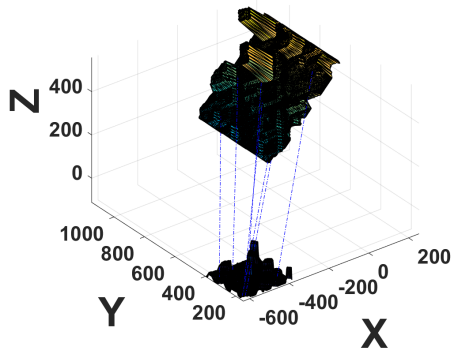
Figure B.3: The scale difference between the Turbine point cloud and its scaled version was detected by the baseline method and approximately by the proposed method. The ratio of keyscales for the baseline method yielded 10 and the proposed method was 9.41.



(a) PC #4 PC scores. Baseline keyscale not found and proposed keyscale found at 31.09.



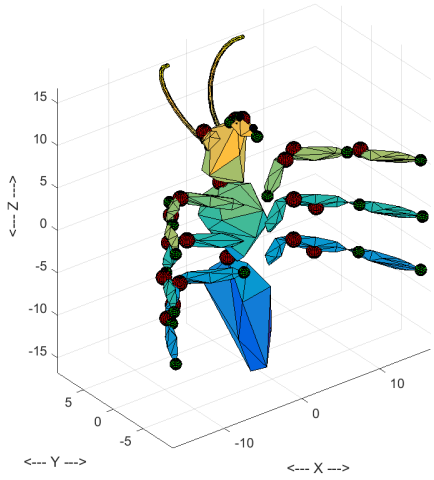
(b) PC #4 (RST) PC scores. Baseline keyscale not found and proposed keyscale found at 62.18.



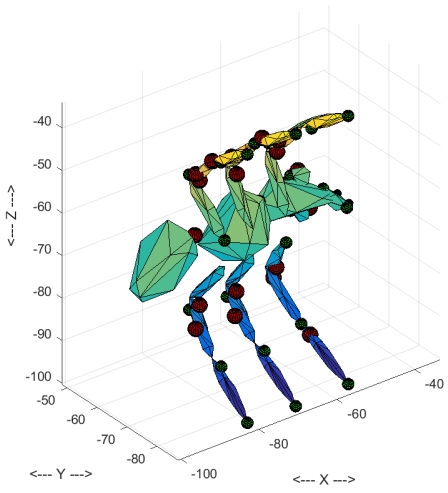
(c) PC #4 matched to PC #4 (RST) with proposed method

Figure B.4: The scale difference between the PC #4 point cloud and its scaled version was detected only by the proposed method which yielded a ratio of keyscales of 2. The baseline method did not have any local minimums.

Appendix C: Feature Matching Results

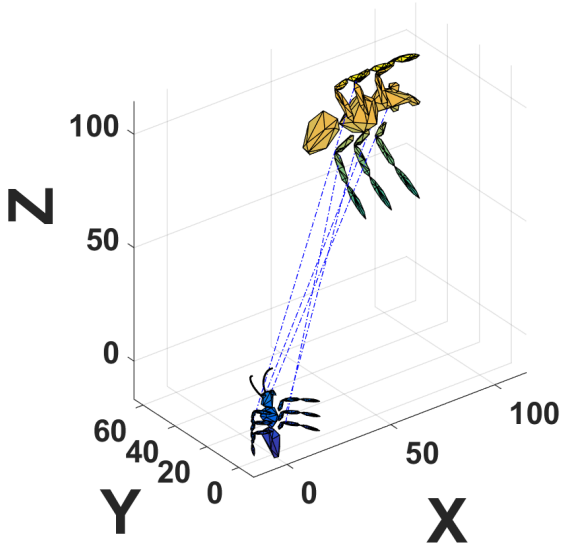


(a) Ant Features

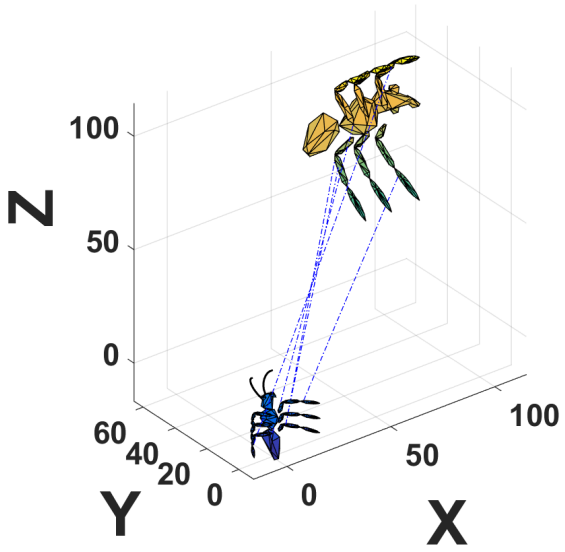


(b) Ant (RST) Features

Figure C.1: Feature points detected using MoPC for Ant (a) and its RST version (b).

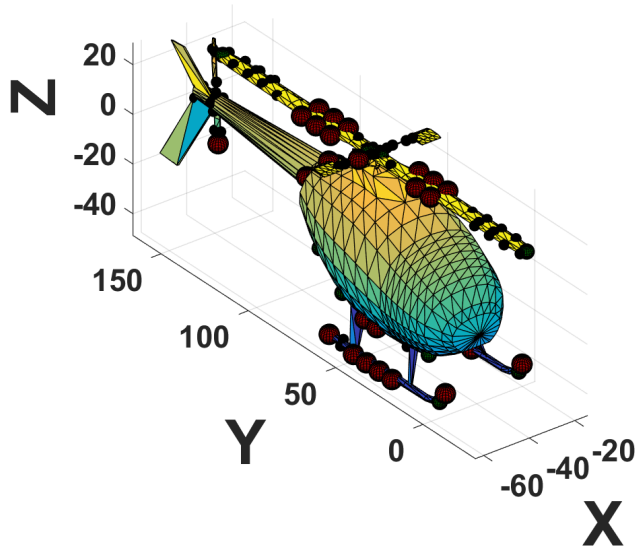


(a) Ant matches with baseline method

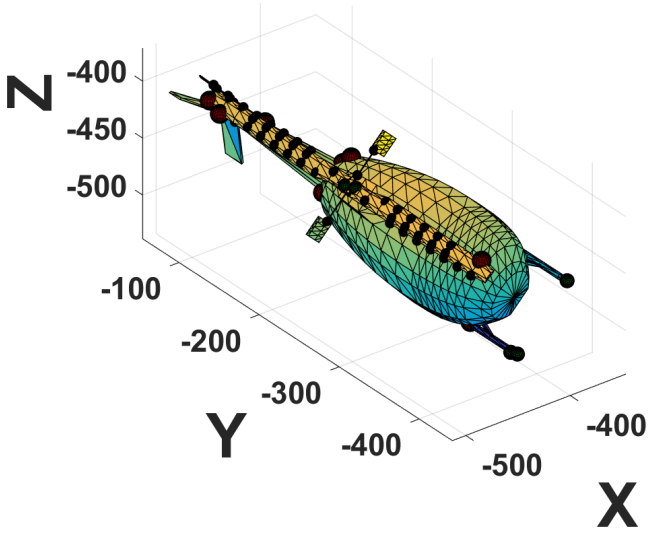


(b) Ant matches with proposed method

Figure C.2: Initial matched features are shown for Ant (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated.

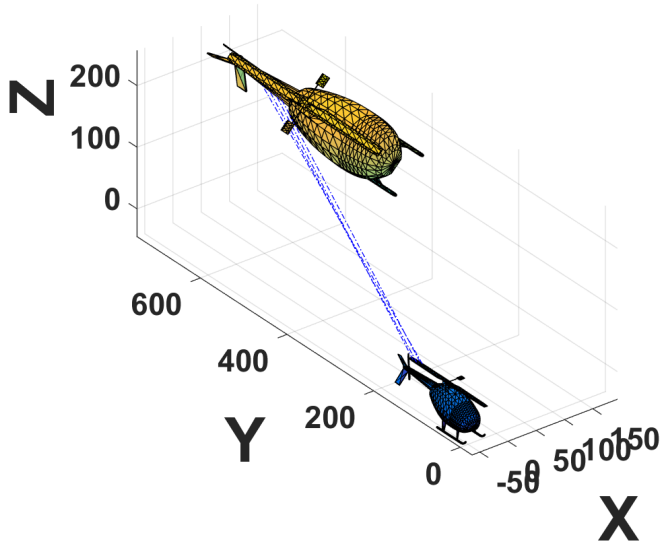


(a) Chopper Features

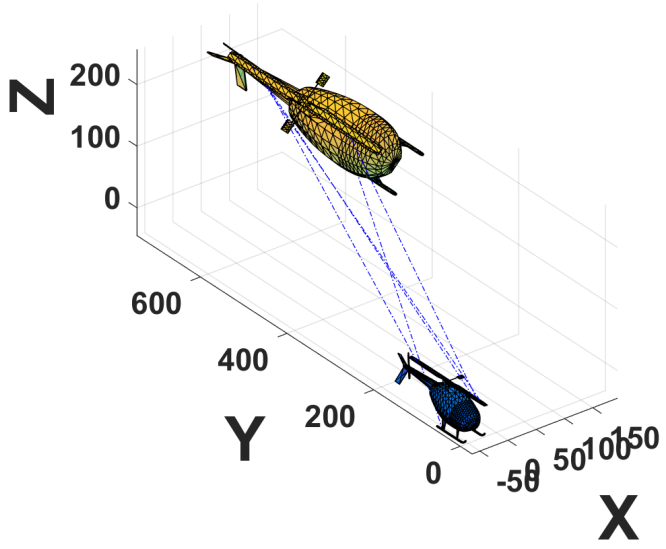


(b) Chopper (RST) Features

Figure C.3: Feature points detected using MoPC for Chopper (a) and its RST version (b).

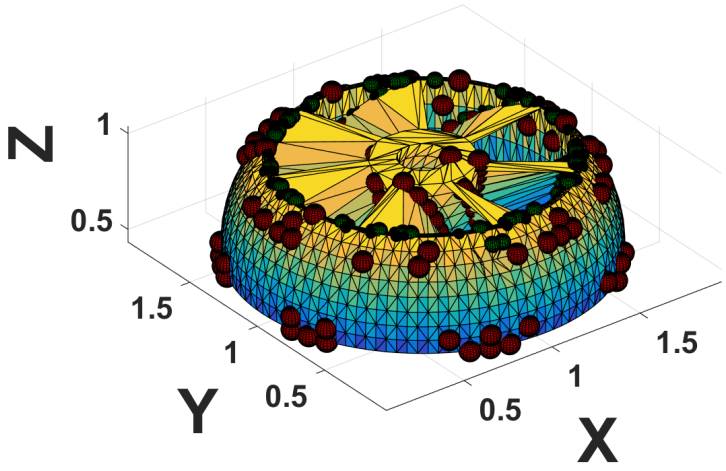


(a) Chopper matches with baseline method

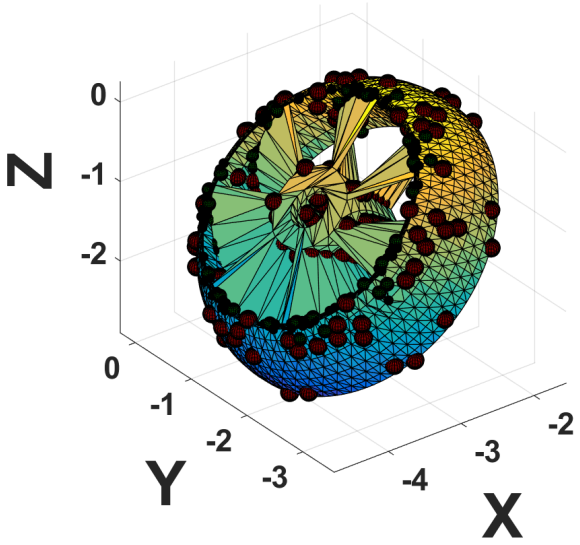


(b) Chopper matches with proposed method

Figure C.4: Initial matched features are shown for Chopper (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated.

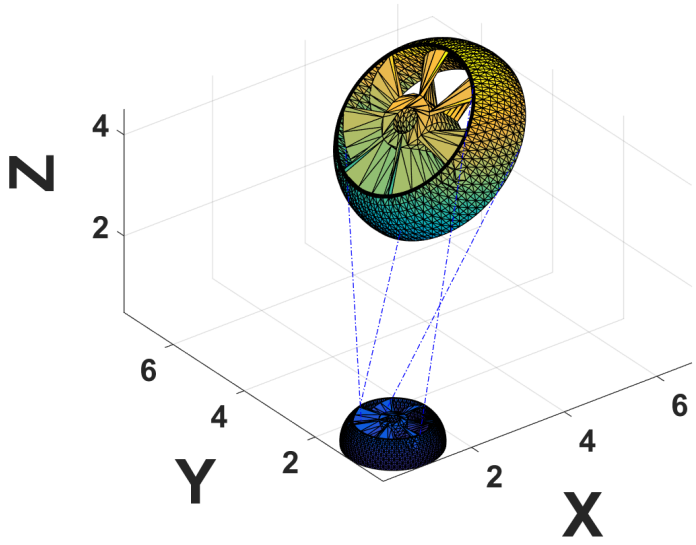


(a) Turbine Features

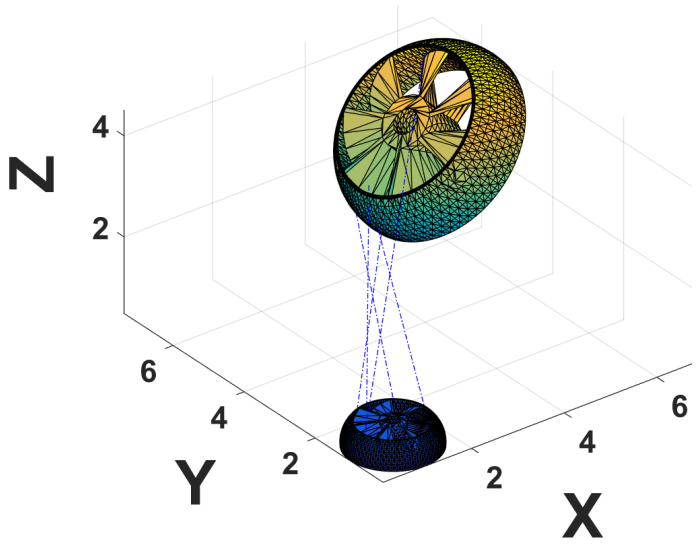


(b) Turbine (RST) Features

Figure C.5: Feature points detected using MoPC for Turbine (a) and its RST version (b).

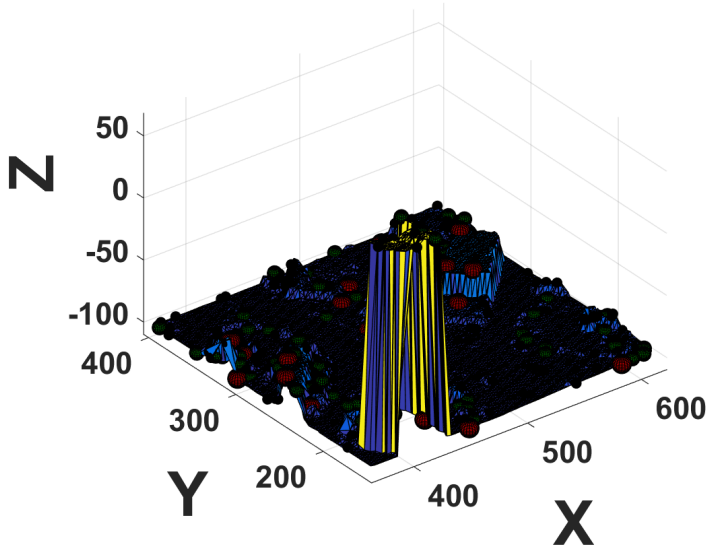


(a) Turbine matches with baseline method

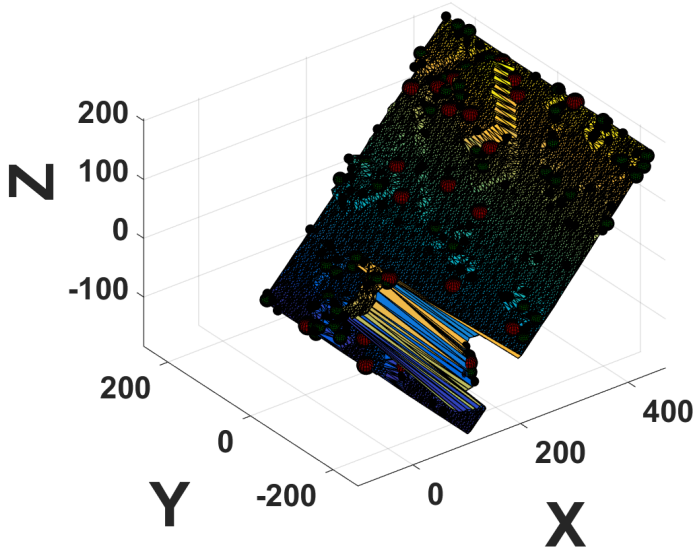


(b) Turbine matches with proposed method

Figure C.6: Initial matched features are shown for Turbine (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated.

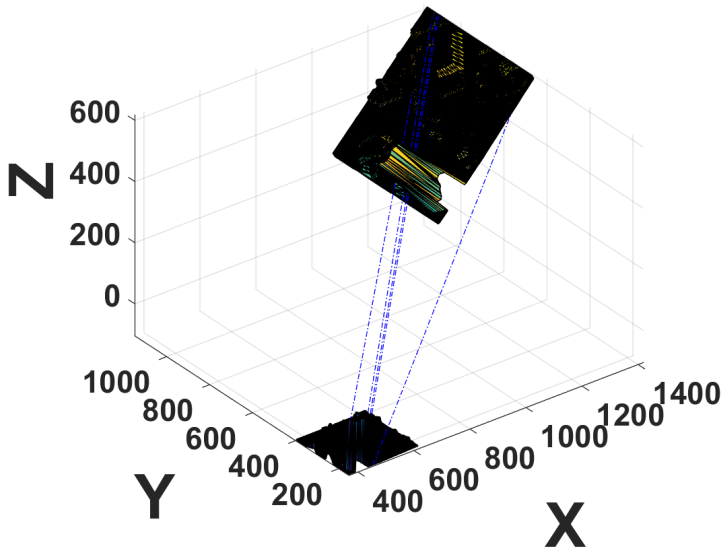


(a) PC-LARGE Features

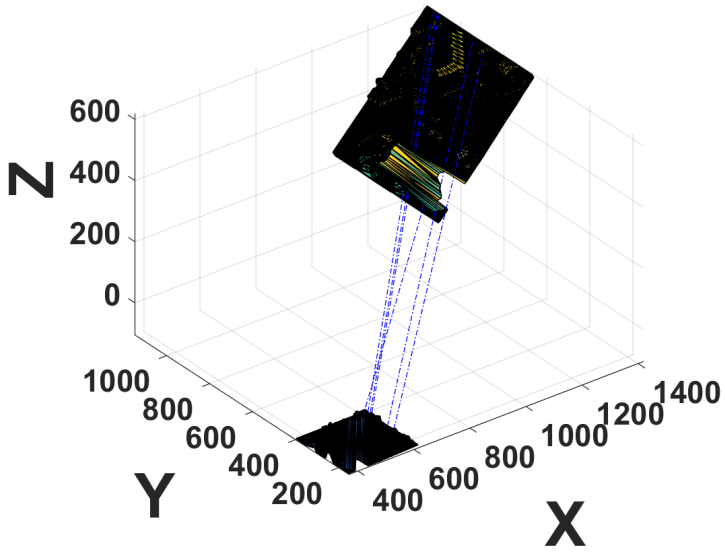


(b) PC-LARGE (RST) Features

Figure C.7: Feature points detected using MoPC for PC-LARGE (a) and its RST version (b).



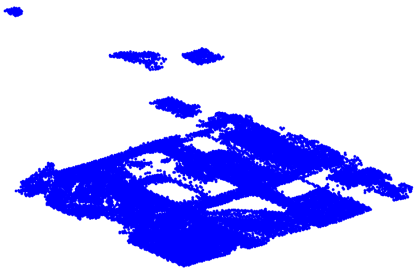
(a) PC-LARGE matches with baseline method



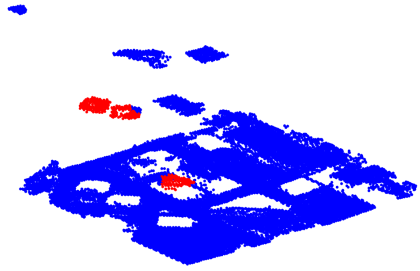
(b) PC-LARGE matches with proposed method

Figure C.8: Initial matched features are shown for PC-LARGE (a) and its RST version (b). Outlier matches are then removed using NNDR and then an affine alignment is calculated.

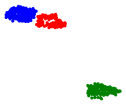
Appendix D: Change Detection Results



(a) PC-TARGET-2 point cloud

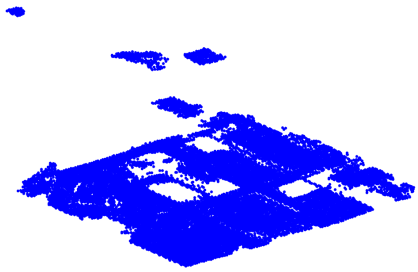


(b) PC-REFERENCE with PC-TARGET-2 changes highlighted

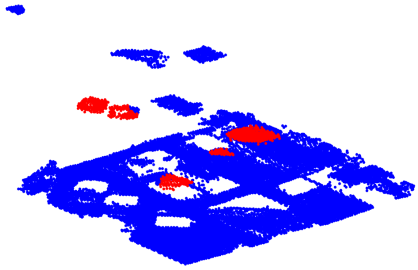


(c) PC-TARGET-2 clusters

Figure D.1: 2D projections of the PC-TARGET-2 point cloud, the expected changes, and the detected clusters of outliers. There are three missing buildings along the left side of the point cloud.



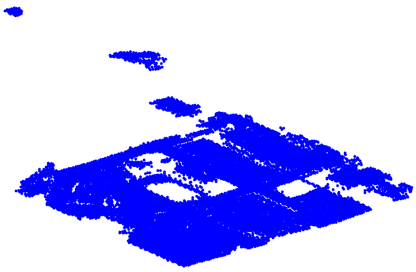
(a) PC-TARGET-3 point cloud



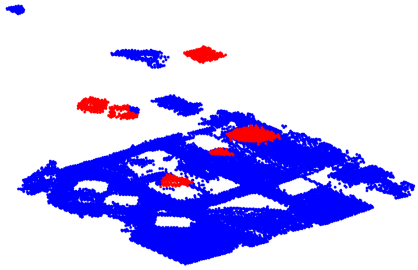
(b) PC-REFERENCE with PC-TARGET-3 changes highlighted

(c) PC-TARGET-3 clusters

Figure D.2: 2D projections of the PC-TARGET-3 point cloud, the expected changes, and the detected clusters of outliers. There are four missing buildings along the left and bottom side of the point cloud.



(a) PC-TARGET-4 point cloud



(b) PC-REFERENCE with PC-TARGET-4 changes highlighted

(c) PC-TARGET-4 clusters

Figure D.3: 2D projections of the PC-TARGET-4 point cloud, the expected changes, and the detected clusters of outliers. There are five missing buildings along the left, bottom, and upper right side of the point cloud.

Appendix E: Publication

The following paper, "**Mining Large Point Clouds for Feature Matching of LIDAR Datasets using Self-Similarity**", was presented at the International Conference on Data Mining (DMIN'17) on July 17, 2017 at Las Vegas, Nevada (ISBN 1-60132-453-7, CSREA Press). It presents a summary of the results detailed in this dissertation for the Self-Similar Spin Image based feature matching algorithm and the Self-Similar Keyscale based scale matching algorithm.

Mining Large Point Clouds for Feature Matching of LIDAR Datasets using Self-Similarity

Daniel Pulido[1], **Anthony Stefanidis**[2]
[1]Leidos Inc., Fairfax, VA, USA
[2]Department of Computational and Data Sciences, George Mason University, Fairfax, VA, USA

Abstract – *LIDAR captured point cloud datasets are another type of big data. Mining such large point clouds for feature selection to perform point cloud matching is a problem of fundamental importance for the geospatial sciences. Standard approaches to point cloud matching face a number of challenges including datasets that exhibit spatial and scale variations. This paper proposes a novel 3D feature descriptor, referred to as "Self-Similar Spin Images", that provides a robust method to perform spatial matching of point clouds by combining the robustness of local self-similarity with the descriptive power of spin images. This descriptor is used to detect the scale difference between point clouds by introducing the "Self-Similar Keyscale" metric. The proposed method was tested with model and LIDAR datasets*.

Keywords: lidar; matching; self-similarity; spin-images; scale

1 Introduction

The rapid growth of Light Detection And Ranging (LIDAR) technologies that collect, process, and disseminate 3D point clouds have allowed for increasingly accurate spatial modeling and analysis of the real world. With millions of points captured per square kilometer, a typical tile of LIDAR data is often of a size of few GBs, and the processing of such datasets represents challenges commonly associated with big data.

LIDAR sensors can generate massive 3D point clouds that provide highly detailed spatial information of an area of interest. Fusing such point clouds allows the generation of broad coverage products that mitigate the particular disadvantages of individual point clouds (e.g. uneven point distribution, small area coverage) and combine their advantages (high accuracy potential). However, this fusion of point clouds collected from a variety of sources can prove challenging.

Pairwise point cloud fusion requires pre-processing to align the two data into a common frame. This common frame may vary from problem to problem, and it can be a common coordinate system, spectral space, or some other abstract reference frame from which the data's features can be compared. A common drawback of such alignment methods is that they often assume knowledge of scale differences in the data. In order to address this challenge, in this paper we present a novel approach to mine features in large point clouds and then use such features to identify scale differences among two point clouds that are to be fused.

1.1 Self-Similarity

Our approach is based on the concept of self-similarity as it can be used to detect prominent features in a point cloud dataset. While self-similarity has featured prominently in fractal theory, and has been applied to computing spatial dimensionality and radar sensor designs, its application to image and point cloud matching is relatively recent. The use of self-similarity has been exploited as a feature descriptor for imagery and video registration [1], detecting deformable shapes [2], and interest point detection [3]. The authors in [4] exploit a self-similarity based feature descriptor to match multi-modal images. In [1], the authors used self-similarity to match objects in an image to hand-drawn templates. Self-similarity was exploited in [5] to match a 2D image to a LIDAR point cloud by using pixel intensities and spatial gradients. Local self-similarity has recently been applied as a local feature descriptor for 3D point cloud matching by Huang [5]. Their approach exploits the local self-similarity of geometric properties of point clouds, principally the local surface normal. In this paper we advance this approach to incorporate Spin Images, a far more descriptive point cloud feature descriptor than the simple local normal metric used in [5].

1.2 Scale Detection

A second component of our approach relates to the assessment of scale variations between two datasets that are to be fused, to ensure that they are indeed comparable ([6], [7], [8]). In [9] a 3D extension of the well-known SIFT descriptor was proposed known as LD-SIFT. Similarly, in [10] a rotational projection statistics (RoPS) multi-scale representation of features is used to perform object recognition in a scale invariant manner. An entirely different approach is to directly detect the scales between two point clouds. In [11] the authors compute a characteristic length for a point cloud, referred to as its "keyscale", that is an optimal scale that best captures point cloud feature descriptors. In our work we extend this keyscale-based process by incorporating self-similar Spin

Images, a more descriptive feature descriptor than the standard Spin Image used in [11].

1.3 Study Description

This study will address the problem of matching two point clouds from potentially different sources. Specifically, we will consider two problems: scale matching and feature matching. Scale matching consists of computing feature metrics of each point cloud and analyzing their distributions to determine scale differences. Feature matching consists of defining local descriptors that are invariant to common dataset distortions (e.g., rotation and translation).

In Section 2 we present the application of local self-similarity to matching point cloud features and the detection of pairwise scale differences. In Section 3 we present experimental results of our proposed approach, and its comparison to a baseline method with model and LIDAR data sets. Finally, Section 4 draws conclusions based on the experimental results.

2 Local Self-Similarity Matching

2.1 Introduction

The notion of local self-similarity is based on the idea of identifying local geometric patterns across data sources regardless of modality. Finding similar features across data sets is based on the notion that "*local internal layouts of self-similarities are shared by these images, even though the patterns generating those self-similarities are not shared by those images*" [1]. That is, by examining how the structure near a feature is self-similar, that self-similarity will also appear in similar features in other data sources. To demonstrate the technique, the authors in [1] were able to match human figures in an image to a hand-drawn stick figures of a human pose.

Specifically, local self-similarity is computed as a correlation surface using a particular local descriptor (e.g., pixel intensity) for a given neighborhood of pixels. This correlation surface is then transformed into a binned log-polar representation to account for local spatial affine deformations. This descriptor is constructed in a manner to ensure it provides a compact representation and mitigates against spatial distortions as well as local non-rigid deformations.

The local self-similarity approach was extended from 2D images to 3D point clouds in [5] by selecting a simple geometric descriptor, the surface normal, to generate the needed correlation surfaces for a given neighborhood of points. The authors also incorporated multiple descriptors in building the self-similarity correlation surface such as curvature and intensity.

In this study, the application of local self-similarity to 3D point features is used to match their respective point clouds in both scale and space. The proposed approach extracts "Self-Similar Spin Image" (SSSI) descriptors, as defined in Section 2.2 at feature locations for both point clouds. The extracted SSSIs are then analyzed using a PCA analysis to compute the "Self-Similar Keyscale" (SSK) metric, as described in Section 2.3, of each point cloud in order to determine their relative scale differences. Once the two point clouds are scale matched, SSSIs that are deemed to be matches define a transformation between the two point clouds which can be used to register them.

Finally, for massive point clouds it can be computationally infeasible to attempt to match every point. Therefore, point clouds must typically be preprocessed using a feature detector to identify points that represent unique features in the scene. The feature detectors in the scientific literature include Harris points [12], Heat Kernels [13], and Mesh SIFT [14], among others. However, the feature detector selected for this study is the "Maximum of Principle Curvature" (MoPC) method as it provides a robust and stable locator of interest regions invariant to rotations and local affine distortions [5]. Figure 1 provides an example of feature points extracted using MoPC.

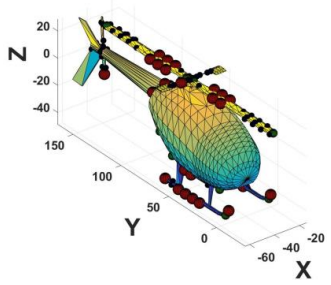


Figure 1: MoPC feature extraction. The marker sizes are proportional to the scale at which the feature was detected.

2.2 Self-Similar Spin Images

The self-similarity framework introduced in [1] computes an attribute (e.g., local normal) for a data point (e.g., 2D pixel or 3D point) and its neighbors. This attribute is then compared against those neighboring attributes using some comparator metric (e.g., correlation) and then normalized. This collection of comparison values defines the self-similarity descriptor.

In this study, it is shown that by using a more powerful descriptor within the self-similarity framework we achieve a more robust method of feature matching. In particular, using a well-established robust descriptor such as Spin Images, instead of just the local normal, can provide a more descriptive correlation surface. The basic approach is to define a neighborhood near each detected feature point, then compute and compare the Spin Image of the feature point against the Spin Image of its neighbors. The comparison is performed by computing a correlation coefficient and then using their values to build a spherical correlation surface.

Therefore, a SSSI is computed follows:

a) Define *C=Point Cloud*, *F = Feature Points*, and *R = Neighborhood Size*.
b) Let $f \in F$ be an extracted feature point
c) Let $N = \{p \mid ||p-f|| \leq R\}$ be a spherical neighborhood of radius R about the feature point that includes all points within that radius (Figure 2)
d) Compute a Spin Image S_n for every point within this neighborhood $n \in N$.
e) Compare the Spin Image of the feature point S_f to every other point's Spin Image in the neighborhood S_n by using a similarity metric, the correlation coefficient, $M_{(fn)} = correlation(S_f, S_n)$.
f) Define a spherical coordinate system as defined in Figure 2 with the origin set at the feature point, the *X* axis is the local principal direction, the *Z* axis is the local normal, and *Y* axis is the cross product of *Z* and *X* to define a right-handed coordinate system.
g) The neighborhood is then spherically binned and each bin is given the value of the average similarity metric $M_{(fn)}$ within that bin.
h) The values in this correlation surface are then normalized to have a maximum of one.

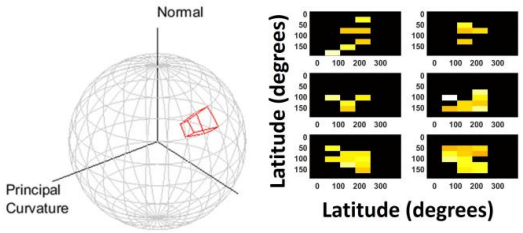


Figure 2: Descriptor local coordinate system and quantization (left). The highlighted region depicts a spherical bin. Example radial slices of the corresponding descriptor (right).

This spherical correlation surface is the desired feature descriptor that can be compared against other features in other point clouds to find a match. Figure 2 also presents six radial slices of the descriptor. For the testing used in this study the number of bins in the radial, longitude, and latitude directions are six, eight, and six, respectively.

2.3 Self-Similar Keyscale

In order for the above descriptor to be truly scale-invariant the size of the spherical neighborhood must be determined. In the approach described in [5] the size of this neighborhood was empirically chosen to be four times "the detected scale at which the principal curvature reaches its local maxima". However, for this method to be scale-invariant and minimize user-interaction a method must be devised to determine the scale at which the point cloud descriptors should be computed. To determine this characteristic size of the point cloud we extend the technique introduced by Tamaki (et al) [11].

The technique relies on computing a "keyscale", which is a characteristic scale by which if two point clouds are scaled by the ratio of their respective keyscales then the point clouds can be matched. A point cloud's keyscale is computed by performing a Principal Component Analysis (PCA) of its Spin Images over a range of scales and finding the scale that yields the minimum cumulative contribution rate. The motivation behind this technique is that for both very small scales and very large scales all Spin Images will tend to look the same, either all representing a plane or a point, respectively. Therefore, it is conjectured that there is an optimal scale in between.

For example, Figure 3 shows a sample point cloud and the same cloud scaled by a factor of 10. The PCA scores for the principal component of both point clouds are displayed in Figure 4. The keyscale for the original Ant point cloud is found to be 6.55 and for the scaled point cloud is 65.5 which yields a keyscale ratio of 10.

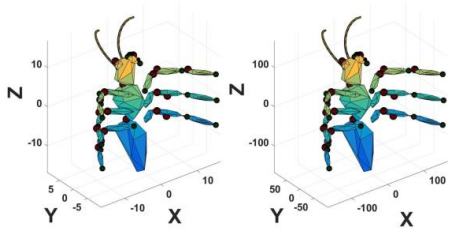


Figure 3: Sample point cloud (left) and its 10x scaled version (right) with their respective extracted features using MoPC.

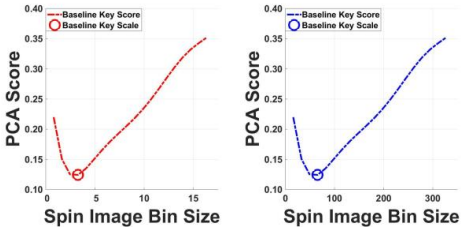


Figure 4: The keyscales of the sample point cloud (left) and its 10x scaled version (right) computed from its PCA score.

In this research we extend this approach to compute the keyscale of a point cloud by performing this multi-scale PCA analysis on the SSSIs. By applying this scale matching technique to a local descriptor that captures the self-similarity of local geometric patterns instead of simply the distribution of neighboring points it is shown that this will provide a more robust scale detector. The basic approach is to compute the SSSI of every feature point for a range of scales and then a perform PCA analysis. The extrema of the principal component yields the keyscale. To compute the SSK:

a) Let R_{min} and R_{max} be minimum and maximum spherical neighborhood sizes, respectively.

b) Let T be the set of neighborhood sizes to be tested in the range $[R_{min}, R_{max}]$ such that the ratio of sequential scales are equal.
c) Let $R \in T$ and compute the spin image S_f for every feature point $f \in F$ with a neighborhood size R.
d) Vectorize each S_f and compute the PCA decomposition of all S_f.
e) Compute the cumulative contribution rates of the PCA bands
f) Repeat for all values of R in T
g) Compute the value of R that maximizes the cumulative contribution rate of the first principal component.

The values for R_{min} and R_{max} must be selected. However, they can be readily chosen based on a characteristic size of the point cloud such that the keyscale algorithm is insensitive to their values. Specifically, we use, ε, the average nearest neighbor distance of all points in the point cloud, as the characteristic size and we set $R_{min} = \varepsilon$ and $R_{max} = 20\varepsilon$.

Figure 5 shows an example of the PCA scores and the highlighted keyscale found at the curve maximum. It should be noted that unlike the standard keyscale approach introduced in [11] the keyscale is not the minimum of the PCA rate curve but for SSK it is the maximum.

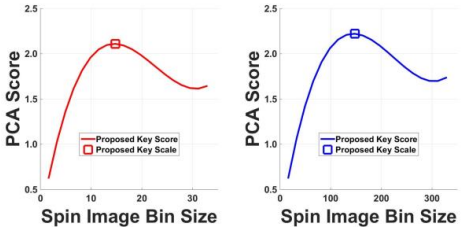


Figure 5: The keyscales of the sample point cloud (left) and its 10x scaled version (right) computed from its PCA score.

The motivation behind this approach is similar to that of the keyscale approach in [11], whereas values of R that are either too small or too large will yield similar flat spin images and so an optimal value in between is expected.

3 Results

3.1 Overview

To evaluate the performance of the proposed algorithms two categories of experiments are performed: Scale Matching and Feature Matching. The purpose of the Scale Matching experiments in Section 3.3 is to evaluate the ability of detecting the scale differences between a reference and target point cloud. Specifically, the reference and target point clouds differ by a Rotation/Scale/Translation (RST) transformation that contains a scale component that is to be computed using the SSK approach defined in Section 2.3. The proposed SSK approach will be compared against the baseline Keyscale approach described in [11]. The purpose of the Feature Matching experiments in Section 3.4 is to evaluate the ability to spatially match the features of a reference and target point cloud. As in the Scale Matching experiments, the reference and target point clouds differ by a RST transformation that must be computed using the SSSI approach defined in Section 2.2. The proposed SSSI approach will be compared against the baseline Self-Similar approach described in [5].

3.2 Test Data

The feature and scale matching methods developed in Section 2.2 and Section 2.3, respectively, were tested against model and LIDAR datasets presented in Figure 6. These datasets were acquired from online academic and government sources (see the Appendix).

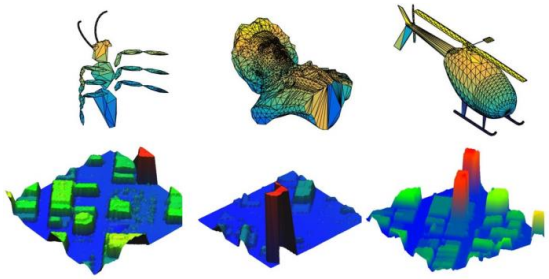


Figure 6: Test Model Point Clouds (Ant, Beethoven, Chopper) and LIDAR (SMALL, LARGE, MIX).

The test data provides a set of point clouds with a varying range of complexity. The model point clouds provide simpler simulated datasets with well-defined feature points. The LIDAR point clouds provide a more realistic data set with the types of typical features found in urban scenes. In addition, to test the proposed and baseline methods for robustness against a wide range of terrains, a set of twenty five LIDAR point clouds were also tested. This robustness test set are presented in the Appendix. The model point clouds contain between 1 and 10 thousand points while the LIDAR point clouds contain approximately 3.5 million points.

Table 1: List of transformations applied to test point clouds.

Label	Rotation	Scale	Translation
Y180	180 deg. about Y	None	None
S10	None	10	None
RST	-45 deg. about Y	2	Max dimension

For the experiments, the test point clouds are transformed using a variety of transformations to test the limits of the proposed algorithms. The transformations and the labels used to identify them are described in Table 1.

3.3 Scale Matching

For the scale matching experiments the test point clouds were initially tested using the S10 transformation. The scale

matching step described in Section 2.3 requires computing a PC analysis of the local descriptors (SSSIs for the proposed method and Self-Similar Normals for the baseline) for all extracted features over multiple scales. This yields a point cloud's keyscale which is defined by the authors in [11] as "*the scale that gives the minimum of cumulative contribution rate of PCA at a specific dimension of eigenspace*".

However, initial experiments using the model point clouds (Figure 7 and Figure 8) showed that using the minimum to define the keyscale proved valid only for the baseline algorithm. For the proposed algorithm using SSSIs the maximum needed to be computed.

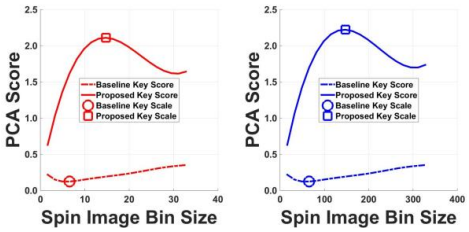


Figure 7: The Ant PCA scores (left) show the baseline keyscale at 6.55 and the proposed keyscale at 14.74 while the respective keyscales for Ant (S10) PC scores (right) were 65.5 and 147.4.

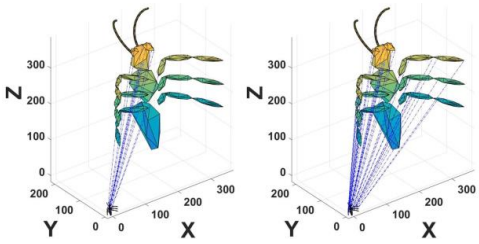


Figure 8: The scale difference between the point cloud (left) and its scaled version (right) was detected by both methods.

Therefore, a more general definition for a point cloud's keyscale would be "*the scale that gives the extremum of cumulative contribution rate of PCA at a specific dimension of eigenspace*".

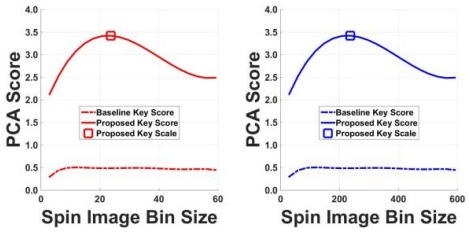


Figure 9: The ratio of the keyscale for the proposed method (23.66 to 236.6) curves yields the correct scale of 10. The baseline method did not yield a keyscale.

Initial experiments for the LIDAR point clouds showed similar results for the proposed algorithm as shown in Figure 9 and Figure 10. However, for the baseline algorithm the PC analysis curves did not have any local minimums regardless of scales selected and, therefore, a keyscale could not be found.

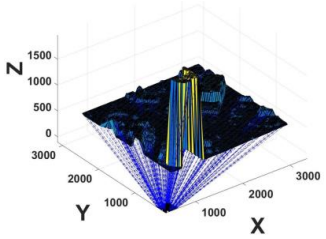


Figure 10: The scale difference between the point cloud and its scaled version was detected only by the proposed method.

Table 2 present the results of performing scale matching on the test dataset. For the simpler model point clouds both the baseline and proposed methods performed well, with the proposed method providing a slightly better estimate of the scale mismatch. For the realistic LIDAR point clouds the baseline method was unable to estimate the scale for any of the point clouds since there was no local minimum in their keyscale curves regardless of the scale ranges tested.

Table 2: Results of performing the scale matching tests.

Point Cloud	Baseline	Proposed	Error	Improve-ment (%)
Ant	10	10	0	0
Beethoven	9.41	10	0.59	5.9
Chopper	10	10	0	0
SMALL	N/A	10	N/A	N/A
LARGE	N/A	10	N/A	N/A
MIX	N/A	10	N/A	N/A

While the initial tests for the baseline method proved it unable to detect the scale difference between the reference and target LIDAR point clouds, tests with the robustness test set showed it able to detect scale difference of some but not all datasets.

Table 3: Results of scale matching on the robustness dataset.

Case	Algorithm	Truth	Avg	Std	Error (%)
S10	Proposed	10	9.98	0.11	0.21
S10	Baseline	10	10.06	0.25	0.64
Y180	Proposed	1	1.01	0.03	0.51
Y180	Baseline	1	0.98	0.07	1.64
RST	Proposed	2	1.96	0.24	1.86
RST	Baseline	2	1.71	0.41	14.41

Finally, in order to evaluate and compare the performance of the proposed and baseline methods for a wide range of terrains and misalignment transformations they were tested against the robustness test set. The results presented in Table 3 show both

the baseline and proposed did a good job of estimating the scale for the simplest transformations S10 and Y180 with errors ranging from 0.21% to 1.64%. However, in both cases the proposed method still reduced the error over that of the baseline by 67.42% for S10 and 68.83% for Y180. For the more complex RST transformation the proposed method, with an error of 1.86%, outperformed the baseline method, with an error of 14.41%, again reducing the error by 87.10%.

3.4 Feature Matching

For the feature matching experiments the test point clouds were tested using the RST transformation. The feature matching step described in Section 2 requires detecting feature points (using the MoPC method), computing a local descriptor for all extracted features (SSSIs for the proposed method and Self-Similar Normal for the baseline), and matching them (using the Nearest Neighbor Distance Ratio method). The baseline and proposed methods were evaluated by comparing their accuracy in estimating the transformation between the reference and target point cloud.

Figure 11 shows the extracted feature points for one of the LIDAR point clouds and its transformed point cloud that were matched.

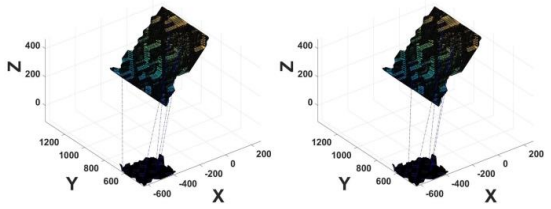


Figure 11: The point cloud and its RST version were matched using the the baseline (left) and proposed method (right).

Table 4: Estimated transformation parameters using both methods for the SMALL LIDAR point cloud.

Parameter	Truth	Baseline	Error (%)	Proposed	Error (%)
Angle	-45	-44.83	-0.37	-44.81	-0.42
X-axis	0	-0.004	-	-0.01	-
Y-axis	1	0.99	-0.02	0.99	-0.03
Z-axis	0	0.02	-	0.02	-
Scale	2.0	1.78	-11.1	2.0	0
X-offset	-249	-322.87	29.67	-255.88	2.76
Y-offset	-249	-224.42	-9.87	-259.35	4.15
Z-offset	-249	-340.34	36.68	-255.75	2.71
Error	-	-	27.94	-	3.29

Table 4 provides the transformation parameters estimated from the matched feature points as well as the true parameter values. The results show that the proposed method was able to provide a better estimate (total error of 3.29%) for the rotation, scale, and translation parameters than the baseline method (total error of 27.94%).

Table 5 presents the total registration error of performing feature matching on the test dataset. For both the simpler model and LIDAR point clouds the proposed methods outperformed the baseline method providing a significantly better estimate of the transformation mismatch with a median improvement of 88%.

Table 5: Total registration error of feature matching.

Point Cloud	Baseline	Proposed	Improvement (%)
Ant	5.11	4.97	-2.79
Beethoven	15.53	0.00	-100.00
Chopper	80.33	27.90	-65.27
SMALL	27.94	3.29	-88.23
LARGE	54.87	6.67	-87.84
MIX	28.88	0.61	-97.89

Finally, in order to evaluate and compare the performance of the proposed and baseline methods for a wide range of terrains and misalignment transformations they were tested against the robustness test set. The results presented in Table 6 show both the baseline and proposed methods did a very good job of estimating the misalignment transformation for the simplest transformations, S10 and Y180, with trivial errors ranging from 2.6E-11% to 1.7E-10%. For the more complex RST transformation the proposed method, with an error of 18.92%, outperformed the baseline method, with an error of 33.23%, reducing the error by 43.08%.

Table 6: Registration errors (%) matching the robustness sets.

Case	Algorithm	Error	Stdev	Improvement
S10	Proposed	1.7E-10	5.5E-11	-4.39
S10	Baseline	1.6E-10	5.4E-11	-
Y180	Proposed	2.6E-11	2.8E-11	9.74
Y180	Baseline	2.9E-11	3.1E-11	-
RST	Proposed	18.92	17.59	43.08
RST	Baseline	33.23	29.94	-

In conclusion, the Scale Matching experiments showed the proposed method outperformed the baseline method by reducing the error in detecting the scale difference by a factor of 67% to 87%. The Feature Matching experiments showed the proposed method outperformed the baseline method by reducing the alignment error by up to 43%.

4 Conclusions

Given the need to mine large 3D point clouds in a variety of fields, this study developed a promising method to align two point clouds with spatial and scale misalignments that only requires processing a subset of the data. To overcome many of the common problems in matching point clouds, this study proposed using a self-similarity based feature descriptor as the basis of its matching process because it can identify local

patterns regardless of how they are generated. Combining this property with a powerful local metric like spin images leads to a robust new feature descriptor: Self-Similar Spin Images.

The distribution of these new descriptors was then analyzed in order to match scale differences between point clouds by defining a Self-Similar Keyscale for each cloud. For both feature and scale matching the proposed methods were able to accurately match the features and scales of the test cases over a variety of point cloud types. In particular, the proposed methods were able to outperform their baseline counterparts by exploiting a more powerful spin image descriptor within the self-similarity framework.

Future work on the SSSI method can incorporate additional metrics into the similarity descriptor in addition to spin images. Potential candidates for additional metrics are the local normal, the local curvature, and photometric intensity. Once new metrics have been added and a new descriptor defined then this will also yield a new SSK.

5 Appendix

The model point clouds are available online at: http://people.sc.fsu.edu/~jburkardt/data/ply/ply.html.

The LIDAR point clouds are available online at: https://earthexplorer.usgs.gov.

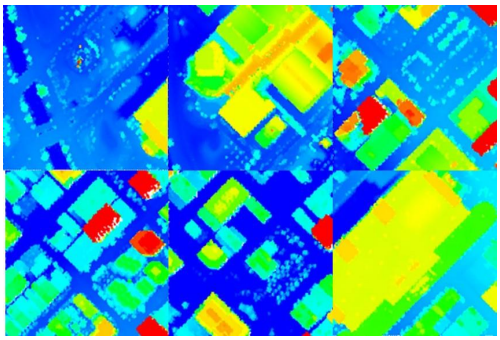


Figure 12: A sample of 6 of the 25 LIDAR point clouds used for robustness testing. The data sets were selected from an urban area in Denver that covered a variety of terrains.

6 Acknowledgement

The first author would like to acknowledge the funding provided by his employer Leidos Inc. that made this dissertation research possible.

7 References

[1] Shechtman, E.; Irani, M. Matching Local Self-Similarities across Images and Videos. IEEE Conference on Computer Vision and Pattern Recognition 2007.

[2] Chatfield, K.; Philbin, J.; Zisserman, A. Efficient Retrieval of Deformable Shape Classes using Local Self-Similarities. Workshop on Non-rigid Shape Analysis and Deformable Image Alignment, ICCV 2009.

[3] Maver, J. Self-Similarity and Points of Interest. IEEE Trans. Pattern Anal. Mach. Intell. 2010, 32, 1211-1226.

[4] Huang, J.; You, S.; Zhao, J. Multimodal image matching using self similarity. IEEE AIPR 20, 1-6.

[5] Huang, J.; You, S. Point cloud matching based on 3D self-similarity. IEEE CVPR Workshops 2012, 41-48.

[6] Stefanidis, A. Using scale space techniques to eliminate scale differences across images. Ohio State University 1993.

[7] Dufournaud, Y.; Schmid, C.; Horaud, R. Image matching with scale adjustment. Computer Vision and Image Understanding 2004, 2, 175-194.

[8] Khaleghi, B.; Baklouti, M.; Karray, F. SILT: Scale-invariant line transform. IEEE CIRA 2009, 78-83.

[9] Darom, T.; Keller, Y. Scale-Invariant Features for 3-D Mesh Models. IEEE Transactions on Image Processing 2012, 21, 2758-2769.

[10] Lu, M.; Guo, Y.; Zhang, J.; Ma, Y.; Lei, Y. Recognizing Objects in 3D Point Clouds with Multi-Scale Local Features. Sensors 2014, 14, 24156.

[11] Tamaki, T.; Tanigawa, S.; Ueno, Y.; Raytchev, B.; Kaneda, K. Scale Matching of 3D Point Clouds by Finding Keyscales with Spin Images. ICPR 2010, 3480-3483.

[12] Mikolajczyk, K.; Schmid, C. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 2005, 10, 1615-1630.

[13] Bronstein, A.; Bronstein, M.; Bustos, B.; Castellani, U.; Crisani, M.; Falcidieno, B.; Guibas, L.; Kokkinos, I.; Murino, V.; Sipiran, I.; Ovsjanikovy, M; Patane, G.; Spagnuolo, M.; Sun, J. SHREC 2010: robust feature detection and description benchmark. Eurographics 2010 Workshop on 3D Object Retrieval (3DOR'10) 20, 79-86.

[14] Boyer, E.; Bronstein, A.M.; Bronstein, M.M.; Bustos, B.; Darom, T.; Horaud, R.; Hotz, I.; Keller, Y.; Keustermans, J.; Kovnatsky, A.; Litman, R.; Reininghaus, J.; Sipiran, I.; Smeets, D.; Suetens, P.; Vandermeulen, D.; Zaharescu, A.; Zobel, V. SHREC 2011: robust feature detection and description benchmark. CoRR 2011.

Bibliography

[1] A. Abedini, M. Hahn, and F. Samadzadegan, "An investigation into the registration of lidar intensity data and aerial images using the sift approach," vol. 37, 01 2008.

[2] C. Toth, H. Ju, and D. Grejner-Brzezinska, *Matching between Different Image Domains*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 37–47. [Online]. Available: https://doi.org/10.1007/978-3-642-24393-6_4

[3] R. M. Palenichka and M. B. Zaremba, "Automatic extraction of control points for the registration of optical satellite and lidar images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 7, pp. 2864–2879, July 2010.

[4] T.-S. Kwak, Y.-I. Kim, K.-Y. Yu, and B.-K. Lee, "Registration of aerial imagery and aerial lidar data using centroids of plane roof surfaces as control information," *KSCE Journal of Civil Engineering*, vol. 10, no. 5, pp. 365–370, Sep 2006. [Online]. Available: https://doi.org/10.1007/BF02830090

[5] J. Lee, C. Lee, and K. Yu, "Autoregistration of high-resolution satellite imagery using lidar intensity data," *KSCE Journal of Civil Engineering*, vol. 15, no. 2, pp. 375–384, Feb 2011. [Online]. Available: https://doi.org/10.1007/s12205-011-1175-z

[6] R. S. Kaminsky, N. Snavely, S. M. Seitz, and R. Szeliski, "Alignment of 3d point clouds to overhead images," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2009, pp. 63–70.

[7] M. Ding, K. Lyngbaek, and A. Zakhor, "Automatic registration of aerial imagery with untextured 3d lidar models," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, pp. 1–8.

[8] A. Wendel, A. Irschara, and H. Bischof, "Automatic alignment of 3d reconstructions using a digital surface model," in *CVPR 2011 WORKSHOPS*, June 2011, pp. 29–36.

[9] C. P. Wang, K. Wilson, and N. Snavely, "Accurate georegistration of point clouds using geographic data," in *2013 International Conference on 3D Vision - 3DV 2013*, June 2013, pp. 33–40.

[10] A. Swart, J. Broere, R. Veltkamp, and R. Tan, "Refined non-rigid registration of a panoramic image sequence to a lidar point cloud," in *Proceedings of the 2011 ISPRS Conference on Photogrammetric Image Analysis*, ser. PIA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 73–84. [Online]. Available: http://dl.acm.org/citation.cfm?id=2050390.2050399

[11] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[12] K. Chatfield, J. Philbin, and A. Zisserman, "Efficient retrieval of deformable shape classes using local self-similarities," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, Sept 2009, pp. 264–271.

[13] J. Maver, "Self-similarity and points of interest," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1211–1226, July 2010.

[14] J. Huang, S. You, and J. Zhao, "Multimodal image matching using self similarity," in *2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Oct 2011, pp. 1–6.

[15] J. Huang and S. You, "Point cloud matching based on 3d self-similarity," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012, pp. 41–48.

[16] F. Schaffalitzky and A. Zisserman, "Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?"," in *Proceedings of the 7th European Conference on Computer Vision-Part I*, ser. ECCV '02. London, UK, UK: Springer-Verlag, 2002, pp. 414–431. [Online]. Available: http://dl.acm.org/citation.cfm?id=645315.649164

[17] T. Tuytelaars and L. Van Gool, "Matching widely separated views based on affine invariant regions," *Int. J. Comput. Vision*, vol. 59, no. 1, pp. 61–85, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000020671.28016.e8

[18] V. Ferrari, T. Tuytelaars, and L. Van Gool, *Simultaneous Object Recognition and Segmentation by Image Exploration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 40–54. [Online]. Available: https://doi.org/10.1007/978-3-540-24670-1_4

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[20] S. Lazebnik, C. Schmid, and J. Ponce, "a Sparse Texture Representation Using Affine-Invariant Neighborhoods Regions," in *International Conference on Computer Vision & Pattern Recognition (CVPR '03)*, vol. 2. Madison, United States: IEEE Computer Society, Jun. 2003, pp. 319–324. [Online]. Available: https://hal.inria.fr/inria-00548232

[21] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, July 2001, pp. 525–531 vol.1.

[22] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 530–535, May 1997.

[23] G. Dorko and C. Schmid, "Selection of scale-invariant parts for object class recognition," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 634–639 vol.1.

[24] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, June 2003, pp. II–264–II–271 vol.2.

[25] B. Leibe and B. Schiele, "Interleaved object categorization and segmentation," in *Proc. BMVC*, 2003, pp. 78.1–78.10, doi:10.5244/C.17.78.

[26] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer, "Weak hypotheses and boosting for generic object detection and recognition." in *ECCV (2)*, ser. Lecture Notes in Computer Science, T. Pajdla and J. Matas, Eds., vol. 3022. Springer, 2004, pp. 71–84. [Online]. Available: http://dblp.uni-trier.de/db/conf/eccv/eccv2004-2.html#OpeltFPA04

[27] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2005.188

[28] A. M. Bronstein, M. M. Bronstein, B. Bustos, U. Castellani, M. Crisani, B. Falcidieno, L. J. Guibas, I. Kokkinos, V. Murino, M. Ovsjanikov, G. Patané, I. Sipiran, M. Spagnuolo, and J. Sun, "Shrec'10 track: Feature detection and description," in *Proceedings of the 3rd Eurographics Conference on 3D Object Retrieval*, ser. 3DOR '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2010, pp. 79–86. [Online]. Available: http://dx.doi.org/10.2312/3DOR/3DOR10/079-086

[29] E. Boyer, A. M. Bronstein, M. M. Bronstein, B. Bustos, T. Darom, R. Horaud, I. Hotz, Y. Keller, J. Keustermans, A. Kovnatsky, R. Litman, J. Reininghaus, I. Sipiran, D. Smeets, P. Suetens, D. Vandermeulen, A. Zaharescu, and V. Zobel, "Shrec 2011: Robust feature detection and description benchmark," pp. 71–78, 2011. [Online]. Available: http://dx.doi.org/10.2312/3DOR/3DOR11/071-078

[30] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[31] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 10 2004, copyright - Kluwer Academic Publishers 2004; Last updated - 2017-05-01. [Online]. Available: https://search-proquest-com.mutex.gmu.edu/docview/1113590426?accountid=14541

[32] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1, pp. 43–72, Nov 2005. [Online]. Available: https://doi.org/10.1007/s11263-005-3848-x

[33] I. Sipiran and B. Bustos, "A robust 3d interest points detector based on harris operator," in *Proceedings of the 3rd Eurographics Conference on 3D Object Retrieval*, ser. 3DOR '10. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association,

2010, pp. 7–14. [Online]. Available: http://dx.doi.org/10.2312/3DOR/3DOR10/007-014

[34] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," in *Proceedings of the Symposium on Geometry Processing*, ser. SGP '09. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2009, pp. 1383–1392. [Online]. Available: http://dl.acm.org/citation.cfm?id=1735603.1735621

[35] U. Castellani, M. Cristani, S. Fantoni, and V. Murino, "Sparse points matching by combining 3d mesh saliency with statistical descriptors," *Computer Graphics Forum*, vol. 27, no. 2, pp. 643–652, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2008.01162.x

[36] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud, "Surface feature detection and description with applications to mesh matching," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 373–380.

[37] C. Maes, T. Fabry, J. Keustermans, D. Smeets, P. Suetens, and D. Vandermeulen, "Feature detection on 3d face surfaces for pose normalisation and recognition," in *2010 Fourth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, Sept 2010, pp. 1–6.

[38] A. E. Johnson, "Spin-images: A representation for 3-d surface matching," Ph.D. dissertation, Carnegie Mellon University, 1997.

[39] K. C. Walli, "Relating multimodal imagery data in 3d," Ph.D. dissertation, Rochester Institute of Technology, 2010.

[40] A. Makadia, A. Patterson, and K. Daniilidis, "Fully automatic registration of 3d point clouds," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, June 2006, pp. 1297–1304.

[41] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[42] M. Greenspan and M. Yurick, "Approximate k-d tree search for efficient icp," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, Oct 2003, pp. 442–448.

[43] T. Jost and H. Hugli, "A multi-resolution icp with heuristic closest point search for fast and robust 3d registration of range images," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, Oct 2003, pp. 427–433.

[44] S. M. Yamany, M. N. Ahmed, E. E. Hemayed, and A. A. Farag, "Novel surface registration using the grid closest point (gcp) transform," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, Oct 1998, pp. 809–813 vol.3.

[45] C. A. Kapoutsis, C. P. Vavoulidis, and I. Pitas, "Morphological techniques in the iterative closest point algorithm," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, vol. 1, Oct 1998, pp. 808–812 vol.1.

[46] d. eggert and s. dalyot, "Octree-based simd strategy for icp registration and alignment of 3d point clouds," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. i-3, pp. 105–110, 2012. [Online]. Available: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/i-3/105/2012/

[47] B. Lin, T. Tamaki, B. Raytchev, K. Kaneda, and K. Ichii, "Scale ratio icp for 3d point clouds with different scales," in *2013 IEEE International Conference on Image Processing*, Sept 2013, pp. 2217–2221.

[48] A. Stefanidis, "Using scale space techniques to eliminate scale differences across images," Ph.D. dissertation, Ohio State University, 1993.

[49] Y. Dufournaud, C. Schmid, and R. Horaud, "Image matching with scale adjustment," *Computer Vision and Image Understanding*, vol. 93, no. 2, pp. 175 – 194, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1077314203001425

[50] B. Khaleghi, M. Baklouti, and F. O. Karray, "Silt: Scale-invariant line transform," in *2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation - (CIRA)*, Dec 2009, pp. 78–83.

[51] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 1150–.

[52] T. Darom and Y. Keller, "Scale-invariant features for 3-d mesh models," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2758–2769, May 2012.

[53] M. Lu, Y. Guo, J. Zhang, Y. Ma, and Y. Lei, "Recognizing objects in 3d point clouds with multi-scale local features," *Sensors*, vol. 14, no. 12, pp. 24 156–24 173, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/12/24156

[54] T. Tamaki, S. Tanigawa, Y. Ueno, B. Raytchev, and K. Kaneda, "Scale matching of 3d point clouds by finding keyscales with spin images," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 3480–3483.

[55] Z. Gao, L. Nocera, and U. Neumann, "Fusing oblique imagery with augmented aerial lidar," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '12. New York, NY, USA: ACM, 2012, pp. 426–429. [Online]. Available: http://doi.acm.org/10.1145/2424321.2424381

[56] C. Berger, M. Voltersen, R. Eckardt, J. Eberle, T. Heyer, N. Salepci, S. Hese, C. Schmullius, J. Tao, S. Auer, R. Bamler, K. Ewald, M. Gartley, J. Jacobson, A. Buswell, Q. Du, and F. Pacifici, "Multi-modal and multi-temporal data fusion: Outcome of the 2012 grss data fusion contest," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 6, no. 3, pp. 1324–1340, June 2013.

[57] O. Boiman and M. Irani, "Detecting irregularities in images and in video," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, Oct 2005, pp. 462–469 Vol. 1.

[58] A. K. Krishnan, S. Saripalli, E. Nissen, and R. Arrowsmith, "3d change detection using low cost aerial imagery," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov 2012, pp. 1–6.

[59] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change detection on points cloud data acquired with a ground laser scanner." in *ISPRS Workshop Laser Scanning*. ISPRS, 2005.

[60] D. Pulido and A. Stefanidis, "Mining large point clouds for feature matching of lidar datasets using self-similarity," in *Proceedings of the 2017 International Conference on Data Mining DMIN'17*, 2017.

[61] M. Borck, R. Palmer, G. West, and T. Tan, "Using depth maps to find interesting regions," in *2014 IEEE REGION 10 SYMPOSIUM*, April 2014, pp. 62–67.

Curriculum Vitae

Daniel Pulido graduated from North Miami Senior High School in Miami, Florida in 1994. He received his Bachelor of Science in Aerospace Engineering from Boston University in 1998. He then worked as an engineer in the field of radar systems for General Dynamics and Riverside Research Inc. In 2003 he graduated from Worcester Polytechnic Institute with a Masters of Science in Applied Mathematics. He has been with Leidos (previously SAIC) since 2006 where he has worked in the area of image science and geospatial software development. He enrolled in the Computational Sciences and Informatics PhD program at George Mason University in 2008 and began working under the supervision of Professor Anthony Stefanidis in 2011.

Education

- Masters of Science, Applied Mathematics, Worcester Polytechnic Institute, 2003
- Bachelor of Science, Aerospace Engineering, Boston University, 1998

Experience

- Senior Software Engineer, Leidos, April 2006 to Present
- Radar Analyst, Magnacom, July 2003 to June 2004
- Member of Research Staff, Riverside Research Institute, July 2000 to July 2001
- Associate Engineer, General Dynamics, September 1998 to June 2000

Publications

- D. Pulido, "Image sharpening toolkit (ISTK)", *Proceedings of the SPIE*, Vol. 7695, pp. 76952F-1-76952F-11 (2010).
- D. Pulido and A. Stefanidis, "Mining Large Point Clouds for Feature Matching of LIDAR Datasets using Self-Similarity", *Proceedings of the 2017 International Conference on Data Mining DMIN'17* (2017).