PCR: APPENDIX-I; PART-II

RAVEPC: Remotely Accessible Visualizer & Explorer of Point Clouds: An Interactive Visualization Application for LiDAR Data

A Report for the Project titled,
"LAN-based Interactive Visualization of
Three-dimensional LiDAR Point Cloud Data",
on the work conducted at IIIT Bangalore,
in August 2013 - February 2015,
funded by NRDMS (DST, GoI).

by

Beena Kumari, Avijit Ashe, Dr. Jaya Sreevalsan Nair *,
Dr. Kiruba Bhagirathi, and Pavithra Rajendran,

Graphics-Visualization-Computing Lab
Center for Data Sciences, IIIT Bangalore.
`http://cds.iiitb.ac.in/gvcl`
* Project PI; e-mail: jnair@iiitb.ac.in

International Institute of Information Technology, Bangalore.
26/C Electronics City, Hosur Road, Bangalore 560100
March 2015

This document is Part II of 2-part series of the report of our project, "LAN-based Interactive Visualization of Three-dimensional LiDAR Point Cloud", submitted to Department of Science and Technology (DST), Government of India. Part I comprises the introduction, background work, literature survey, and our contributions to feature detection and extraction. Part II includes our contributions to feature tracking and remote visualization system.

Acknowledgements

The authors would like to thank the Department of Science and Technology, Government of India, for funding this project under the NRDMS (Natural Resources Data Management System) Programme. The authors would specially like to thank Dr. P. S. Acharya (DST), and Prof. N. L. Sarda (IIT Bombay), from the expert committee, who have supported the project in various ways. Special mention of gratitude to Prof. S. Rajagopalan at IIIT Bangalore for his valuable advice on managing the project. Last but not the least, the project reached its completion owing to the encouragement and help of the administrative staff, faculty, and students at IIIT Bangalore.

Table of Contents

Part I

Part II

List of Figures

List of Tables

Preface

(Salient excerpts from the project proposal)

Project title: "LAN-based Interactive Three-dimensional Visualization of LiDAR Data."
Project objectives: The following are the objectives of our proposal:

- To explore LiDAR for obtaining interesting datasets on 3D campus GIS in order to develop user interactive visualizations and if required, processing them to obtain point datasets.
- To explore LiDAR for obtaining time series datasets on 3D campus GIS in order to develop temporal feature tracking algorithms to enable user interactive visualization and if required, processing them to obtain point datasets.
- To develop various multi-resolution algorithms for performing scalar field topology based analysis on the point datasets and compare with those specified in related work.
- To develop algorithms for temporal feature tracking by used scalar field topology based analysis.
- To implement a visualization system for a server-client architecture for remote visualization, enabling (a) a server with high end graphics card and computational capability to perform data processing and final image computation, (b) one or more clients, which are thin interfaces, functioning as display devices as well as user interaction receiver, and (c) the network between server and client(s) to transport the user feedback and final image data.

Approaches/ methodologies for the work plan: Our approach to implementing the work plan is driven by an end product which is to be deployed on local area network (LAN) of an organization willing to share their LiDAR datasets.

- We will identify such organizations with a need to analyze LiDAR datasets as well as for collaborative analysis across a LAN.
- All algorithms for topological analysis as well as for developing the remote visualization will be implemented as desktop applications.

End-of-project status: Our implementation of the remote visualization tool will include a equal mix of algorithms in computational geometry and visualization, and GPU virtualization technologies.
Suggestions for replicability of the research outcomes: The algorithms, modeling methodologies, and tools that we develop and implement will be made available to both the funding agencies as well as other interested researchers in the community. We will be publishing our findings in leading conferences and journals in related areas.
Suggested plan of action for utilization of expected outputs from the project: The following is our plan of action for utilization of expected outputs from the project:

- We will be hosting our application at an appropriate organization which will have a requirement to analyze and explore LiDAR datasets.
- We will be sharing our source code with researchers in similar areas of interest.
- We will be using the results of our research and implementation to publish at top conferences.
- We will publish at our website an extensive documentation for using our visualization tool.

Publications from the project:
We are in the process of publishing our work in several top tier conferences. We have the following publications so far from this work:

- Kumari, B., and Sreevalsan-Nair, J.,An Interactive Visual Analytic Tool for Semantic Classification of 3D Urban LiDAR Point Cloud (to appear) in Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, Seattle, November 2015.
- Kumari, B., Ashe, A., and Sreevalsan-Nair, J. (2014). "Remote Interactive Visualization of Parallel Implementation of Structural Feature Extraction of Three-dimensional Lidar Point Cloud," In Big Data Analytics (pp. 129-132). Springer International Publishing, December 2014.
- Kumari, B., and Sreevalsan-Nair, J. (2013). "Three-dimensional Visualization of LiDAR Point Cloud Using Structural Feature Extraction," in Proceedings of NSDI (National Spatial Data Infrastructure) 2013 and Poster presentation, IIT Bombay, November 2013.

Chapter 1

Introduction

LiDAR, popularly known as an acronym for Light Detection And Ranging, is actually a portmanteau of the words 'Light' and 'Radar' but as the word 'radar' was already an acronym for Radio Detection And Ranging, people in the scientific community thought it appropriate to label it as an acronym instead. This technology has been around since 1960 when laser was introduced, radar had been used for remote sensing purposes and it was a wise idea of combining these two technical innovations to form another method of remote sensing much more accurate and precise called LiDAR.

Currently, LiDAR is being extensively used in exploration and studying topographic data through the use of airborne altimetry, which is one of the three main type of LiDAR collection techniques, as of now. LiDAR is often available in the form of dense or sparse but a huge collection of unstructured data-set of 3D points over a three dimensional space, where every point has its own coordinates with respect to a given local frame or the world frame of reference. These dense clouds can accurately collect information about even complex structures on the terrain like trees, buildings, coastal belts, mountains etc. at high resolution. As a result, multi-resolution and multi-scale representation of the data is made possible which allows a three-dimensional perspective to studying various salient features associated with them. This further allows the user to decide the amount of information to be included in the final display. Our project revolves around attaining these preliminary objectives.

1.1 Motivation

Since LiDAR are large scale dataset of the order of gigabytes, the processing of LiDAR data become expensive both in terms of time and power for real-time applications. Thus, there is a severe need of cost effective solutions. Secondly, LiDAR datasets can often contain billions of points and most of them may be redundant. We aim to reduce the complexity of data-set by extracting important points, i.e. features without losing the salient information of the data.

Another important requirement is to design a visualization system that not only can be easily deployed into existing infrastructure of any organization but also leverage the advantages of modern technical advancements. We have proposed a server-client based architecture for our visualization system where server

Chapter 5

Feature Tracking

Feature tracking is used to understand, analyze and study the underlying physical phenomenon in the data collected over time. It is used in various field by domain experts and scientists to track, visualize and analyze the specific features to better understand the data. It helps the scientists to locate the interesting time-intervals in the experiments where the behavior of data can be more useful to analyze the data. It can aid to scientist to find the interesting temporal patterns in the data, volume change over time, analyzing the evolution of the data, etc. Feature tracking of time-series data can help us to understand the data both in space and time dimension.

Our idea is to track and visualize the local features present in the time-varying lidar data-sets. Time-varying lidar data means series of data of same targeted object but collected at different time-interval. We are mainly interesting in feature tracking in lidar data-set of buildings. In urban areas, changes can be brought about in the structures of buildings and other artificial monuments due to age, natural calamities or human-made accidents. These changes can be studied automatically when the essential features can be identified and tracked over time. One of the most important cause lies in studying the impact caused due to an earthquake in an urban area. This allows assessing the damages much more quickly, especially when traversing these areas becomes quite impossible just after a disaster. These disasters could be landslides, hurricanes, cloud burst, flood etc. and all of them make terrain traversing very dangerous just after it occurs. Therefore, 3D LiDAR feature tracking gives an efficient alternative to tackle this cause.

Feature tracking involves identifying those salient features in the data which remain persistent with time and identifying them in all the time stamps, of the same data available with us. Once such feature which is persistent in time is detected, we can track them over time through all the time steps and visually see the changes brought about to the original data from time zero through animation. We have classified the data into different feature class as discussed in chapter 8.1 and tracked them using feature IDs. Sometimes, it is useful to understand the behavior of a particular feature over the time. Using feature IDs, user analyze the behavior of a particular feature over the time. Our application allows user to select a particular feature in all available time-stamps of the data and visually track it as animation.

5.1 Algorithm

Feature detection and classification is done for all time-varying data-sets using the algorithm discussed in chapter 8.1. Mainly, we have two feature class after point reduction:- curve- and surface- type. Now next step is to track the particular feature present in all time-stamps of the data. In our application, user can select which type of feature wants to track i.e. curve- or surface- type. After selecting the type, user can iterate over all the features present in that class and manually tagged a particular feature with the help of feature ID in all time-stamp. Tagged features will be tracked by playing them as video. Each tagged feature of a single time-stamp will act as a single frame.

Connected component algorithm has been used to assign feature ID to each line/curve- type features. Line-type features are present in the form of graph and connected component algorithm is used to find all the weakly connected component present in feature graph of line-type features and then feature ID is provided them.

Surface based region growing algorithm is used to cluster the surface feature points into different region based on proximity and continuity of the data and same feature ID has been provided to all points clustered in the same region.

5.1.1 Feature ID for Line-type Features

Line type feature points are extracted in the form of graph as mentioned in chapter 8.1. All points belong to same line/creases/loop are connected together in the graph. We need to find all connected component [2] in the feature graph of line type feature points and assign same feature ID to all points connected together.

5.1.2 Feature ID for Surface-type Features

Surface type feature points are segmented into different cluster based on region growing algorithm [25]. Surface type points are clustered into different region based on proximity, continuity and smoothness constraint. First, consider all surface type points as seed points. Then start region growing with that seed point which has low curvature value. Clustering has been started with low curvature value seed points so that smoothness can be considered for segmentation. Then search the nearest neighbor for each point to connect them to seed points provided they satisfied the criteria for smoothness and proximity constrained.

Clustering has done based on two criteria: 1 Angle between normals and 2 and similar curvature value. Surface normal at any point reflects the underlying geometry of the surface at a given scale. Here, radius value for neighbor search acts as scale. All points belongs to same surface at a given scale will have the less angle between their normals i.e. difference in their normals will be less. The second criteria is curvature value. Curvature value of all neighbored points should be low if they belong to same flat area. Once clustering is done, feature ID has been assigned to clustered region based on increasing order of their size.

Figure 5.1: Surface type feature for three different time-stamps for Autzen-stadium data-set

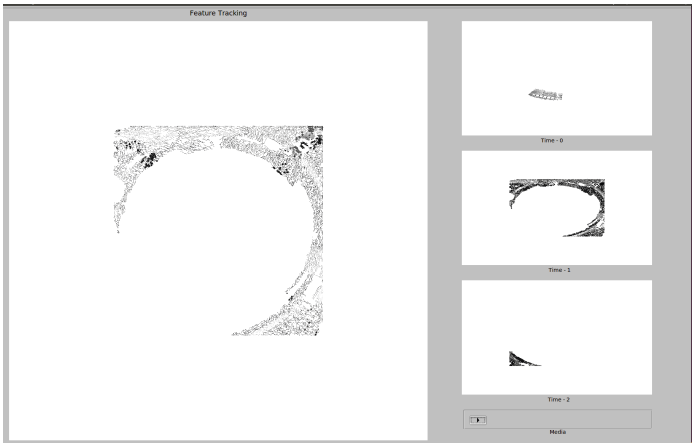

Figure 5.2: Feature tracking of surface type features for Autzen-stadium data-set

5.2 Feature Tracking

Feature tracking has been done in the form of animation. First user has to perform the point reduction for all available time-stamps data. In GUI, various options have been provided to the user for visualization and exploration of time-varying data-set.

For feature tracking, first user has to select the type of feature. There are two options provided to the user - Line- and Surface- type feature. Figure 5.1 show that user has selected the surface type of feature for visualization and tracking purpose. User can iterate over all the feature sets through *next* and *previous* widget present in the GUI. User will manually tagged the feature in all time stamps through *save* widget.

Tagged feature will be track in the form of animated video as shown in Figure 5.2. User can visualize and track the evolution of tagged feature in the form of animation. Video shows how feature has been evolved over the time. User can pause the video at any time to analyze any interesting phenomenon in more details and also reset to start it again. Various other options are also provided in GUI like zoom-in, zoom-out, translation and rotation.

Chapter 6

Remote Visualization

Along with the attainment of the algorithmic objectives, as proposed by the project, another concern has been regarding the implementation of the application on a LAN (Local Area Network). The idea behind this is to configure a high-end system as the server and enable thin-systems as clients. This process is similar to the establishing a remote desktop environment wherein the processing takes place at the server and the thin-clients are used as display terminals. Each user connects to the server with the help of individual credentials, provided by the administrator, and create a isolated session running off the server. The administrator reserves the rights to grant any level of privileges to each user independent of other users, running off the same server simultaneously.

Another goal of such an architecture is also to enable the integration of the new model into our existing systems much more easily, cost-effectively and even making use of low-end systems without having to dispose them off entirely as a result of this establishment.

6.1 Inherent Objectives

In order to fulfill our requirement, it was necessary to first outline the various packages and applications that were used to design the data-modeling, data preprocessing and visualization system. Our application design is entirely open-sourced. OpenGL is used for the graphics support, CUDA programming allowed us to make use of the GPU on board in coherence with the CPU and other packages which are discussed in more details in chapter 7. It was also in our interest to be able expand our generic visualization system by including many other kinds of visualizations like surface, mesh, etc. This creates the possibility to allow user interactive sessions and provides real-time image manipulation and rendering and therefore, a very suitable VNC (Virtual Network Controller) is required to carry the necessary information to and fro over the network. The VNC must possess low latency, carry enough information at relatively lower bandwidth, must create secure sessions and support other general features that are expected for remote visualization system.

Our main objective boils down to designing a set of protocols which enables actual 3D hardware acceleration at the server-side and transport user-interactions and associated image data across the network at low bandwidth, at

low latency and be highly responsive.

6.2 Literature Survey

VirtualGL was found to be the most appropriate alternative to allow OpenGL applications to be ported via a VNC. It is an open source package which gives any Unix/Linux system, mainly a LSB (Linux Standard Base) the ability to run OpenGL applications with full 3D hardware (accelerated graphics) acceleration. Some remote display software like VNC lack this ability and others force OpenGL applications to use software-only OpenGK renderer, which detriments the performance as well as compatibility. Traditionally the method of displaying OpenGL applications was to make use of a remote X server (indirect rendering) supports 3D hardware acceleration but required heavy bandwidth usage because it required all the 3D data to be transmitted over the network to be finally rendered on the client machine. 3D graphics hardware acceleration is used to accelerate our processing and rendering of processed as well as other derivatives of point cloud data and that is why it must always take place on the server, a high-end machine. The approach used until now in many remote display software is plausible only if the data is very limited in amount, there exists a high-speed internet connection and the OpenGL applications are specifically tuned for that remote X-Windows environment. So, basically what happens here, VirtualGl causes the OpenGL commands and the 3D data to be redirected to a 3D graphics accelerator on the application server (high-end machine), and only the rendered 3D images are sent over network to the client machine. It also allows 3D graphics to be shared among multiple users and thus is able to provide "workstation-like" performance out of the thin-clients, even thinner than laptops. It works that great only with the very modest network requirements and thus helps us acknowledge our need of replacing big, noisy, hot-3D workstations with small laptops and hand-held devices. Users can now interact in real-time without having to wait for the 3D data to be copied over the network and get rendered on the client but just visualize what rendering has already been done on the server side.

Among VNCs and their different open-sourced versions available like TigerVNC and many others, we found TigerVNC with the appropriate features to support us. It provided ample security for running sessions without compromising with the high bandwidth and high latency needed for sending encoded data to and fro the network. It used OpenSSH for creating secure sessions and provided further options for including other forms of protection like smart-cards too.

This left us with the choice of procuring some solution that comprises of the above mentioned software bundle. Among the already available remote desktop alternatives, there are more than seventy of them listed in wikipedia, only a few of them delivered this feature set. We settled upon a product called ThinLinc from Cendio AB.

6.3 ThinLinc

ThinLinc is a thin client/terminal server solution. It is a Remote Linux Desktop Server, entirely built on open-source and thus providing extended capabilities

of modifications like feature addition or removal and much more. ThinLinc provides users with centralized remote access to both Windows and Linux desktop applications at the same time, simultaneously. This is equivalent to merging of two different remote desktops into a single one at the client side without letting any kind of inconvenience. The server client architecture of thinlinc is shown in Figure 6.1. It is similar to the experience on working on one's own local computer and is really efficient in delivering a superuser experience, out of the box. ThinLinc is a secure, cost-effective and free to install solution for up to 10 concurrent users.

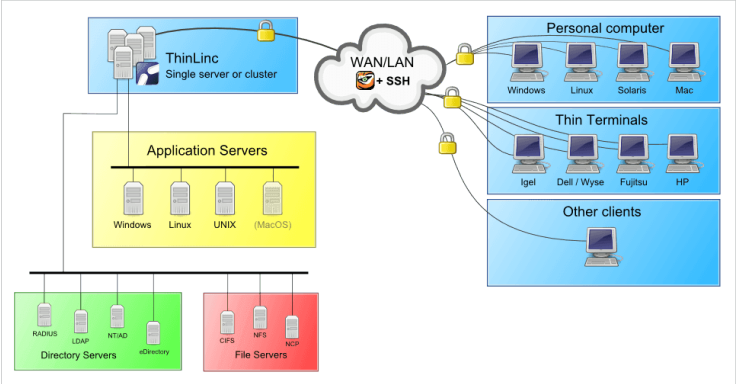

Figure 6.1: Server-Client Architecture [1].

Among verifying its capability, we tested its online demo version available at their own server, which Cendio AB provides with a demo user account upon request. The online demo is a linux server with a few 2D and 3D graphics applications like Remo3D which perform very well even being on a WAN. It was also tested with HD video rendering of sample video on the file system and through the browser of the remote desktop, which also went quite satisfactorily. There are a couple of tests like hardware isolation, software isolation etc. carried out to judge its capabilities that are listed in our user manual. We also went through the various case studies, available at their website, of their successful customers that include companies like ANSYS, SAAB, VOLVO, Konsberg and even many institutes from various corners of the world. In National Super Computer Center of Link-oping University, sweden, thinlinc is installed at their supercomputer to provide remote access capabilities particularly for driving graphics rich applications out of the server with least performance overhead and network saliency. Triolith is their current flagship with computational capability of 407 tera flops/sec. It uses HP's cluster platform 3000 with SL230s Gen8 computing nodes. These evidences were relevant enough to start testing the application for our specific requirement. The essential features of this product are mentioned below.

6.3.1 Desktop Virtualization Solution

Our application will be installed on the server and data will also be resides on server instead of thousands of PCs and workstations. These workstations can be now replaced with thinner clients like laptops or old desktops that are

less expensive, simpler in configuration and most importantly easier to manage without the help of any dedicated management department. It simplifies hardware and software requirements and their maintainability, which may include frequent backups, upgrades, updates and all kinds of technical support and services. These are economically cheaper alternatives.

It allows administrators to deploy, manage and support desktops and applications much more easily from the comfort of a single server room. It improves device independence as users can access their desktops from practically any thin client like smart-phones and tablets. Remote access can be granted both in an internal local area network or over internet which is a wide area network with negligible degradation in performance. ThinLinc uses advanced lossless data compression methods for sending screen updates between server-client pairs and performs much better than any other web application(it was tested).

6.3.2 Enhanced Security

Security increases bandwidth requirements and slows down data transfer but thinlinc handles this issue very carefully with implementing OpenSSH connections for every session. Thus, minimizes security risks using client authentication of the server protecting against man-in-the-middle attacks. It employs smart cards support, public key authentication and one-time-passwords. It also offers single-sign-on functionality, a premium facility provided by many expensive service providers, with full security integration with even Novell Directory, Microsoft Active Directory and RADIUS. This allows telecommunication among users without risks involved.

As sensitive data can be lost or corrupted or even fall in wrong hands, security increases as it resides on locked down servers instead of your device. Therefore, it is less prone to attacks and viruses as only few central servers need to checked and scrutinized regularly, which is far easier and less time consuming.

6.3.3 Cost Reduction

ThinLinc can be easily installed on a wide range of LSB distributions. The network is always divided into master servers and agent servers wherein a LSB is always the master and other Windows ones are agents handled by masters alone. One major cost reduction is due to reuse of old desktops and thinner clients. This has been reported by the studies conducted by the Fraunhofer Institute.

Reduction in Opex Cost: It involves cost of time and effort involved and also the cost of operation and maintenance.

Reduction of Capex Cost: It involves cost of hardware and software in general terms.

6.4 Installation Procedure

Please follow user-manual to install thinlinc and ravepc.

Chapter 7

System Architecture

RAVEPC is an open source application for exploration of 3D point cloud. It is specifically designed for airborne lidar data. It is developed using open-source libraries and packages. Figure 7.1 shows the system architecture of RAVEPC and it can be seen that the input/output mechanism of the application can be presented in a three-tier layout showing various operations performed by various compiled packages.

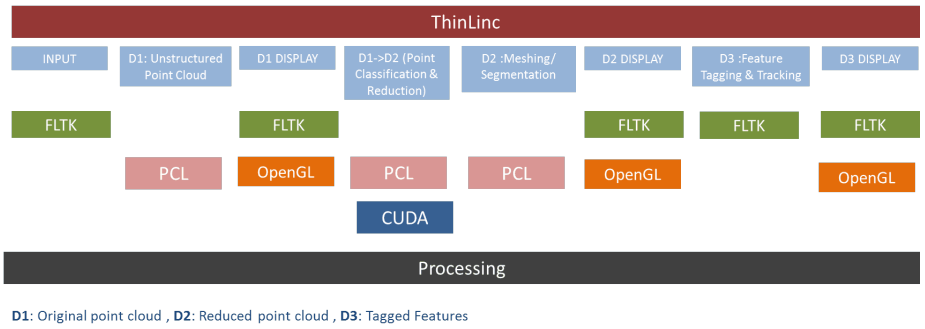

Figure 7.1: System Architecture

The main goal of the RAVEPC is to remotely explore the data such that only one machine with dedicated hardware is required to perform the computational operations and other thin-devices can be used for display purpose. The entire processing system of RAVEPC has been segregated into three modules: a server performs all the back-end operations, a thin-client performs all the front-end operations and forms a part of the user interface, and finally the transporting mechanism that takes care of the communication between the front-end and the back-end, forms the the mid-end channel as shown in the Figure 7.2.

The idea behind this was to configure a high-end system as the server and enable thin-systems as its clients. Each user connects to the server with the help of individual credentials, provided by the administrator, and create an isolated session with privileges granted to users independent of each other, running off the same server simultaneously.

Another goal is the integration of a new model into our existing systems much

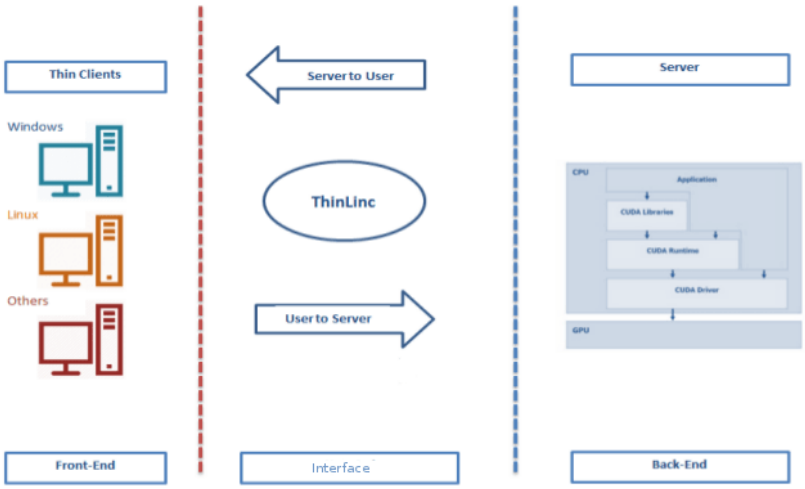

Figure 7.2: System Design

more easily, cost-effectively and even making use of low-end systems effectively without having to dispose them off entirely as a result of this up-gradation.

7.1 Back-End Computing

In the Back-end, RAVEPC will be installed on the server to perform all the computational operation. Only those files which resides inside the server can be visualized and analyzed by the user. All the computational operations like file reading, outliers removal, point reduction, segmentation and mesh construction will be performed on the server. Data for final image will be transferred to the client.

7.2 Transporting mechanism

Our application needs user interactive sessions, real-time data manipulation operations and 3D rendering over the network. Therefore, a very efficient VNC (Virtual Network Controller) was needed. TigerVNC was the most suitable protocol for this endeavor. Thus, we needed a set of protocols that enables actual 3D hardware acceleration at the server-side and transport user-interactions and associated image data across the network at low bandwidth, low latency and be highly responsive as shown in the Figure 7.3. Our application is developed for linux platforms and OpenGL is used for all graphics-related operations like 3D rendering, etc. VirtualGL has been found to be the most appropriate alternative to allow OpenGL applications to be ported via a VNC. It is an open source package which gives any Unix/Linux system, mainly a LSB (Linux Standard Base) the ability to run OpenGL applications with full 3D hardware acceleration. ThinLinc is used to establish the communication between server and client Which can work very well in conjunction with VirtualGL packages.

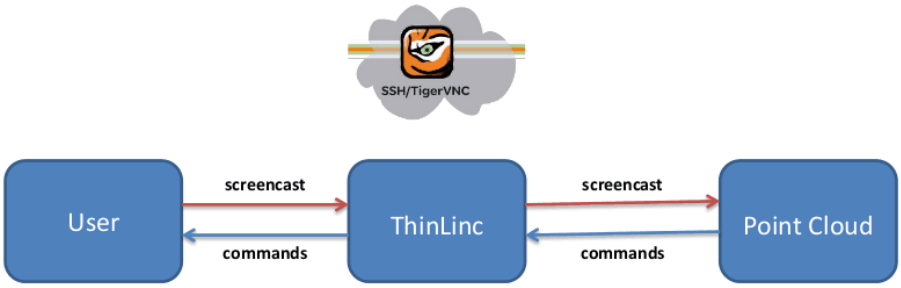

Figure 7.3: User-Interactivity Over Network

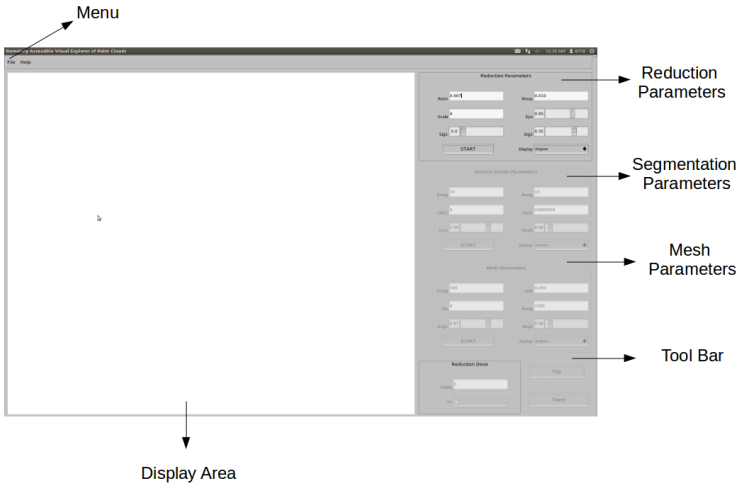

Figure 7.4: RAVEPC user interface for analysis of point cloud data-set

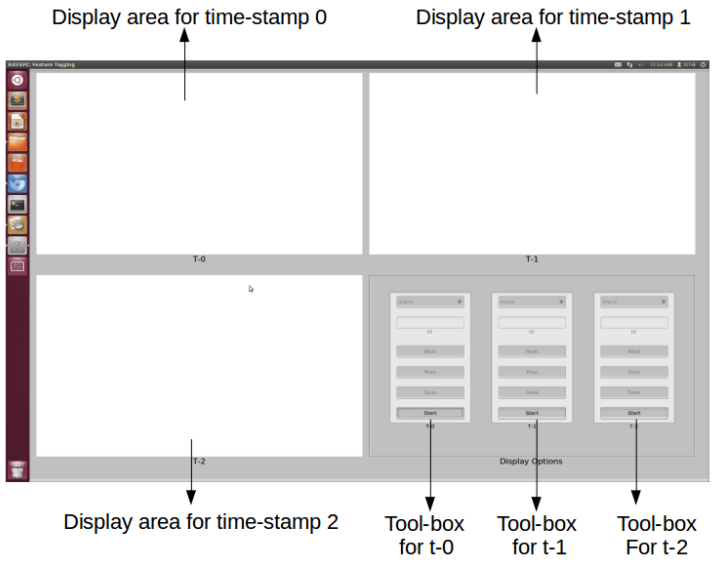

Figure 7.5: RAVEPC user interface for feature tagging in time-varying point cloud data-set

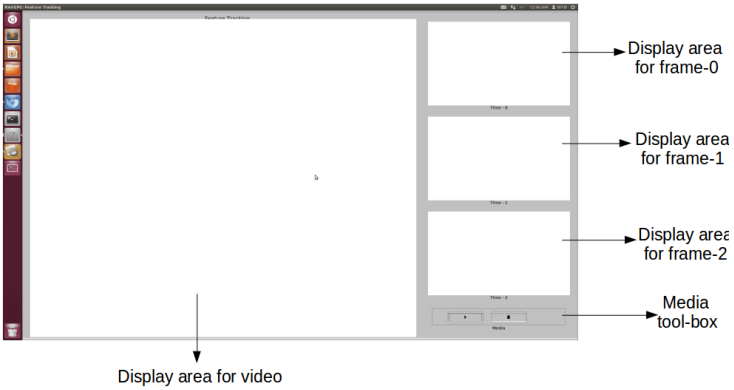

Figure 7.6: RAVEPC user interface for feature tracking in time-varying point cloud data-set

7.3 Front-End Visualization

Front-end visualization is divided into two parts: RAVEPC UI and ThinLinc client UI.

7.3.1 RAVEPC User Interface

RAVEPC consists three user interface, first one for exploration and analysis of point cloud, second for visualization and tagging of features in time-varying point cloud and third one for feature tracking as given in the Figures 7.4, 7.5 7.6. For more details on user interface (UI), please refer user manual.

7.3.2 ThinLinc Desktop and Web Client

ThinLinc client desktop is required to connect to the server. In chapter 6, a detailed explanation has been provided on various thinlinc clients available for various platforms including Mac and Windows.

Chapter 8

Experiments & Discussions

In this chapter, we demonstrate the output of our application and its performance by testing the algorithm on various data-sets. We have used Keller et al. [15] to analyze the structural characteristics of LiDAR point cloud and extract the important features likes edges, corners, crest lines, etc. This algorithm is driven by user-defined parameters. To get the correct output, user needs to fine tune these parameters. All the experiments which we have conducted are performed on computer with Intel Xeon(R) processor at 3.2GHz quad-core, 8 GB RAM with NVIDIA GeForce GTX480. Our algorithm mainly consists of four major blocks:

- Point Classification and Reduction
- Segmentation
- Mesh Construction
- Feature Tracking

Except feature tracking, other modules needs various user-defined parameter. In feature tracking, user has to manually tag the features and then tagged features will be displayed as animation. Its output depends upon the point reduction and segmentation module. We will not discuss the feature tracking module here as it is already explained in detail in the chapter 5. Default values and expected range are given for all the parameters in the application. User can fine tune these values if he/she is not satisfied with the output. Point classification and reduction, segmentation and mesh construction are given in the subsequent section.

8.1 Point Classification and Reduction

We have discussed theory part in the chapter. Here we will talk about the results. Table 8.1 gives the classification of the points in each of the data to points with highest likelihood of having curved (cylindrical) or planar (disc-like) shape local structure, and their respective critical points, with spherical shape local structure. Table 8.2 shows the statistics for point reduction. It can be noted that more planar the data set, the reduction of the point data set will be

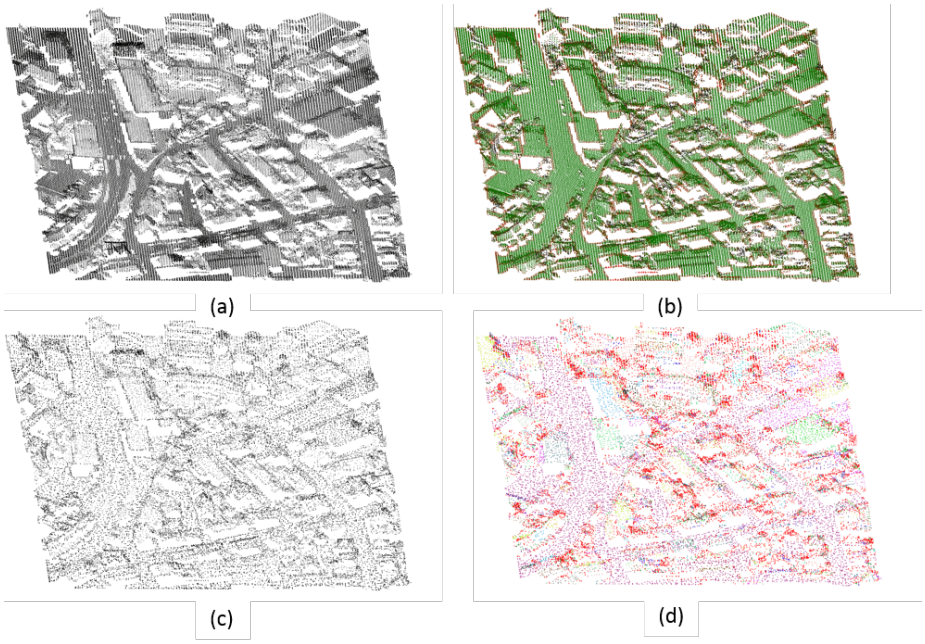

Figure 8.1: Point Classification, Down-Sampling and Segmentation of area 1 of Vaihingen data-set: (a) Original Points, (b) Saliency Map, (c) Reduced Points, and (d) Segmentation. In Figure (b), Green shows surface-type points, Blue shows critical-type points and Red shows line-type points. In Figure(d), Colors are randomly assigned to different clusters. Figure (a) contains 1,70,087 which is reduced to 42,247 in Figure (c).

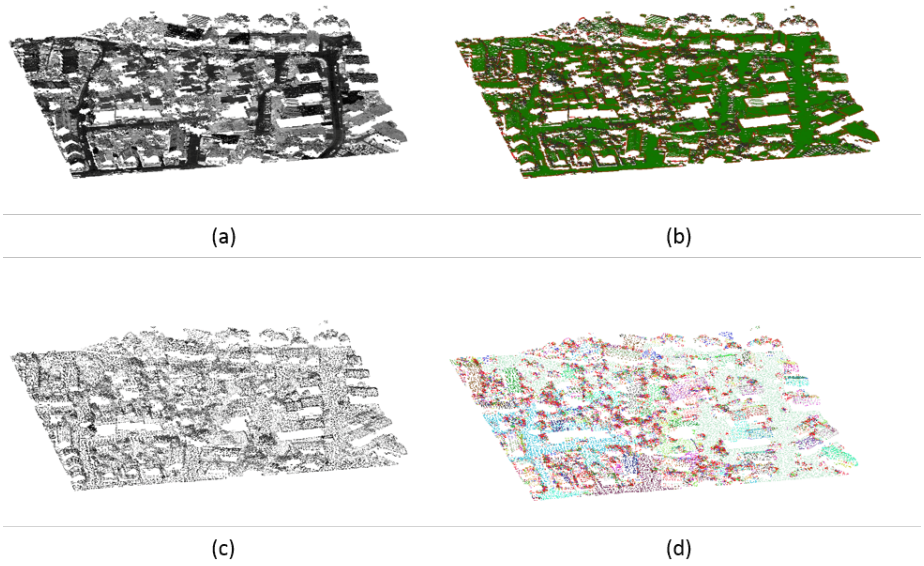

Figure 8.2: Point Classification, Down-Sampling and Segmentation of area 1 of Vaihingen data-set: (a) Original Points, (b) Saliency Map, (c) Reduced Points, and (d) Segmentation. In Figure (b), Green shows surface-type points, Blue shows critical-type points and Red shows line-type points. In Figure(d), Colors are randomly assigned to different clusters. Figure (a) contains 2,33,782 which is reduced to 61,915 in Figure (c).

higher. Figure 8.1, 8.2, 8.3 and 8.4 show output for area 1, area 2 and area 3 site of vaihingen data-set, and autzen stadium respectively.

Original Dataset		Point Classification			
	# Points (Total)	# Points (Curved)	# Points (Planar)	# Critical Points (Curved)	# Critical Points (Planar)
Area 1	1,70,087	39,184	1,63,898	3	397
Area 2	2,33,782	51,598	2,20,766	17	1,129
Area 3	1,99,892	43,201	1,91,497	14	707
Stadium	6,93,894	28,906	6,83,199	16	76

Table 8.1: Classification of point cloud data to points with highest likelihood for a curve (cylindrical) or a planar (disc-like) neighborhoods and critical points using Keller et al.'s algorithm [15].

Dataset	# Original points	# Reduced points	# % reduction
Area 1	1,70,087	42,247	75.1
Area 2	2,33,782	61,915	73.5
Area 3	1,99,892	51,811	74.1
Stadium	6,93,894	99,164	85.7

Table 8.2: Points Reduction for area 1, area 2, area 3 of vaihingen data-set [7] and autzen stadium [4]las data-sets

8.1.1 Discussions

The main crucial parameter for point reduction is scale i.e. radius for neighbor search. We have observed that on decreasing the radius more points are categorized into spherical class while on increasing the radius more points are categorized into disc and curve class. User has to fine tune this parameter to get the correct local structure.

σ_{max} controls the transition between features like creases and corners. On the other hand, σ_{min} differentiate between the creases and non-creases type features on surface. Determining the correct value of radius may take some time. Once the correct value of radius is found, finding σ_{min} and σ_{max} is a lot easier. Result will be varied for different set of parameters.

The percentage of reduction depends upon the initial input parameters. We have set default values for all parameters in the application and later user can fine tune these parameters to get better results. We have tested the algorithm for few las data-sets available at `http://www.liblas.org/samples/`. Table 8.2 shows result by how much each of the data set could be reduced after obtaining the feature graphs.

8.2 Segmentation

Segmentation of reduced point sets is done using region growing algorithm which is discussed in details in the chapter 5. The main critical parameters in this modules are *curvth* and *smooth*. *curvth* defines threshold for curvature value

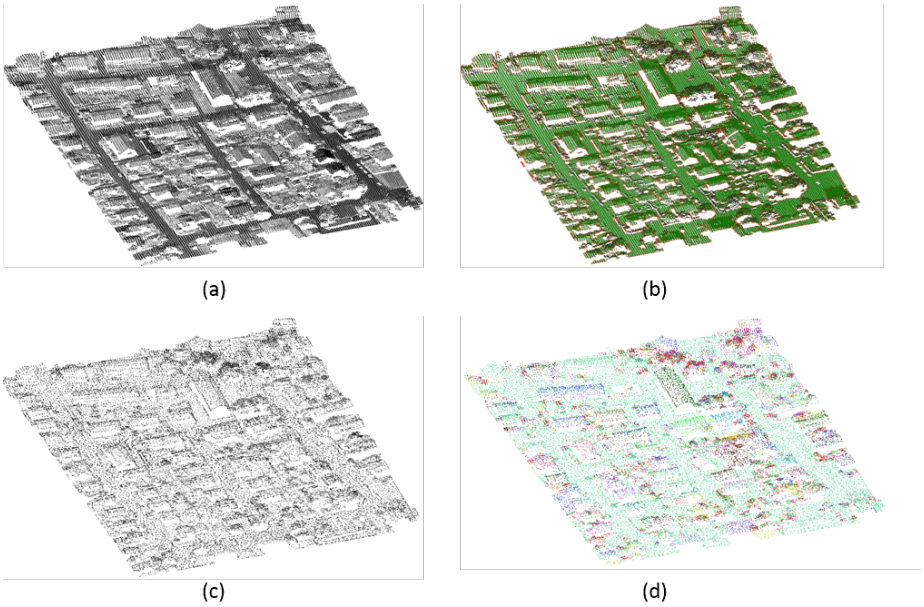

Figure 8.3: Point Classification, Down-Sampling and Segmentation of area 1 of Vaihingen data-set: (a) Original Points, (b) Saliency Map, (c) Reduced Points, and (d) Segmentation. In Figure (b), Green shows surface-type points, Blue shows critical-type points and Red shows line-type points. In Figure(d), Colors are randomly assigned to different clusters. Figure (a) contains 1,99,892 which is reduced to 51,811 in Figure (c).

of a point. All points having low curvature value are considered to be part of a flat area. *smooth* defines the threshold for angle between the normal of neighbor and current seed points. Points which belongs to the same surface have less differences between their normals. Image (d) of Figure 8.1, 8.2, 8.3 and 8.4 show output of segmentation for different las data. Colors are randomly assigned to different clusters.

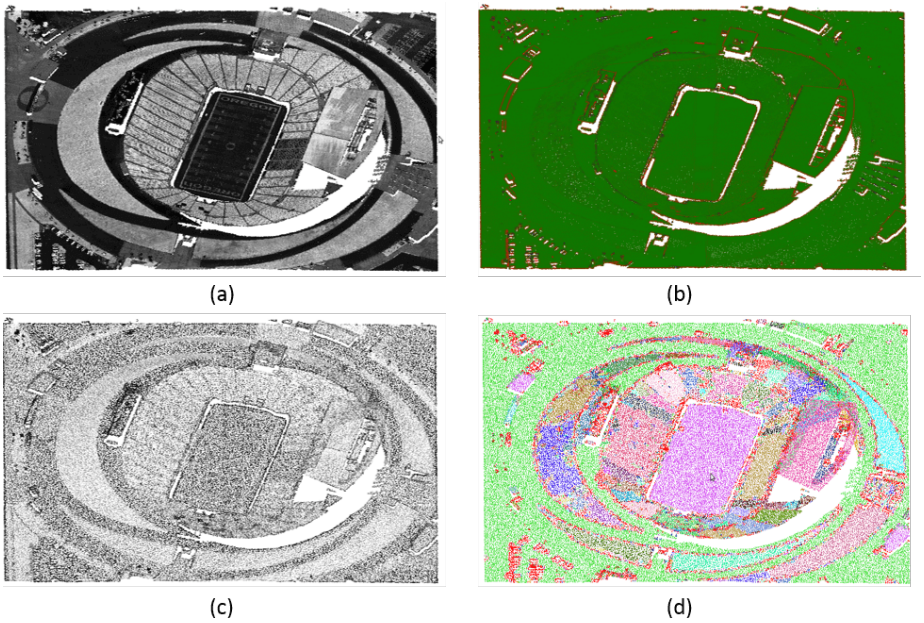

Figure 8.4: Point Classification, Down-Sampling and Segmentation of area 1 of Vaihingen data-set: (a) Original Points, (b) Saliency Map, (c) Reduced Points, and (d) Segmentation. In Figure (b), Green shows surface-type points, Blue shows critical-type points and Red shows line-type points. In Figure(d), Colors are randomly assigned to different clusters. Figure (a) contains 6,93,894 which is reduced to 99,164 in Figure (c).

8.3 Mesh Construction

Algorithm for mesh generation is discussed in detail in the chapter 8.1. There are many parameters in the algorithm that can be tweaked in order to produce even more appropriate mesh, in case the default values do not work well. They have been listed below.

Controlling Your Neighborhood

setMaximumNearestNeighbors(unsigned) and *setMu(double)* are the two functions that control the way you grow your neighborhood and its ultimate size. The first one takes care of the size of the neighborhood, in terms of the number of members that belong to it, which can be varied by varying the value it takes.

The second one caters to the maximum distance that must be considered to keep looking for neighbors.

Edge Length

setSearchRadius(double) is the function that takes care of the maximum possible edge length of any triangle, which must be user defined to keep a constraint on the largest triangle that can be formed to build the mesh, eventually.

Angles Of Triangle

setMinimumAngle(double) and *setMaximumAngle(double)* are the next two important parameters that can be set by the user to control the minimum and the maximum angle between any two sides of a triangle in the, to be formed, mesh. The default values are set at 10 and 120 radians respectively.

(a) (b)

Figure 8.5: (a) and (b) represent the mesh constructed from original and reduced point cloud of Oregon's Autzen Stadium data set [4] respectively.

8.4 Performance

Serial implementation of algorithm proposed by Keller is computationally very expensive and time consuming. But algorithm is embarrassingly parallel. Therefore, we have done the parallel implementation to make the application real-time and user interactive. We have analyzed the results of Keller et al. [15] and compare the performance of serial implementation with parallel implementation. All experiments have been performed on Intel Xeon(R) processor at 3.2GHz quad-core , 8 GB RAM with NVIDIA GeForce GTX480. We have tested the algorithm for various las files [4] and scanned point clouds in ply format [18].

Table 8.3 gives the timing measurement in CPU seconds of the serial implementation as well as parallel algorithm. Timing measurement of parallel implementation are reduced drastically as compared to serial implementation. Parallel implementation has speed up the application as shown in Figure 8.6.

Dataset(las files)		Timing measurement in CPU seconds	
	size	serial Implementation	parallel implementation
test2	11,765	2.02	0.07
test1	33,703	6.73	0.25
galvestone	99,600	48.76	1.36
mscstsc	1,60,101	84	2.25
spring2	2,01,474	144.63	4.76
srsota	3,86,530	262.96	4.34
N144835	4,31,276	369.95	9.4
N440375	4,97,536	439.51	10.04
autzen-stadium	6,93,895	460.21	18.91

Table 8.3: Timing measurements in CPU second for serial and parallel implementation

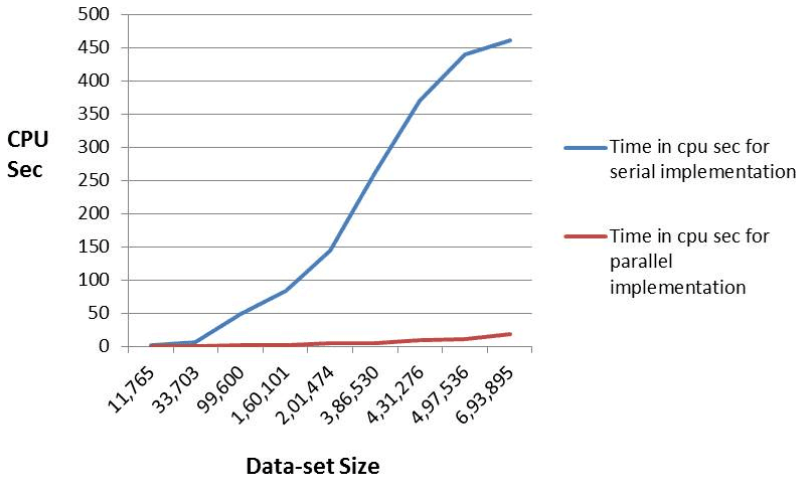

Figure 8.6: Time comparison between serial and parallel implementation

Chapter 9

Conclusions

We have identified using Keller et al.[15] for feature extraction from LiDAR point cloud data for efficient and effective visualization. Our goal is to enable interactive real-time three-dimensional visualization of the point cloud. Hence we propose using extracted feature points and lines to preserve salient features of the data and reducing the large-scale data sets. Since several modules in the preprocessing step are embarrassingly parallel, we have done parallel implementation of Keller et al.'s algorithm [15], using data parallel paradigm using GPU computing. We have also implemented the server-client architecture using thinlinc to deploy the application on a local area network and feature tracking is done in the form of animated video. Organizations with a dedicated local area network can be used this tool to explore and analyze the large LiDAR datasets within the network. Our application can also be deployed on WAN.

As per the specifications of our project documented in the preface, we have implemented a visualization tool which can classify points, identify features, and track features across series of point cloud datasets, pertaining to the same region. This tool is accessible via remote desktop application using the ThinLinc software. We have additionally tested accessing the software through a web browser.

As per our project requirements, the incomplete task is to identify organizations using LiDAR data, which require the visualization, and enable them to use our tool. For completing this task, we would like to request the DST to help identify such organizations.

To ensure the completion of the project, we shall submit/support the following deliverables:

- Provide an exhaustive project report,
- Provide a user manual and informative videos on how to use our tool,
- Publish the documents on our lab web-page, `http://cds.iiitb.ac.in/gvcl`
- Submit the source code and executable scripts, along with system specifications for running the code, to DST.

Future work

Manual feature tracking is done for time-varying lidar data-sets. Automatic feature tracking can be done by comparing the signature of the features in a data-sets over different time stamps. Feature tracking of time-varying LiDAR data-sets can also be used to find temporal patterns in a LiDAR data-sets.

Bibliography

[1] Cendio AB. Thinlinc, 2014.

[2] Stanford University Ashish Goel. Algorithms for Modern Data Models, 2011.

[3] Raoul Bott. Morse theoretic aspects of yang-mills theory. In *Recent Developments in Gauge Theories*, pages 7–28. Springer, 1980.

[4] Howard Butler and Mateus Loskot. Sample Datasets at libLAS - LAS 1.0/1.1/1.2 ASPRS LiDAR data translation toolset, 2013.

[5] Frédéric Chazal, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. Analysis of scalar fields over point cloud data. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1021–1030, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[6] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[7] Michael Cramer. The dgpf-test on digital airborne camera evaluation–overview and test design. *Photogrammetrie-Fernerkundung-Geoinformation*, 2010(2):73–82, 2010.

[8] Joel Daniels, Ii Linh, K. Ha, Tilo Ochotta, and Cludio T. Silva. Robust smooth feature extraction from point clouds. In *In IEEE International Conference on Shape Modeling and Applications*, pages 123–136, 2007.

[9] J De la Porte, BM Herbst, W Hereman, and SJ Van Der Walt. An introduction to diffusion maps. In *The 19th Symposium of the Pattern Recognition Association of South Africa*. Citeseer, 2008.

[10] Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Morse-smale complexes for piecewise linear 3-manifolds. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 361–370, New York, NY, USA, 2003. ACM.

[11] Zeev Farbman, Raanan Fattal, and Dani Lischinski. Diffusion maps for edge-aware image editing. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 145:1–145:10, New York, NY, USA, 2010. ACM.

[12] Yani Ioanou, Babak Taati, Robin Harrap, and Michael Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. In *Proceedings of the 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 3DIMPVT '12, pages 501–508, Washington, DC, USA, 2012. IEEE Computer Society.

[13] Martin Szummer Tommi Jaakkola and Martin Szummer. Partially labeled classification with markov random walks. *Advances in neural information processing systems (NIPS)*, 14:945–952, 2002.

[14] Patric Keller, Oliver Kreylos, Bernd Hamann, Louise H. Kellogg, Eric S. Cowgill, Martin Hering-Betram, and Hans Hagen. Extraction of features from high-resolution LiDaR point cloud data. In *Abstract Proceedings of American Geophysical Union (AGU) Fall Meeting 2008, Eos. Trans. AGU 89(53), Fall Meeting Suppl., AGU Meetings Department, Washington D.C.*, number G53B-0636 (poster presentation), 2008.

[15] Patric Keller, Oliver Kreylos, Marek Vanco, Martin Hering-Bertram, Eric S. Cowgill, Louise H. Kellogg, Bernd Hamann, and Hans Hagen. *Extracting and Visualizing Structural Features in Environmental Point Cloud LiDaR Data Sets*, pages 179–193. Springer-Verlag, Heidelberg, Germany, 2010.

[16] Patric Keller, Oliver Kreylos, Marek Vanco, Martin Hering-bertram, S. Cowgill, Louise H. Kellogg, Bernd Hamann, and Hans Hagen. Extracting and visualizing structural features in environmental point cloud lidar data sets. 2009.

[17] Stephane Lafon and Ann B Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1393–1403, 2006.

[18] Marc Levoy. The Stanford 3D Scanning Repository, 2013.

[19] B. Lohani. Airborne Altimetric LiDAR: Principle, Data Collection, Processing and Applications, 2007.

[20] Caren Marzban and Ulvi Yurtsever. Baby morse theory in data analysis. In *Proceedings of the 2011 Workshop on Knowledge Discovery, Modeling and Simulation*, KDMS '11, pages 15–21, New York, NY, USA, 2011. ACM.

[21] Nvidia. CUDA, 2013.

[22] John D. Owens, Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, May 2008.

[23] Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust on-line computation of reeb graphs: Simplicity and speed. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[24] Mark Pauly, Richard Keiser, Markus Gross, and Eth Zrich. Multi-scale feature extraction on point-sampled surfaces. 2003.

[25] PCL. Region growing segmentation, 2014.

[26] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL), May 9-13 2011.

[27] Issam Ibrahim Safa. *Towards Topological Methods for Complex Scalar Data*. PhD thesis, Columbus, OH, USA, 2011. AAI3493527.

[28] Keil Schmid, Kirk Waters, Lindy Dingerson, Brian Hadley, Rebecca Mataosky, Jamie Carter, and Jennifer Dare. Lidar 101: An introduction to lidar technology, data, and applications. *NOAA Coastal Services Center*, 2012.

[29] Georgios Stylianou and Gerald Farin. Crest lines for surface segmentation and flattening. *IEEE Transactions on Visualization and Computer Graphics*, 10(5):536–544, September 2004.

[30] Marc van Kreveld, René van Oostrum, Chandrajit Bajaj, Valerio Pascucci, and Dan Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, SCG '97, pages 212–220, New York, NY, USA, 1997. ACM.

[31] Miao Wang and Yi hsing Tseng. Automatic 3d feature extraction from structuralized lidar data, 2005.

[32] T. Weyrich, M. Pauly, R. Keiser, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Proceedings of the First Eurographics Conference on Point-Based Graphics*, SPBG'04, pages 85–94, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.