Remote Interactive Visualization of Parallel Implementation of Structural Feature Extraction of Three-dimensional Lidar Point Cloud

Beena Kumari, Avijit Ashe, and Jaya Sreevalsan-Nair

Abstract

Lidar (Light detection and ranging) is a popularly used technology to collect dense and precise data of topographic structures of the surface of the earth. In this paper, we have proposed a user assisted remote visualization system for extraction and visualization of structural features obtained from point cloud data obtained from lidar data. The sharp feature lines such as crest lines, edges of buildings, ravines, ridges are known as structural features of point cloud. Our work includes: (a) parallel implementation a topology-based algorithm for extraction of structural features from lidar point cloud using GPGPU (General Purpose Graphics Processing Unit) computing, and (b) using a LAN (Local Area Network)-based server-client architecture to achieve remote visualization.

Keywords: 3D interactive visualization, lidar point clouds, feature extraction, stochastic methods, GPGPU (parallel) computing

I. Introduction

The lidar (Light Detection and Ranging) technology is heavily used for flood-modeling and similar applications, such as, bathymetry, geomorphology, glacier modeling, etc. Point cloud or positional vector data can be processed from raw lidar data. We have focused on using topology-based algorithm to extract features from such point cloud data. Point cloud datasets are generally visualized directly as a point cloud or in a derived form as a mesh. Our primary focus in this work is to reduce the point cloud to essential points, which encode structural features.

Additionally, raw lidar datasets contain noise due to instrumental errors and atmosphere. Hence, proper pre-processing and filtering are necessary for both managing large scale dataset as well as for denoising. We have used the heuristic algorithm proposed by Keller et al. [3] to denoise and to extract the structural features from lidar datasets. However, the algorithm is highly computationally intensive and the serial implementation of the preprocessing module is time consuming. In order to speed up the preprocessing for data simplification, we have exploited the data-parallel characteristic of the algorithm and implemented the parallel algorithm using GPGPU computing techniques [5] using CUDA libraries [4].

Visualization of lidar data has been motivated by target users of this visualization who are domain experts from national laboratories or similar organizations. The target users additionally expressed the interest to access such a visualization across a LAN (local area network), so that the data resides in a server and the interactive displays may be enabled on thin clients. The server-client architecture has been implemented using a product called ThinLinc [2].

II. System Description

The processing system consists of three modules: (a) a server, which performs all the back-end operations, (b) a thin-client, which performs all the front-end operations and presents the user interface, and (c) the virtualization interface, which is the transporting mechanism that enables the communication between the front-end and the back-end.

A. Back-end (Server) Computing & Rendering

We have implemented the server-client architecture using HP Z400 Workstations running Ubuntu 12.04.1 LTS as the server. We have used OpenGL libraries for the graphics support, and CUDA libraries for GPGPU computing. As an alternative to our OpenGL rendering application, we have used Paraview, which is an open-source cross-platform data analysis and visualization application developed by Kitware. Our OpenGL-based rendering tool is dedicated towards point cloud visualization while Paraview has extensive capabilities to perform visual data analytic operations like slicing, masking the point cloud, etc. Using Paraview, we have performed meshing using the greedy surface triangulation algorithm on a point cloud dataset with normals, where the outcome is a triangle mesh based on projections of the local neighborhoods. While currently we have focused on point cloud visualization, we are interested in extending the modes of visualization to other standard techniques, such as, surface and volume visualizations.

This document is preprint, as of September 4, 2014. The authors are with Graphics-Visualization-Computing-Lab, International Institute of Information Technology Bangalore ; email: {beena.kumari@iiitb.org, avijitashe@iiitb.ac.in, jnair@iiitb.ac.in}

B. Virtualization Interface

User interactive sessions and real-time image manipulation and rendering calls for a very efficient VNC (Virtual Network Controller). TigerVNC is the most suitable protocol for this endeavor, especially for our application that includes integration of high-end graphics framework. ThinLinc is a Remote Linux Desktop Server developed by Cendio AB [2], entirely built on open-source. It provides users with centralized remote access to both Windows and/or Linux desktop applications simultaneously and is a secure, cost-effective and freely install-able solution for up to ten concurrent users. Overall, ThinLinc meets with our requirements for this work. TigerVNC is integrated into ThinLinc.

1) OpenGL Rendering Support: VirtualGL is one of the most robust solutions to allow OpenGL applications to be ported via a VNC. Traditionally the 3D data is transmitted over the network to be finally rendered on the client machine. VirtualGL is designed to direct the OpenGL commands and the 3D data to the graphics accelerator on the application server, as in our case, and direct the rendered images across to the client machine. Thus it enables users to interact in real-time using even hand-held devices as the front-end device is used only for display and user interactivity. VirtualGL is optionally integrated with ThinLinc, and we have additionally tested our system with VirtualGL integration on ThinLinc.

C. Front-End (Thin-Client) UI

The thin-client essentially has X libraries installed to enable the display and interaction for the GUI (graphical user interface). We have tested the system on the following thin-client configurations: (a) Ubuntu 12.04.1 LTS running on Dell XPS as well as on Dell Inspiron 3520, and (b) Windows 7 Ultimate running on ASUS X42.

III. Implementation

We have implemented the algorithm proposed by Keller et al. [3] to detect and extract structural features in the objects in the lidar point cloud. These features include corners, border-, crease-, or ridge-lines in the form of a feature graph. The algorithm consists of the following modules: (a) outlier removal/denoising, (b) stochastic point classification, (c) smoothening of feature values, and (d) feature graph construction. The serial implementation of the algorithm is slow, owing to its computationally intensive nature. Hence we have implemented the algorithm in parallel using CUDA [4] and PCL [6] libraries, and tested on an Intel Xeon(R) processor at 3.2GHz quad-core, NVIDIA GeForce GTX480. Table I compares the performances of the serial and parallel implementation.

Datasets [1]	test2	test1	galvestone	mscstsc	spring2
# Points	11765	33703	99660	160101	201474
Serial (in CPU seconds)	2.02	6.73	48.76	84	144.63
Parallel (in CPU seconds)	0.07	0.25	1.36	2.25	4.76
Speedup	28.85	26.92	35.85	37.33	30.38
Datasets [1]	srsota	N144835	N440375	autzen-stadium	spring1
# Points	386530	431276	497536	693895	926276
Serial (in CPU seconds)	262.96	369.95	439.51	460.21	2529
Parallel (in CPU seconds)	4.34	9.4	10.04	18.91	38.73
Speedup	60.58	39.35	43.77	24.33	65.29

TABLE I
COMPARISON BETWEEN SERIAL AND PARALLEL IMPLEMENTATION OF STRUCTURAL FEATURE EXTRACTION ALGORITHM [3]

IV. Experiments and Discussions

We have analyzed the results of the serial and parallel implementations of the algorithm proposed by Keller et al.[3] for various datasets. Table II gives the classification of the points into different classes - curved, planar, critical curved and critical planar and also shows the extent of reduction in, each case, upon computing the feature graphs. We can see that the coplanarity of the points in the data set, and the reduction of the point data set are positively correlated

Dataset	**Point Classification**					**Point Reduction**	
	# Points			# Critical Points		# Points	%age
	Total	Curved	Planar	Curved	Planar	(Reduced)	Reduction
Leica	1,642,660	106,654	1,589,446	321	2,744	108,276	93.40
Stadium	693,895	54,174	633,524	1,358	1,376	139,435	79.90
Dragon	437,645	35,975	420,319	215	220	146,882	66.43
Stanford Bunny	35,947	14,621	29,893	29	29	20,337	43.43

TABLE II
POINT CLOUD CLASSIFICATIONS AND PERCENTAGE OF REDUCTIONS

V. Conclusions and Future work

We are reducing the point cloud data procured using lidar technology using derived information of inherent structural features, using a heuristic topology-based algorithm [3]. We have observed that several modules in the algorithm are embarrassingly parallel and hence, we have used CUDA for efficient parallel implementation. To enable the server-client architecture for this application, we have used ThinLinc to deploy the application over a LAN. Such a setup enables organizations with a dedicated LAN to use our tool to explore and analyze the lidar datasets without any hardware upgrade. We will be further working on the tracking of the features across time-series data-sets to find temporal patterns.

References

[1] Butler, H., Loskot, M.: Sample Datasets at libLAS - LAS 1.0/1.1/1.2 ASPRS LiDAR data translation toolset (2013), http://www.liblas.org/samples/
[2] Cendio AB: Thinlinc (2014), https://www.cendio.com/products/thinlinc/
[3] Keller, P., Kreylos, O., Vanco, M., Hering-Bertram, M., Cowgill, E.S., Kellogg, L.H., Hamann, B., Hagen, H.: Extracting and Visualizing Structural Features in Environmental Point Cloud LiDaR Data Sets, pp. 179–193. Springer-Verlag, Heidelberg, Germany (2010)
[4] NVIDIA: CUDA (2013), https://developer.nvidia.com/category/zone/cuda-zone
[5] Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU computing. Proceedings of the IEEE 96(5), 879–899 (May 2008)
[6] Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL) (May 9-13 2011)