RAVEPC: Remotely Accessible Visualizer & Explorer of Point Cloud

An Interactive Visualization Application for LiDAR Data : Part III

Authors: Beena Kumari, Avijit Ashe and Jaya Sreevalsan Nair

Graphics Visualization Computing Lab,
International Institute of Information Technology Bangalore,
26/C Electronics City, Hosur Road, Bangalore 560100, India.
http://cds.iiitb.ac.in/gvcl

Software Manual Ver-0.0, August, 2015

This document is a Part III of 3-part series of the user manual 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 instruction for installation and point reduction. Part II contains instruction for feature tracking and Part III includes procedure for remote visualization, and appendix section.

Software Description

RAVEPC is an open source desktop application for remote visualization of 3D range data. It can also be used to manually track the features in time-varying 3D range data. Currently, it supports las, ply and pcd data file format and is developed for Ubuntu operating system. It has server-client architecture where server should have high-end graphics card with all computational capability and clients can be used as thin devices for display purposes. User interface has been written using C++, fltk library and OpenGL. Algorithm is implemented using C++, PCL library and CUDA. Remote visualization is done using an open source product ThinLinc developed by Cendino AB. RAVEPC supports following features:

- RAVEPC is a stand alone desktop application for visualization and exploration of 3D Range data.
- It supports las, ply and pcd data file format.
- Points Classification into Geometrical classes
- Manual Feature Tracking up to 3 time-stamp data
- Remote visualization and analysis of the 3D range data
- Mesh Construction from unstructured 3D point cloud

Disclaimer

Contents

Part I

Part II

Part III

List of Figures

Chapter 4

Remote Visualization

4.1 Remote Desktop Access

Remote visualization is achieved with the help of ThinLinc. RAVEPC should be installed on the server with high-end computational capability and thin devices can be used as clients for display purpose. Installation procedure for RAVEPC and Thinlinc is already discussed in the chapter 1. Figure 4.1 shows the client desktop. User can search for ThinLinc application and open it. It will prompt a window to enter the details for server to which user want to connect it. To log into server, user has to entered the following details: Domain name or ip address, user name and password. Figure 4.2 and Figure 4.3 show steps to log into the server. Once the Connection is established between server and client, Server desktop will be prompted in the client desktop as shown in the Figure 4.4. User can directly work on the server desktop and open the RAVEPC to process the data. Only those data-sets can be visualized and processed which resides inside the server. User can toggle between client and server desktop using *fullview option* and disconnect the session using *disconnect* option in the server desktop as given in the FIgure 4.4.

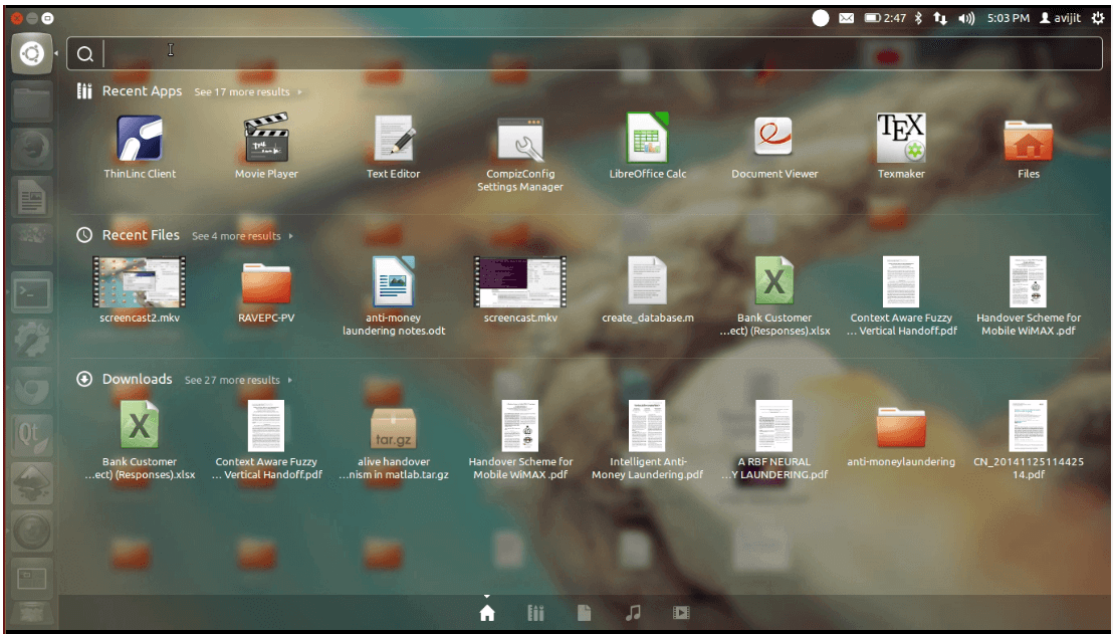

FIGURE 4.1: Client Desktop

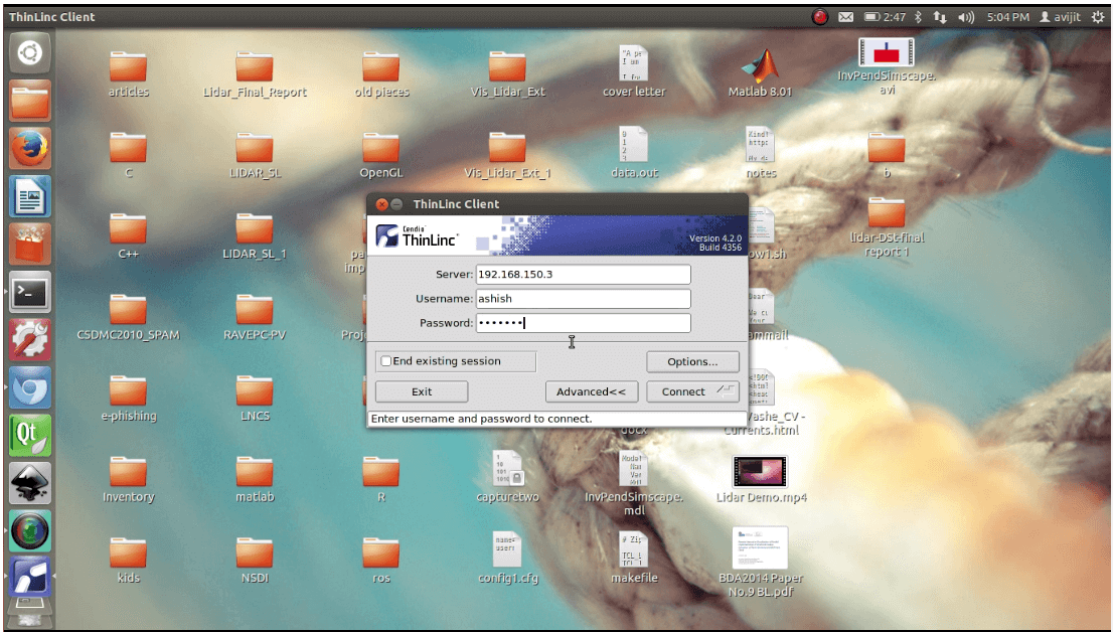

FIGURE 4.2: Client Desktop

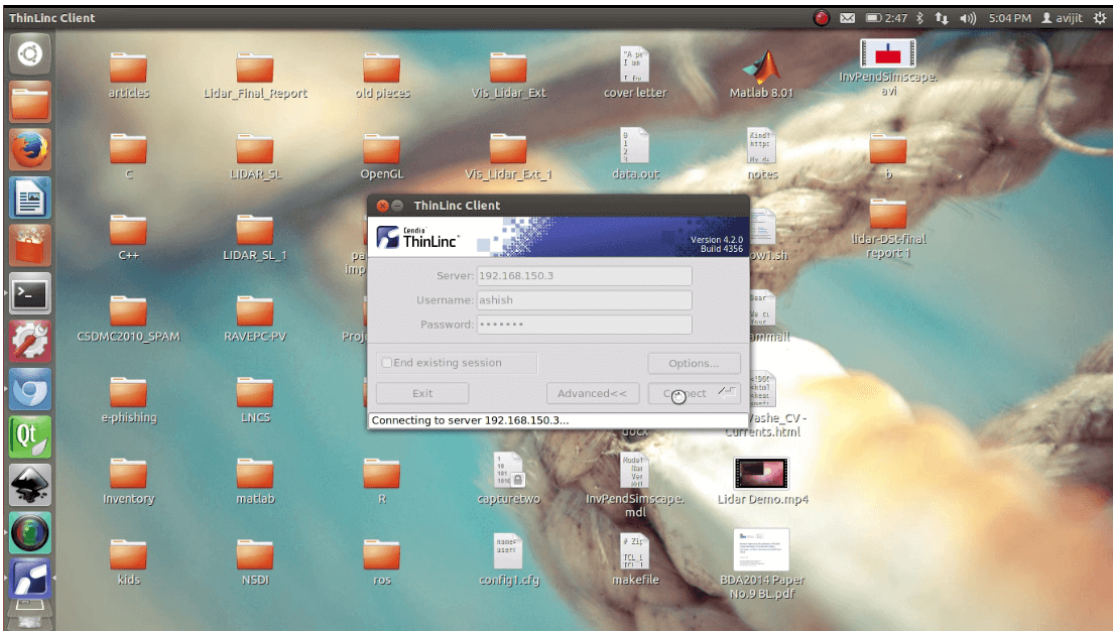

FIGURE 4.3: Client Desktop

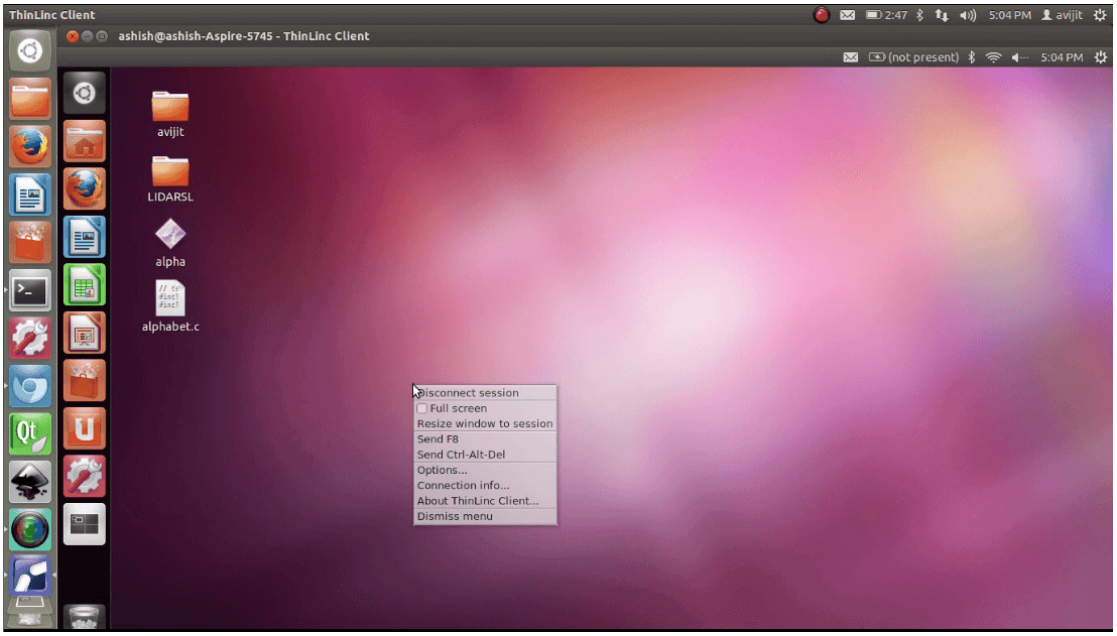

FIGURE 4.4: Server Desktop

4.1.0.1 Web Browser Access

User can connect to the server through internet also. Provided, user have valid domain name of the server. Installation of ThinLinc for web-access is also discussed in the chapter 1. Enter the correct domain name with valid port number in the web browser. It will prompt a window where user can enter the logging details and connection will be established to the server. Login to the server using web is similar to the procedure required to login to the server on LAN as shown in the Figure 4.5.

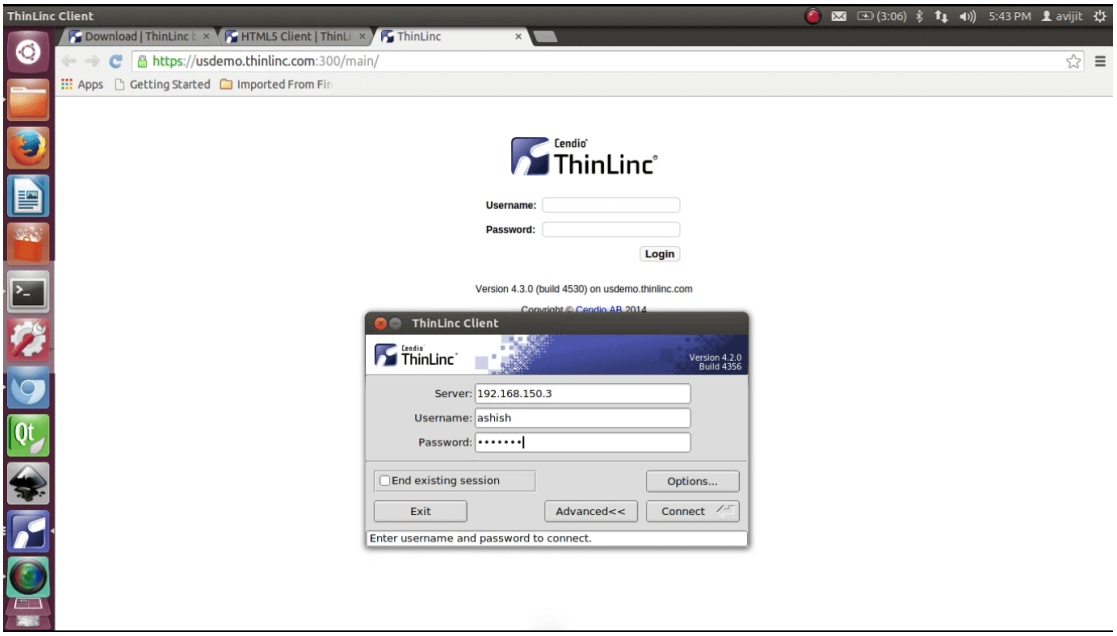

FIGURE 4.5: Web Browser Access to server

Appendix A

RAVEPC

A.1 Frequently Asked Questions

Q1. Can RAVEPC be used as a desktop application?
Yes, provided machine has high-end graphics card and all computational capability.

Q2. What is the maximum size of the data that can be processed by RAVEPC?
It depends upon the system configuration (RAM and GPU memory). RAVEPC gives out of memory error if system does not have enough memory to process the file.

Q3. What is the minimum architecture of the CUDA supported by RAVEPC?
Cuda-2.0 architecture or higher.

Q4. What is the minimum version of PCL library supported by RAVEPC?
PCL-1.7.

05. How many time-varying files can be used for feature tracking?
Only 3 time-varying files can be processed at a time. If number of files is more than 3, user can processed them in groups.

06. what is the data-formats supported by RAVEPC?
Las, Ply and PCD data format.

Appendix B

ThinLinc

This chapter discusses some important frequently asked questions related to ThinLinc, which forms the first part. It has two parts in all, the second one being targeted to various test cases that we have performed with this remote connectivity tool.

B.1 Frequently Asked Questions

Q1. What is ThinLinc?

A product from Cendio AB, ThinLinc is a Linux Remote Desktop Server. It provides a very cost-effective means to deploy a server-client architecture across various platforms to allow remote desktop virtualisation for running graphics intensive applications with full 3D hardware acceleration with minimum resources required apart from the expensive server.

VirtualGL used for rendering graphics on the server, a VNC (Tiger VNC) for transfer of data across a WAN/LAN and PulseAudio for sound support. All the connections are made secure using SSH sessions between server(s) and client(s).

Q2. What are the supported platforms?

Practically any dumb terminal can be converted to act as a ThinLinc Client and any LSB terminal can be converted to act as a ThinLinc Master Server. Application Servers may be of any platform like Windows, Mac, Unix etc. which must be controlled by the Master Server. This enables extreme flexibility, scalability and robustness.

Q3. What is the installation procedure?

The setup consists of two parts: server setup and client setup. The setup files can be downloaded fom their official website download page. There are two executable files which require

just a "double click" to initiate installing necessary packages and their dependencies and you are ready to test the default establishment right away. It might take about 5-10 minutes for the server setup to get completed, depending on the internet speed, and about a minute for the client setup. The servers can be accessed even by a browser for which Java needs to be installed on the client. An HTML5 version is also availble that is completely HTML based and doesn't need Java anymore. By using the browser both the native client as well as direct connection can be established. If using the browser, then no client setup is needed anymore and practically any device with a web browser with HTML5 support can run applications remotely. It also settles the issue of using it as a windowing application that products like NOMACHINE provide, however, without needing any client side installation.

Q4. How do we know that it is running on the other system and not on the local? or Why isn't there an option like a windowing application to be able to minimize between local and remote by default?

Both these questions actually answer each other. Let us suppose that thinlinc sessions ran just like any other application e.g VMware or VirtualBox. So the thing to consider here is that when such a 'virtualisation' occurs it actually makes use of the locally available resources (hardware as well as software) and thus needs equally eligible constructs that can support them and as well run them smoothly. Hence we have the facility to use them just as any other application. Now creating an exclusive session provides evidence that the user has been actually transferred from the local machine to the remote machine and that any further communication by no means shall make use of local resources except like a power source, mouse, keyboard, NIC card et cetera. But however, in the recent releases, it includes a browser version with HTML5 support and another one Java based. They allow the user to connect to the remote application servers through the browser (not via native client installed) and thus making thin clients even more thin and portable. The essence being a ultra low profile device can have the ability to run cross-platform applications remotely on high-configuration servers, and above all using free open source software.

Q5. How does the image reconstruction take place at the thin terminal? or How does this architecture help minimizing the need of expensive client machines?

The clue lies in OpenGL which is the heart of this architecture and all that goes and comes between the server and the client. This is also the inherent feature of ThinLinc and that is why both of them are together. ThinLinc is one of the fewest remote desktop applications that enable runing OpenGL based applications with full 3D hardware acceleration. Full 3D hardware acceleration means that the OpenGL applications are not forced to use a software only rendering but actually enable the graphics processing unit (onboard graphics card power) to participate or completely handle the rendering of the images. This is quite cumbersome for the CPU alone, even if its were a supercomputer.

The traditional method of rendering included remote X-server to cause the rendering and if 3D hardware acceleration was needed, it had to be present on the remote client. All the 2D as well as 3D data had to be transferred across the network and this all affected the cost, fluency

and the bandwidth requirements. VirtualGL, which is the virtual OpenGL, the commands and 3D data are instead redirected to the GPU of the server and only the rendered 3D images are transferred to the client machine. VirtualGL thus virtualizes the graphics hardware present on the application server to the thin client. Whenever the user interacts with the image, the commands are sent over to the server, it renders the image, checks the differences brought by it on the screen of the earlier display and transfers only the changed positions to the client where it gets displayed on the screen. It is just using a telescope to view the stars instead of flying right to it on a spaceship. Thus it saves money of the space travel and in our case of the client machines.

Q6. How does it differ from something like rdesktop or ulteo?

Comparisons among different products in the market, both commercial and free, need to be based on certain objectives and then the pros and cons of the listed product in attaining those.

In here the objective is to provide remote desktop virtualization for graphics intensive applications with optional full 3D hardware acceleration using as minimum resources as possible excluding the expensive but necessary server. This mainly includes the network connecting the two and the client machine. On the server VirtualGL takes care of rendering of the point cloud. A VNC protocol is used to transfer the image and user interaction data to and fro the network making it the only data being transferred, compared to traditional methods of providing 3D data visualization.

B.2 Test Cases

B.2.1 Creating Client Sessions

To test these we used two separate systems, one with only Windows 7 installed and the other running Ubuntu 12.04, Our aim was to run LIDAR data files (.las) using OpenGL and some of the required libraries. This was necessary to test out 3D data visualization feature, with and without 3D hardware acceleration. All the installations were done on the Ubuntu running system (including the ThinLinc Master Server). The ThinLinc Client package for Windows was downloaded from the website.

Problem Encountered: We were unable to connect to the server using the usual server's (Ubuntu) login username and password.

Solution: We tried pinging one terminal from another. Even if both were able to connect to the Internet but still were unreachable, how so ever. The problem might possibly be in the router firewall configuration. We tried to connect to the systems even using public IP but that did not resolve the issue. Thus we created a separate LAN for these two systems. Now we could connect easily by just typing in the login name and password from the client side (Windows 7).

B.2.2 Providing Server Side Computing

With the same setup as above, the next thing was to check whether it actually runs the applications out of the Ubuntu system. Generally in an office environment or in a lab all the terminals are clones of each other. So verifying this gets complicated. Hence we used two systems completely different from each other.

Conclusion: The OpenGL application ran smoothly on the Windows 7 system.

NOTE: Opening or closing applications on a client doesn't revert back these operations onto the server and the server side is still clean for whatever tasks that need to be carried , completely unhindered by the client activities.