RENDERING OF DENSE, POINT CLOUD DATA IN A HIGH FIDELITY DRIVING SIMULATOR

FINAL PROJECT REPORT

by

David S. Hurwitz (PI)
Oregon State University

Michael Olsen (Co-PI)
Oregon State University

Patrick Marnell
Oregon State University

Hamid Mahmoudabadi
Oregon State University

for

Pacific Northwest Transportation Consortium (PacTrans)
USDOT University Transportation Center for Federal Region 10
University of Washington
More Hall 112, Box 352700
Seattle, WA 98195-2700

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The Pacific Northwest Transportation Consortium and the U.S. Government assumes no liability for the contents or use thereof.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
2012-S-OSU-0010	01538105	

4. Title and Subtitle	5. Report Date
Rendering of Dense, Point Cloud Data in a High Fidelity Driving Simulator	September 15, 2014
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
David S. Hurwitz, Michael Olsen, Patrick Marnell, and Hamid Mahmoudabadi	10-739437

9. Performing Organization Name and Address		10. Work Unit No. (TRAIS)
PacTrans Pacific Northwest Transportation Consortium, University Transportation Center for Region 10 University of Washington More Hall 112 Seattle, WA 98195-2700	Oregon State University	**11. Contract or Grant No.** DTRT12-UTC10

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
United States of America Department of Transportation Research and Innovative Technology Administration	Research 9/1/2012-7/31/2014 **14. Sponsoring Agency Code**

15. Supplementary Notes

Report uploaded at www.pacTrans.org

16. Abstract

Driving Simulators are advanced tools that can address many research questions in transportation. Recently they have been used to advance the practice of transportation engineering, specifically signs, signals, pavement markings, and most powerfully to examine the safety and efficiency of alternative transportation solutions. These simulators are a powerful 3D, virtual environment enabling the study of how drivers respond to potential designs or policies. A key challenge is virtual environment that maintains high fidelity to the real world. 3D laser scanners, which use Light Detection and Ranging (LIDAR), are line-of-sight technology that emits laser pulses at defined, horizontal and vertical angular increments to produce a 3D point cloud, containing XYZ coordinates for objects that return a portion of the light pulse within range of the scanner. This detailed point cloud is a virtual world that can be explored and analyzed by a variety of people. Through the combination of these two technological systems, more authentic, virtual, built-environments can be used by transportation engineering professionals for the purpose of 3D design.

This research project focuses on the technical issues of importing and displaying 3D laser scan data within a driving simulator. For import in the simulator, datasets need to be in the VRML97 format with color values scaled from [0 1]. A transformation needs to be applied to convert between real-world coordinates and screen coordinates. Large datasets should be filtered, when possible, and tiled into very small increments (< 35 MB) to maintain system interactivity.

17. Key Words	18. Distribution Statement
Driving Simulator, LIDAR, 3D-Design, Point Clouds	No restrictions.

19. Security Classification (of this report)	20. Security Classification (of this page)	21. No. of Pages	22. Price
Unclassified.	Unclassified.		NA

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Table of Contents

List of Figures

List of Tables

List of Abbreviations

3D: Three-dimensional
MLS: Mobile LIDAR systems
LIDAR: Light Detection and Ranging
OSU: Oregon State University
PacTrans: Pacific Northwest Transportation Consortium
RTI: Real-Time Technologies Inc.
VRML: Virtual Reality Markup Language
WSDOT: Washington State Department of Transportation

Acknowledgments

This project was funded by the Pacific Northwest Transportation Consortium (PacTrans). The authors would like to recognize the contributions of time and technical expertise to the project made by graduate research assistants Patrick Marnell and Hamid Mahmoudabadi.

Additionally, the authors would like to recognize the matching support provided by the Portland Bureau of Transportation. Without this matching support, the research would not have been possible.

Finally, the authors thank Leica Geosystems and Maptek I-Site for providing software that was used for this study.

Executive Summary

Driving Simulators are advanced tools that have been applied to the study of engineering, physiology, and medicine. Recently they have been used to advance the practice of transportation engineering, specifically signs, signals, pavement markings, and most powerfully to examine the safety and efficiency of alternative transportation solutions. These simulators are a powerful 3D, virtual environment enabling the study of how drivers respond to potential designs or policies. A key challenge is virtual environment that maintains high fidelity to the real world. 3D laser scanners, which use Light Detection and Ranging (LIDAR), are line-of-sight technology that emits laser pulses at defined, horizontal and vertical angular increments to produce a 3D point cloud, containing XYZ coordinates for objects that return a portion of the light pulse within range of the scanner. This detailed point cloud is a virtual world that can be explored and analyzed by a variety of people. Through the combination of these two technological systems, more authentic, virtual, built-environments can be used by transportation engineering professionals for the purpose of 3D design.

The specific purpose of this research effort was to determine if dense 3-D point clouds could be rendered in the driving simulator. In order to interface between the laser scanner data and a high fidelity driving simulator one must perform three tasks:

1. Export the point cloud from the scan software package into a data format that can be introduced into a driving simulator efficiently and without data loss
2. Import the point cloud into the simulator's scenario editing tool in a way that allows manipulation and scenario design
3. Optimize the performance of the point cloud scenario in the driving simulator

These tasks were addressed by using a 3D laser scanner to collect a point cloud representation of the intersection of 14th and Campus Way on the Oregon State University campus in Corvallis, Oregon. The following major findings were determined from the investigation:

- ✓ VRML97 is the only common format that can be imported into RTI driving simulators that are exported from various 3D scanning software packages. Any other format must be converted into VRML97.
- ✓ The color scale must be coded in floating point precision values from [0 1].
- ✓ Creating a transformation node within the VRML file is an effective, and sometimes necessary, way to translate and rotate the point cloud model to reduce large coordiantes before importing into *SimVista*.
- ✓ Alternatively, using scanner coordinate system brings point clouds near the origin in *SimVista* making further manipulation simpler. When using multiple scans a truncated state plane (or similar) coordinate system will allow for the easy manipulation of data while providing a common reference frame.
- ✓ Per object, file sizes must be kept relatively small, requiring careful thought in processing.
- ✓ Removing any extraneous data points from the model can help control total file size and still maintain high quality data. This can be accomplished by filtering points far from the scanner (by range) and areas of high point density close to the scanner (by minimum separation).
- ✓ Splitting large files into several, smaller partitions that are easier for the simulator to process. Although each driving simulator's computer system is unique, partitions smaller than 35MB work best with the OSU driving simulator.

Chapter 1 Introduction

The purpose of this project is to create accurate, high-resolution, three-dimensional (3D) models of real world sites using laser scanning point clouds that are compatible with a high fidelity driving simulator. The use of laser scanners to create point cloud models enables the rapid creation of extremely accurate digital 3D models of real world sites. If point cloud models can be imported into a high fidelity driving simulator efficiently, this would allow researchers and engineers to study potentially dangerous real world transportation facilities in a low risk simulation environment. The three main challenges associated with this task will be exporting the point cloud in a format that can be introduced into a driving simulator, importing the point cloud into the simulator's scenario editing tool, and optimize the performance of a point cloud scenario in a driving simulator.

Although some similarities exist between different driving simulators, there little interoperability and coordination in development between platforms. In particular, the diversity of software used to model driving simulator scenarios creates challenges unique to each platform rendering scenario sharing between platforms inefficient and impractical. For this reason the research team focused on developing a process to import point clouds models into a mid-range driving simulator located at the Oregon State University (OSU) Driving Simulation Laboratory, which runs the SimVista software from the vendor Realtime Technologies Incorporated (RTI). While the exact procedure and constraints will vary depending on the platform, it is likely that many challenges overcome in this study will exist with other platforms.

Chapter 2 Literature Review

2.1 Driving Simulators

There is a wide price spectrum of driving simulators, depending on their ability to recreate and interact with a virtual environment. On the low end, costing as little as $10,000, there are small desktop based simulators with video game like controls (Figure 2.1) (Fisher et al. 2011, STSIM 2013). On the high end, there are facilities like Toyota's $15 million driving simulator which has a 35 meter by 20 meter motion platform enabling motion in nine degrees of freedom (translation in the X, Y, & Z directions, yaw, pitch, & roll, and three vibration degrees of freedom (McNamara 2009, Fisher et al. 2011). Table 2.2 shows the Toyota driving simulator.

The software that operates a driving simulator and is used to create scenarios is specific to each driving simulation platform. In the *Handbook of Driving Simulation for Engineering, Medicine, and Psychology* Fisher et al. (2011) present a comparison of four scenario authoring systems (Table 2.1).

Figure 2.1 Desktop Simulator (from STISIM 2013)

Figure 2.2 Toyota Driving Simulator (from McNamara 2009)

Table 2.1 Comparison of Four Scenario Authoring Systems (Adapted from Fisher et al. 2011)

	STISIM Drive:	***SimVista:***	***Hank:***	***ISAT:***
Developer	Systems Technology Inc.	Realtime Technologies Inc.	Hank Project, Dept. of Computer Science, The University of Iowa	The University of Iowa, NADS
Type	Commercial, low cost, small scale	Multi-purpose, commercial simulator	In-house, small scale research simulator	Large-scale research system; smaller systems running miniSim
Web Site	www.stisimdrive.com	www.simcreator.com	www.cs.uiowa.edu/~hank	www.nads-sc.uiowa.edu
Interface	Text-based	GUI	Text-based	GUI
Object Placement	By route	On Map	By address	On Map
Scene	Integrated development; text-based	Integrated development: tile-based	Developed separately; text based "logical" scene + visual model	Integrated development; tile based
Ambient Traffic	Created manually by individual vehicle events	Automatically generated using "bubble-based" algorithm	Generated by sources	Generated by sources
Critical Events	Controlled by triggers built into individual events	Controlled by special scenario objects - sensors	Controlled by triggers built into other scenario objects	Controlled by special scenario objects - triggers

2.2 The OSU Driving Simulator

The OSU Driving Simulator (Figure 2.3) is a high-fidelity, motion-based simulator, consisting of a full 2009 Ford Fusion cab mounted above an electric pitch motion system with one degree of freedom capable of rotating ±4 degrees. The vehicle cab is mounted on the pitch motion system with the driver's eye-point located at the center of the viewing volume. The pitch motion system enables the accurate representation of acceleration or deceleration on tangent section of roadway (OSU 2011).

Figure 2.3 Views of the OSU Driving Simulator from inside (left) and outside (right) the vehicle

Three silicon-based liquid-crystal display (LCD) projectors with a resolution of 1,400 by 1,050 pixels each are used to project a front view of 180 degrees by 40 degrees. These front screens measure 11 feet by 7.5 feet. A digital light-processing (DLP) projector is used to display a rear image for the driver’s center mirror. The two side mirrors have embedded LCDs. The refresh rate for the projected graphics is 60 hertz. Ambient sounds around the vehicle and internal sounds to the vehicle are modeled with a surround sound system.

The OSU Driving Simulator runs the simulation software *SimCreator,* also provided by vendor RTI. The computer hardware that the driving simulator operates on consists of a server integrating five computers to run the car interface and a sixth server for the bicycle interface (not used in this study). Of the computers for the car interface, one computer processes each of the visual channels center, rear, right, and left, and the fifth computer acts as the host, coordinating the others. Table 2.2 shows technical specifications for the OSU Driving Simulator server.

Table 2.2 OSU Simulator Server Specifications

	OS:	**Processor:**	**Cores:**	**Speed:**	**RAM:**
Center	*Windows XP Professional* Service Pack 3	Intel Core 2 Quad Q6600	4	2.40 GHz	2 GB
Rear	*Windows XP Professional* Service Pack 3	Intel Core Duo E6400	2	2.13 GHz	2 GB
Left	*Windows XP Professional* Service Pack 3	Intel Core Duo E6400	2	2.13 GHz	2 GB
Right	*Windows XP Professional* Service Pack 3	Intel Core Duo E6400	2	2.13 GHz	2 GB
Host	*Windows XP Professional* Service Pack 3	Intel Core 2 Quad Q6600	4	2.40 GHz	3 GB
Bicycle	*Windows XP Professional* Service Pack 3	Intel Core i7-860	4	2.80 GHz	3.18 GB

Researchers typically build scenario environments and track subject drivers from within the operator workstation, shown in Figure 2.4. Scenario modeling is performed using *SimVista*, a variant of *Internet Scene Assemble* (ISA), a 3D authoring tool that allows for the creation of dynamic interactive scenes. *SimVista* will only accept scenario elements in the form of Virtual Reality Modeling Language (VRML97) with the file extension ".wrl". Although VRML is an older language and has been superseded by X3, it still is widely used within the graphics community and within software. VRML 97 provides adequate graphics for use in driving

simulation. However, one key disadvantage to the VRML format is that it is an ASCII clear-text format, which leads to larger file sizes and longer load times compared to structured, binary database formats.

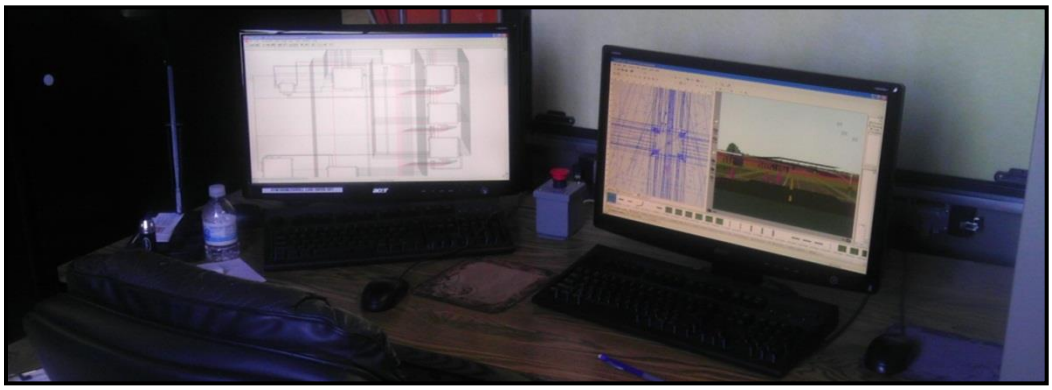

Figure 2.4 Operator Workstation for the Driving Simulator

2.3 Laser Scanning and LIDAR

3D laser scanners, which use Light Detection and Ranging (LIDAR), are a line-of-sight technology that emits laser pulses at defined, horizontal and vertical angular increments to produce a 3D point cloud, containing XYZ coordinates for objects that return a portion of the light pulse within range of the scanner. This detailed point cloud is a virtual world that can be explored and analyzed. After data acquisition, a generated model from the point cloud can be used for different purposes. It worth mentioning that the basic model is the point cloud itself.

Time-of-flight and phase shift are two methods for range measurements used by scanners; however, most of the laser scanners are based on the time of flight principle. This technique allows measurements of distances up to several hundred of meters based on recording the time difference between emitted laser pulse and return signal. Phase shift scanners emit a sinusoidally modulated laser pulse, and calculate distance using a phase shift principle which

leads to determine distance at close range (up to one hundred meters) with accuracies of some millimeters and a higher data rate. The primary differences in terrestrial scanners between time of flight based scanners and phase based are therefore: higher range for "time of flight based" and higher measurement speeds and better precision for "phase based" laser scanners. However, recent advances in technology have improved speed and range capabilities of each system.

2.3.1 Applications

One of the key benefits of laser scanning is that you can collect the data once, but use many times across a transportation agency. A recent TRB project (NCHRP 15-44 Guidelines for the Use of mobile LIDAR in transportation applications; Olsen et al. 2013a) identified various applications of 3D laser scanning in transportation. A recent TRB synthesis also discusses applications, challenges, and benefits of scanning and other geospatial technologies in transportation (Olsen et al. 2013b). Figure 2.5 summarizes current and emerging applications of mobile LIDAR within transportation.

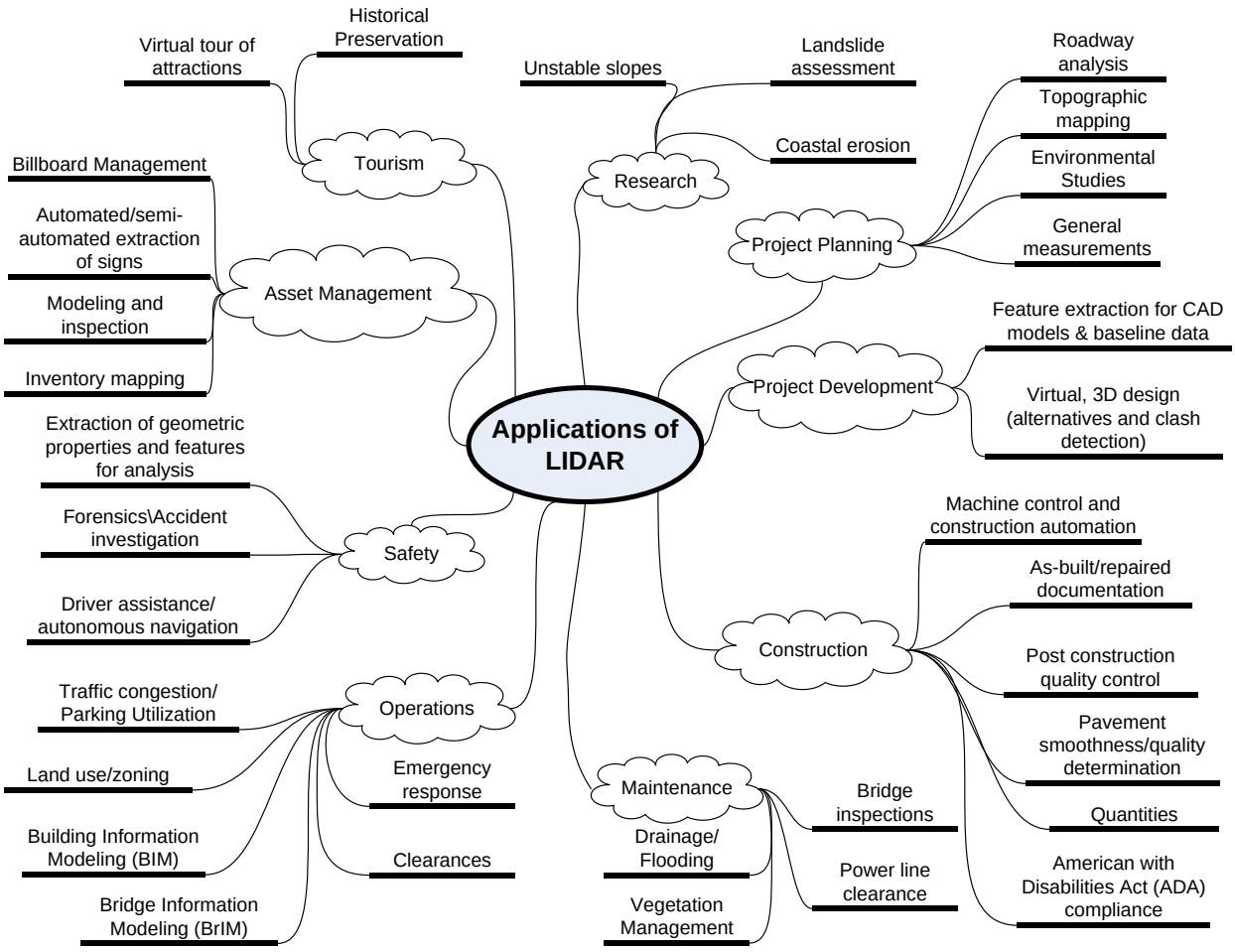

Figure 2.5 Examples of Applications of LIDAR in Transportation Applications (modified from NCHRP Report 748)

2.3.2 Platforms

LIDAR systems can be mounted on several platforms, typically airborne, static terrestrial, and mobile (vehicle platforms). Decreased sensor size and weight is enabling scanners to be attached to unmanned vehicle systems (UVS). Airborne LIDAR is the most mature platform, enabling coverage of large areas; however, its accuracy and resolution are lower than terrestrial-based techniques. Mounting the instrument on a static setup such as a tripod, where it remains for the duration of the scan provides data of the highest accuracy and resolution; however,

coverage is limited compared to the other techniques. In mobile LIDAR systems (MLS), a scanner(s), a GPS receiver, and an IMU (Internal Measurement Unit) are mounted to a vehicle such as an automobile or boat. Due to its ability to capture detailed information along highway corridors rapidly, many transportation agencies are planning on using mobile LIDAR in the near future, if they are not already (Olsen et al. 2013a).

2.3.3 Integration with photography

Unfortunately, the laser scanner is not able to directly capture the color and texture of the measured surfaces, so many scan systems use calibrated digital cameras or video recorders to co-acquire color information, providing greater detail (Toth 2009). Most scan software packages can then map red, green, and blue (RGB) color values to each point in the scan, rendering a more realistic point cloud. This imagery can be used by itself as a video log without the scan data, if needed. McCarthy *et al*. (2008) discuss advantages to using combined LIDAR and photographic information for transportation applications including improved measurements, classifications, workflows, quality control checks, and usefulness. In that study, the scan data was particularly important for measurements on large objects such as bridges and embankments, while the photographs were most helpful for smaller objects.

2.3.4 Challenges

Despite the large benefits, LIDAR presents some challenges, including a steep learning curve, large datasets, equipment costs, and software costs. Another challenge is that because it is a line of sight technique, occlusions will often occur from obstructions.

Chapter 3 Methods

3.1 Data Collection

This project used a 3D laser scanner to collect a point cloud representation of the intersection of 14th and Campus Way on the Oregon State University campus in Corvallis, Oregon. The primary dataset analyzed for this project consists of a scan with a 360^{o} horizontal and a 100^{o} vertical field of view (-40 to +60^{o} from horizon) was obtained, resulting in a dataset of approximately 7 million points. Seven digital images were obtained to form a panorama. Note that a point cloud comprised of several scans at multiple locations would result in a more complete point cloud. However, for the goals of this investigation the single scan was more than adequate.

3.2 Formatting the Data File for the Simulator

After the point cloud data was collected, it was necessary to export the data in the VRML97 format that the simulator could read. However, there are many software packages available for 3D laser scanning data acquisition and processing. Each laser scanner manufacture generally provides its own package. Table 3 shows the supported export formats common 3D laser scan software packages, including Faro Scene, Leica Cyclone, Maptek I-Site Studio, Riegl Riscan Pro, and Trimble RealWorks.

As seen in Table 3.1, some terrestrial scanning software packages can export triangulated objects with texture information directly into VRML file format. In situations that software is not able to export the files in VRML, a third party convertor program can be used such as Meshlab, an open-source model editing program. Simple construction and the well-defined standard of VRML format ensures the availability of a number of modeling tools and convertors. For

example the program “msh2wrl” functions as direct conversion between formats MSH (a Leica *Cyclone* software format for TIN) and VRML.

Table 3.1 Supported Export Formats in Common 3D Laser Scan Packages

SCENE **by Faro:**	***Cyclone*** **by Leica:**	***I-Site*** **by Maptek:**	***RiSCAN PRO*** **by Riegl:**	***Realworks*** **by Trimble:**
.dxf	.dxf	.dxf	.dxf	.dxf
ASCII	ASCII (.txt)	ASCII (.txt)	ASCII	ASCII
VRML (*.wrl)		VRML (*.wrl)	VRML (*.wrl)	-
E57	E57	-	-	-
.ptc	-	-	.ptc	.ptc
-	-	.obj	.obj	.obj
-	-	.dwg	-	.dwg
.pts	.pts	-	.pts	-
.xyz	.xyz	-	-	-
.ptx	.ptx	-	.ptx	-
-	.tif	.jpg	-	.tif
-	.xml	-	-	.xml
.xyb	.coe	.3dv	.3DD	.dgn
.igs	.sdf / .sdnf	.00t	.3pf	.kmz
.pod	.msh	.ireg	.pol	.kml
	.sim	.ma	.ply	.bsf
	.svy	.dxb	.asc	
	.ptz	.3dp		

3.2.1 VRML Format

Texture style and translation/rotation parameters are saved in Geometry and Transformation nodes respectively. VRML stands for “Virtual Reality Modeling Language". Created 3D scenes by VRML can be visualized through the World-Wide Web. Due to its widespread use over the last decade, there are a lot of helpful resources and sample datasets for VRML. As such, herein we provide a basic description, but do not provide full technical details. VRML (*.wrl) files have 3 basic elements:

1.) Header: contains one line telling the browser that the file is VRML and its version,

2.) Comments: A '#' in front of each line in VRML format marks that line as comment, and

3.) Nodes: The important part of VRML format which has everything. There are 9 major nodes in VRML II (Table 3.2).

Table 3.2 Classifications of VRML II Nodes (modified from Web3d.org)

Nodes:	Definitions:
Grouping Nodes	Grouping nodes have a children field and define a coordinate space for its children. This coordinate space is relative to the coordinate space of the node of which the group node is a child. So transformations accumulate down the scene graph hierarchy.
Special Groups	This node has three subnodes: - Inline: for reading children data from a location in the WWW. - LOD: to identify complexity of the given object and - Switch: traverses zero or one of the nodes specified in the choice field
Common Nodes	To specifies audio data, point light source at 3D location in the local coordinate system and Script to program behavior in a scene.
Sensors	To generate events based on user actions, such as a mouse click. This node has the ability to generate events as time passes too.
Shapes and geometry	The Shape node associates a geometry node with nodes that define that geometry's appearance. A Shape node contains exactly one geometry node in its geometry field. This following node types are valid geometry nodes: box, cone, cylinder, ElevationGrid, extrusion, IndexedFaceSet, IndexedLineSet, PointSet, sphere, and text.
Geometric Properties	This node describes objects by Coordinate, Color, Normal, and TextureCoordinate. The geometric property nodes are defined as individual nodes so that instancing and sharing is possible between different geometry nodes.
Appearance	Appearance properties of an object can be defined by this node. The main options are Material and Textures.
Interpolators	Interpolators nodes are designed for linear keyframed animation. There are six different types of interpolator nodes, each based on the type of value that is interpolated: ColorInterpolator, CoordinateInterpolator, NormalInterpolator, OrientationInterpolator, PositionInterpolator and ScalarInterpolator.
Bindable Nodes	The browser maintains a stack for each type of binding node because they have the unique behavior that only one of each type can be active. The node at the top of stack, (the most recently bound node), is the active node for its type and is used by the browser to set world state. Bindable nodes are Background, Fog, NavigationInfo, and Viewpoint.

3.2.2 Color Scale

In most 3D laser scan packages, texture information can be collected using the amplitude of the returned laser beam and digital images (with RGB color) for coloring. In 3D scan packages color can be exported in a variety of ways. Typically, RGB color is exported within three ranges or scales, [0 1], [0 255], [0 65535] depending on the bit-depth allocated. However, in order to display colored point clouds in *SimVista,* RGB colors need to be coded as floating point values from [0 1].

3.2.3 Coordinate System

Selecting an appropriate coordinate systems plays an important role in exporting a file that can be manipulated and viewed in *SimVista*. Typically, there are three types of coordinate systems:

- Scanner coordinate system – all data are referenced to the origin of the scanner when it was collected. These coordinates are typically small since the scanner origin is located at (0,0,0).
- Local project coordinate system – data are referenced to common points between the multiple scans collected in a local system. Typically, these coordinates will range in values of 100's or 1,000's.
- Geo-referenced coordinate system – data are referenced to a real-world coordinate system such as State Plane, UTM, etc. This coordinates are usually in the hundreds of thousands to millions in size.

In attempting to import files into the *SimVista*, the first file was using Oregon State Plane Coordinate System (OSPCS), which is a geo-referenced coordinate system. However, due to the large coordinates, point clouds using coordinates in the OSPCS were extremely difficult to

manipulate in *SimVista*, which fixes the origin at (0,0,0). Further, coordinates are truncated at the meter level for floating precision (7 digits), leading to significant display artifacts.

The state plane coordinate system is designed to deal with large distances (half of the state of Oregon), and thus the coordinate values for locations in Corvallis are also large. A location in Corvallis might have an OSPCS designation of (104255.321 N, 2279548.465 E, in meters). When the OSPCS was used, data imported into *SimVista* was located about of 100 kilometers away from the scenario origin in the north-south direction and 2280 kilometers in the east-west direction. This caused great difficulty in translating and rotating the point cloud data to create a scenario, and often exceeded the boundary limits of *SimVista*.

To address this situation, the scanner coordinate system was first used. Using a scanner coordinate system set the location of the laser scanner as (0,0,0) and moved the point cloud much closer to the origin when imported into *SimVista*. This allowed researchers to more easily manipulate the data to create a simulation scenario. However, a limitation of this approach is that it only allows one to use data from one scan position or to transform other scans into the coordinates of the reference scan. Further, orientation (rotation) of the data relative to north and the level plane are not present in the scanner coordinate system.

Another fix is the truncation of the point clouds by a fixed amount (e.g., utilizing only the last four or five digits of the coordinates), which still allows for easy manipulation within a typical site. Additionally, this would provide a common reference frame so that multiple scans could be integrated into a single scene. However, care should be taken to document the truncation so that the data can be linked quickly back to their original coordinates. The preferred method would be to apply a transform at the beginning of the VRML file by including the following:

```
Transform
{
    translation -2270000 -104000 0
    rotation 0 0 0 0
}
```

where the translation vector should be adapted to the coordinates of the particular dataset.

3.2.4 Rotating the Point Cloud

Another challenge in importing the point cloud model into *SimVista* is that engineers and computer graphic designers use different coordinate conventions. Computer graphic designers typically define the Z-axis as pointing out of the screen, and engineers typically define the Z-axis as elevation. ISA follows the computer graphic designers' definition of the Z-axis, while most laser scanners use the engineers' definition. This causes the point clouds imported into *SimVista* to be rotated sideways upon import.

In order to resolve the difference between the axes in *SimVista* and the laser scanner 3D point clouds, two methods were tested. In the first method, Y and Z directions from the point cloud were switched and the Z values were made negative to conform to the ISA definitions. Unfortunately, the results of this time-consuming method did not completely solve the problem. In the second method, a transformation node is created in the VRML file and initiated using the following code:

```
Transform
{
    translation 0 0 0
    rotation -1 0 0 1.57
}
```

Orientation of an object is defined by yaw (around the y axis), pitch (around the x axis) and roll (around the z axis). In VRML format, the Y-axis or the Z-axis can be consider as up

direction. In the first line, each value shows the amount of translation in X, Y and Z direction while the second line defines the rotation around the X, Y and Z axis (yaw, roll and pitch) by the amount or magnitude of rotation, in radians (last value).

This transformation allows the point cloud model to be imported into *SimVista* while conforming to the axes definition used by ISA, making user-interaction and manipulation of the point cloud much simpler.

3.3 Importing the Data File into the Simulator's Scenario Editing Tool

Once the point cloud model is in VRML format and the proper transformations have been applied, further manipulation of the VRML files is needed to create a scenario for us in the driving simulator. As an initial step, a small point cloud model was imported into the simulator. For this initial investigation, a stop sign and surrounding ground was isolated from the larger point cloud model and examined.

3.3.1 Creating a Scene with a Point Cloud Object

The point cloud .wrl file was opened using *SimVista* and saved as an object (Figure 3.1). This allowed the point cloud to be manipulated in a scenario with other element from *SimVista.* Next a generic ground tile is opened in *SimVista.* The point cloud object is then placed on the ground tile as any other object would be in *SimVista.* Once the scenario is complete the file must be published for use in *SimCreator* and the driving simulator. The published .wrl file will load and display properly in the driving simulator (Figure 3.2).

The work process to import a small point cloud for use in the driving simulator is described in Figure 3.3.

Figure 3.1 Point Cloud of Stop Sign Saved as an Object

Figure 3.2 Point Cloud Object in Driving Simulator

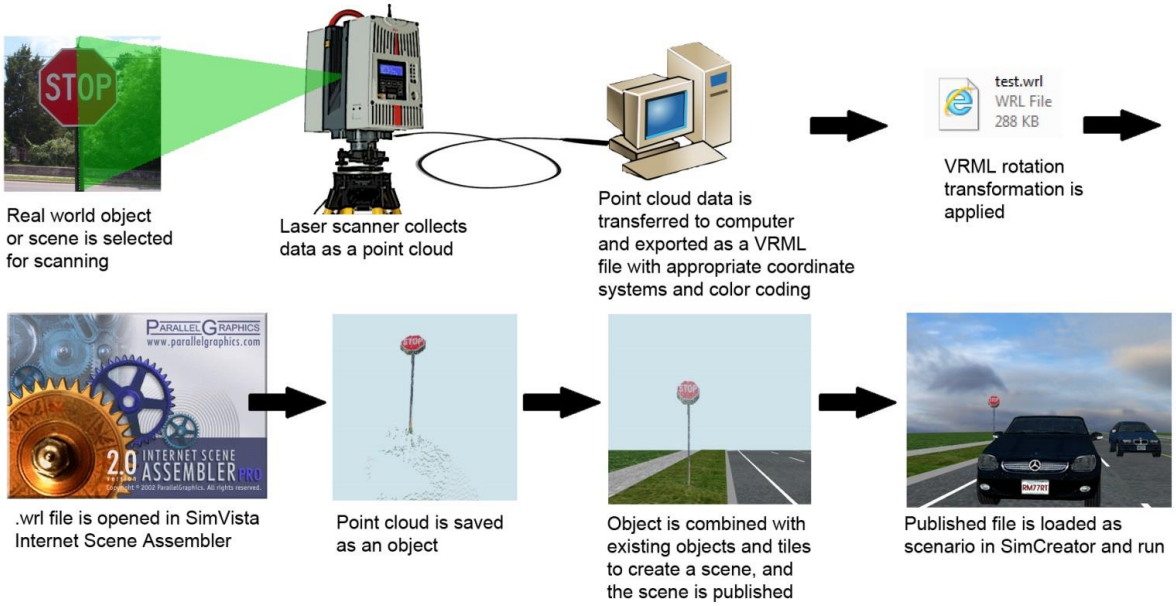

Figure 3.3 Work Flow to Import Small Point Clouds

3.3.2 Creating Larger Scenes

Having proof that point clouds could be imported into the driving simulator in a drivable environment, it was next desired to determine size limitations for larger datasets.

The same process as describe in Figure 3.3 was attempted using the entire point cloud model from the scan of 14th and Campus Way. This file was considerably larger (450MB with 3,419,357 points) as compared to the stop sign (394KB with 2,982). Although this file was able to load into *SimVista*, attempts to publish this took several hours, and often crashed *SimVista*. When *SimVista* was able to publish a scenario of the full 450MB intersection point cloud model, *SimCreator* was unable to load and run the file in the driving simulator. This file was simply too large to be handled by the simulator software. The next chapter will discuss methods implemented to overcome these challenges.

Chapter 4 Methods Optimization

Having established that small point clouds, on the order of 400KB, could be imported with relative ease, but files on the order of 450MB were impossible to import in the current software version, the next task was to develop a strategy to optimize the performance of the simulator using the largest of a point cloud environment possible. Several strategies were examined for controlling the size of the point clouds and optimizing performance in the simulator.

4.1 Point Reduction

The first and simplest way to reduce the size of the point cloud was to remove any extraneous points. For example the objects visible by line of sight to the scanner through a window in a building were captured in the initial scan. However, these object server no practical purpose in modeling the scenario and were removed.

Additionally, the method being used to model the point clouds in the simulator places objects from the laser scan on a pre-existing ground tile, making much of the terrain information in the point cloud extraneous data. Removing unneeded features, although time consuming, significantly reduces the size of the files.

4.2 Creating a TIN

Another method for controlling the size of the point files was to create Triangular Irregular Network (TIN) from the point clouds. Two TIN models in VRML format were created from the same point cloud model of the intersection of 14th and Campus Way. The TIN models represented Kearney Hall, a building located near the intersection, and a stop sign at the intersection. One of the TIN models contains only triangular surfaces, while the other TIN model

was a combination of triangular surfaces and point clouds. Each of the files had an approximate size of 17 MB.

Although the TIN models were able to load in the driving *SimVista* and *SimCreator,* they would not display correctly. The simulator displayed the TIN models as a collection of a few scattered triangles, not forming any real shape or pattern. Although further manipulation of the TIN models might have created versions that would display correctly, this effort was ultimately abandoned due to the significant processing required for TIN models.

4.3 Sensitivity Analysis

A sensitivity analysis was performed by splitting a large point cloud scene in to smaller partitions to examine the capability of simulator. A 275 MB point cloud model, with 2.1 million individual points, was divided into four partitions. Table 4.1 shows size information about each partition, the complete model, and the other files examine in this project.

Table 4.1 VRML files used in Project

File Name:	Size:	# of points:
Stop Sign	394 KB	2,982
Complete Model	450 MB	3,419,357
Partition1	37 MB	283,700
Partition2	37.5 MB	284,450
Partition3	35 MB	269,630
Partition4	166 MB	1.27 million
Complete model	275 MB	2.1 million
TIN models	17 MB	272,500

Initially, the *Complete model* (275 MB with 2.1 million points) was published as a scenario. The simulator was unable to load and run this file. Next, all four partition files were placed on a tile and published as a scenario. When this scenario was loaded into the simulator,

the system ran, but with significant lagging. When *Partition4* (166 MB with 1.27 million points) was removed, the scenario the simulator ran without lag. Then, scenarios were created with multiple sets of partitions 1-3 to examine how many objects that were 35MB in size could be run in a scenario without lag (Figure 4.1). With a total of 12 objects, approximately 35MB each, the system ran without lag. Note that the total size of these 12 objects is larger than the *Complete model* showing that many smaller point clouds objects run more efficiently in the simulator than a single large point cloud object. When the number 35MB objects were increased to 18 objects, the system began to show lag.

Figure 4.1 Simulator Screen Shot from Environment with Twelve 35MB Point Clouds

This same process was examined at its logical extreme. A simulation scenario with enormous of small point clouds, 1800 of the 394KB stop signs, was created and published (Figure 4.2). This scenario ran without lag, although *SimVista* began having trouble dealing with the 1800 individual objects. The total size of these 1800 objects is larger (~710MB total) than the any other scenario tested. This again shows that a scenario of many small point clouds objects run more efficiently in the simulator than a single large point cloud object. When the number of stop signs was increased to 2200 the system began to lag.

Figure 4.2 Simulator Screen Shot from Environment with 1800 Stop Sign Point Clouds (394KB each)

Since the simulator software is more capable of processing many small files more than one large file, it can be concluded that it is optimized to represent a complex scene with several smaller partitions. The ideal size to make each partition is not a set value, but must balance the total size of the point cloud, the additional work required to create the partitions, and the performance of the simulator.

Chapter 5 Conclusions and Recommendations

Accurate 3D point cloud models of real world sites can be imported into a high fidelity driving simulator. Although the challenges in this process will be different for each driving simulation platform, the basic tasks remain the same, and many of the methods used to address challenges across platforms will be similar. In order to interface between the laser scanner data and a high fidelity driving simulator one must perform three tasks.

1. Export the point cloud in a format that can be introduced into a driving simulator
2. Import the point cloud into the simulator's scenario editing tool in a way that allows manipulation and scenario design
3. Optimize the performance of the point cloud scenario in the driving simulator

A variety of software exists to address the exporting of point cloud model into several formats. This makes the first task simply finding the correct software to export or convert the point could model into the desired format.

The challenges associated with second task will be different depending on the simulation platform used. Examining the OSU Driving Simulator running the software *SimVista* and *SimCreator* from the vendor RTI, the following suggestion will facilitate the creation of scenarios from point cloud models.

- ✓ VRML97 is the only format that can be imported into RTI driving simulators. Any other format must be converted into VRML97.
- ✓ The color scale must be in floating point precision values from [0 1]
- ✓ Creating a transformation node is an effective, and sometimes necessary, way to translate and rotate the point cloud model before importing into *SimVista*.

- ✓ Using scanner coordinate system brings point clouds near the origin in *SimVista* making further manipulation simpler. When using multiple scans a truncated state plane coordinate system will allow for the easy manipulation of data while providing a common reference frame. However, using a transformation node in the VRML file is the preferred method.

Finally, the task of optimizing the performance of a point cloud scenario is of paramount importance. A driving simulator is pointless if it cannot accurately reflect the real world, and the real world does not lag. The two most effect tools for combating system lag in the OSU Driving Simulator are:

- ✓ Removing any extraneous data points from the model to control total file size. This can be accomplished by filtering points far from the scanner (by range) and areas of high point density close to the scanner (by minimum separation).
- ✓ Splitting large files into several smaller partitions that are easier for the simulator to process. Although each driving simulator's computer system is unique, partitions smaller than 35MB work best with the OSU driving simulator.

This study has shown that importing point clouds in a driving simulator is feasible, but more work could be done in this area.

- ✓ In this project, data from a single scan position was used. In order to create a full 3D model, more scan positions should be used. In this case, one would want to use a project coordinate system with coordinates of (0,0,0) at the center. Alternatively, one can use the transform node function for geo-referenced data.

- ✓ Data from mobile LIDAR and imaging systems captured along a highway corridor can be integrated into the simulator.
- ✓ Validation is always a concern when using a driving simulator. The uses of point cloud scenarios should be validated against other simulator models and the real world.
- ✓ Other driving simulator platforms could be examined to identify and address the unique challenges associated with importing point cloud models in different simulation platforms.
- ✓ A standard format for driving simulators scenarios should be developed to allow interoperability of data between driving simulators. This format should be a structured, binary format for optimal loading.

References

Fisher, Donald L., Matthew Rizzo, and Jeff K. Caird. 2011. "Handbook of driving simulation for engineering, medicine, and psychology." CRC Press.

Hurwitz, David S., Tuss, Halston, Olsen, Michael J., Roe, Gene V., and Knodler, Michael A. 2013. "Transportation applications for Mobile LIDAR scanning: A state-of-the-practice questionnaire," Transportation Research Record Annual Meeting, CD-ROM.

McCarthy, T., Zheng, J., and Stewart, F., 2008. "Integration of dynamic LIDAR and Image Sensor Data for Route Corridor Mapping," ISPRS Congress

McNamara, Phil. "Inside Toyota's $15m driving simulator" Car: The World's Best Car Magazine. 2009 <http://www.carmagazine.co.uk/Community/Car-Magazines-Blogs/Phil-McNamara-Blog/Inside-Toyotas-15m-driving-simulator/> Accessed 5/8/2013

Olsen, Michael J., Roe, Gene V., Glennie, Craig, Persi, Fred., Reedy, Marcus., Hurwitz, David, Williams, Keith, Tuss, Halston, Squellati, Anthony, and Knodler, Michael A. 2013a. "Guidelines for the use of mobile LIDAR in transportation applications," TRB NCHRP Final Report 748, 250pp.

Olsen, Michael J., Roe, Gene V., and Raugust, John. 2013. "Use of advanced geospatial data tools, technologies, and Information in DOT projects", NCHRP Synthesis 446, Topic 43-09, 87 pp.

STISIM. "M100 Series Simulation Systems" <http://www.stisimdrive.com/products/simulation-systems/m100-series> Accessed 5/8/2013

Oregon State University. Driving Simulator. Retrieved from Driving and Bicycling Research Lab. 2011 <http://cce.oregonstate.edu/research/drivingsimulator/DrivingSim.php> Accessed 5/24/13

Schneider, Daniel K., Martin-Michiellot, Sylvere. 1998. "VRML Primer and Tutorial", University of Geneva.

Toth, Charles. K., 2009. "R&D of mobile LIDAR mapping and future trends." In Proceeding of ASPRS 2009 Annual Conference (Baltimore, Maryland).

Web3d. "The Virtual Reality Markup Language: Concepts" <http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/part1/concepts.html> Accessed 10/29/2013.