LiDAR-Based Virtual Environment Generation for Vehicle Digital Twins

Aleksi Vuorinen[1], Ella Peltonen[1]

[1]Empirical Software Engineering in Software, Systems, and Services, University of Oulu, Finland

Abstract

Digital twins and LiDAR technologies are significant topics of interest among many industries today, including the vehicular vertical. For example, automotive manufacturers can use digital twins to analyze product performance. LiDAR sensors are used in many autonomous driving systems today. This, along with the increased use of digital twins for testing purposes, has created a need for mapping real-world environments so that they function as digital twins in a simulation environment. However, software pipelines and workflows have not kept up with this development. This paper provides a preliminary pipeline for transforming LiDAR point clouds to be used in a simulation environment for vehicle digital twins. We reconstruct 3D mesh models from mobile laser scanning obtained point clouds using three popular software solutions, Cloud Compare, MeshLab, and Metashape, with their respective reconstruction algorithms, Poisson surface reconstruction, ball pivoting algorithm, and marching cubes, and validate the results of each tool. We show that LiDAR can create a model of a target environment with its well-preserved physical features. In addition, we discuss challenges that need to be overcome to thoroughly polish the pipeline of turning raw sensor data into simulation-ready 3D models.

Keywords

LiDAR, Vehicular Computing, Digital Twins

1. Introduction

Modern cars can be referred to as software products. At the same time, the development of digital twin technology has created potential use cases in the automotive industry [1]. Virtualization, such as using a virtual replica of an actual vehicle and its environment, can be used to represent their real-life counterparts for validation and testing purposes. Accurate simulation models can provide a versatile testing environment regarding vehicles and human safety. They are cost-effective and safe under different conditions and can be used to gather valuable data without wasting excessive resources [2]. The automotive industry has also seen great potential in self-driving cars and advanced driving assistance systems (ADAS), with companies such as Mercedes-Benz and Honda using LiDAR for their respective autonomous driving systems [3, 4]. Indeed, virtualization capabilities are already seen as critical in safe implementations in the vehicle software stack and software-defined vehicle [2]. Regarding the safety aspect of vehicles, and therefore human safety, the simulated data must be as accurate as possible compared to reality, both regarding the environment and the vehicle itself. Thus, the means of creating virtual testing environments or digital twins must accurately represent the varying aspects of reality, which, in this case, are real driving environments – more accurately than traditional satellite-based mapping tools can achieve.

For different 3D modeling purposes, LiDAR technology has been increasingly popular among researchers and various industries over the past decade. The LiDAR devices today can effectively produce dense point clouds of a target environment with a range up to hundreds of meters. However, scanning the environment alone is not enough; scanned data has to be transformed into an interactable digital twin. Until now, based on the best of our knowledge, there has been no systematic workflow and best practices for mapping LiDAR data into a virtual environment. Unlike many studies that rely on publicly available, professionally created datasets, our study is conducted in an uncontrolled, real-world outdoor environment, which more realistically reflects the challenges typically encountered in LiDAR scanning.

TKTP 2025: Annual Doctoral Symposium of Computer Science, 2.—3.6.2025 Helsinki, Finland

aleksi.vuorinen@oulu.fi (A. Vuorinen); ella.peltonen@oulu.fi (E. Peltonen)

0009-0005-2125-5939 (A. Vuorinen); 0000-0002-3374-671X (E. Peltonen)

These challenges include environmental noise, such as people or vehicles, and the variability of point cloud density depending on range and surface properties around the scanned environment.

In this paper, we investigate three different tools, each containing a distinctive algorithm for virtual reconstruction. This workflow consists of scanning point clouds in natural environments, creating a solid mesh out of them, and integrating the environment into a game engine for virtual testing. We use the GeoSLAM ZEB Horizon LiDAR device equipped with the ZEB Vision camera attachment (see Figure 4). The Unreal Engine 5 game engine is used to function as a platform for the virtual environment.

We consider the appropriateness of the reconstructed models by assessing the performance of the mesh generation algorithms. We use three state-of-the-art software products and compare the outcome of the models. The main contribution is to explore the possibilities of using a mobile LiDAR scanner to create virtual environments using state-of-the-art tools and their respective reconstruction algorithms provided to test vehicular digital twins. These emerging results will contribute towards advancements in software-defined vehicles and their virtualization capabilities. Our contributions can be summarized as follows:

- We define a workflow for turning a LiDAR-scanned point cloud into a 3D mesh model integrated into a digital twin.
- We compare existing methods of virtually reconstructing point clouds into a realistic mesh model.
- We validate the results and define a workflow for creating a LiDAR-based simulation environment from self-captured scans.

2. Background

LiDAR is an advanced active remote sensing technology that uses laser scanning, making it possible to obtain 3D point clouds of a target object or environment in the real world [5]. This technology has come a long way, and in addition to the automotive industry, it has been used for a wide variety of different purposes such as landslide investigations [6], measuring snow depth [7], creating a digital twins of city objects [8], and covering archaeological remains through dense jungle [9]. Depending on the area and purpose, LiDAR scanners can be handheld or mounted on Unmanned Aerial Vehicles (UAVs), vehicles, or tripods. LiDAR technology has turned out to be an excellent solution for examining the earth's surface and detailing a range of geospatial information [10].

LiDAR systems typically consist of four subsystems: range finding unit, beam detection unit, power management unit, and master controller unit, as seen in Figure 1. The core of the LiDAR system is the range-finding unit, which contains the components required to generate, transmit, and receive short laser pulses. Within this unit, electronic components such as laser diode, photodiode, trans-impedance amplifier, and time-to-digital converter can be found [11]. For the scanner to map its environment with ground truth-equivalent dimensions, it calculates the time it takes for the emitted laser pulse to reflect off an object and return to the scanner; this is called time-of-flight, and it is how the distances are measured with laser pulses. In addition to the previously mentioned components, the range-finding unit also contains optical elements such as the collimating lens and focusing lens, which reduce the divergence of the transmitted laser beam and focus the received beam into the detector. The beam deflection unit deflects the laser beam transmitted to obtain 2D or 3D spatial information, instead of the LiDAR functioning as a 1D LiDAR, also called a laser range finder. The job of the control unit is to handle signal processing, signal generation, and communication with the LiDAR software or processing unit within the device.

Point cloud is a set of points carrying geometry and attribute data in a 3D space [12]. The "cloud" in point cloud refers to the unorganized cloud-like nature that the points together form. Point clouds can be either static or dynamic. In the case of dynamic point clouds, a different point cloud is issued at every frame, resulting in a continuous variation with constant change in the number of points. The geometry data of the points refers to the position in a given Cartesian coordinate system, expressed in X, Y, and Z coordinates, usually in floating-point values, but can also be expressed in integers for faster computation time and memory optimization, depending on the use case. The attribute data refers to the

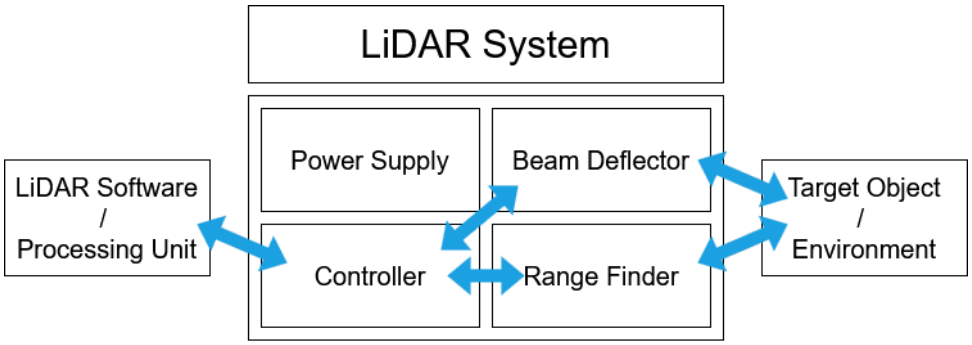

Figure 1: Illustration to visualize the main subsystems of a LiDAR.

visual appearance of the point, which can also vary depending on the use case. Some commonly used values for attributes are, for example, colour values (R, G, B), in case the LiDAR device has a camera attached, normal vectors (nx, ny, nz), and intensity values based on the strength of the laser return signal, which might have a range of 0-255 (8-bit), 0-65535 (16-bit), or simply, 0-1.

Reconstruction in the context of point clouds essentially means turning the point cloud into a 3D polygonal mesh. Mesh is a collection of triangular or quadrilateral faces that do not overlap with each other and are connected along their edges [13]. A mesh, therefore, contains vertices, edges, and faces, and the most straightforward representation of a mesh can be presented as a single face. Point clouds do not have an inherent structure compared to a mesh, and the points are in no way connected. While both mesh and point cloud represent 3D geometry, meshes offer a far more structured visual presentation of a 3D model and a more well-defined topology for performing geometric operations and collision detection for realistic interactions between objects, which are necessary for simulative purposes.

3. Key Challenges

We identify three open challenges in obtaining the highest quality point cloud and virtually reconstructing it to preserve the environment's physical features. The first key challenge is **quality data acquisition**. LiDAR devices are often combined with a traditional camera to address the challenge of tracking and defining different real-world objects and to enable the colorization of the points. Even with the supplementation of a camera, outliers appear within the point cloud. The quality of the preliminary point cloud often corresponds to the quality of the sensor and/or camera used. Things such as deviation of the camera, lighting, reflections, and measurement noise affect the outcome of the raw point cloud [14]. Filtering algorithms eliminate most noise and outliers from the raw point cloud while preserving the desired features within the scan. Still, getting a noise-free scan of an area with people, vehicles, or other moving entities is virtually impossible.

When working with point clouds, depending on the area of the surveyed environment, the second challenge is **process demands high computational resources**, as data size and processing times are demanding. Some LiDAR devices equipped with high-resolution cameras and scanning colored, highly dense point clouds from large areas result in a large amount of data to process. In addition, transforming point clouds of real-world environments, especially large spaces, to mesh models is computationally demanding because the high number of points naturally results in a high vertex count. This is because surface reconstruction methods require constructing spatial data structures whose complexity scales with the input size: the larger the area, the more there is to process. Our LiDAR scans, exported as raw point clouds, are typically between 500 to 2000 MB, depending on the scale and complexity of the environment. If the scans were done by surveying larger environments using a vehicle, the file sizes would increase even further.

The third and likely the most central challenge is **point cloud reconstruction**. Because point clouds lack complete surface coverage, additional information is typically required to fill in the gaps for

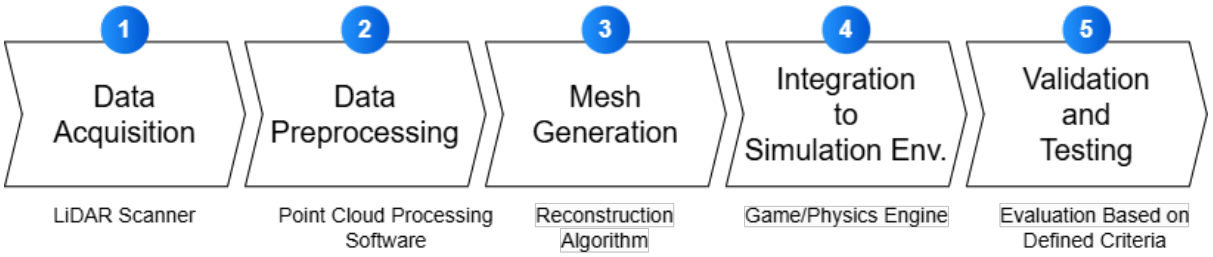

Figure 2: Workflow from a LiDAR-scanned point cloud into a 3D mesh model.

converting the point cloud to mesh has to be generated [15]. Some state-of-the-art mesh reconstruction algorithms that we also experiment with include marching cubes [16, 17], ball pivoting [18], and Poisson surface reconstruction [19]. In practice, these perform the interpolation required to turn a point cloud into a more structured mesh model. However, as later seen through our artifact evaluation, the process of mesh model building is far from easy to simplify. Common issues with reconstructed mesh models from point clouds include issues in producing a quality point cloud with the right amount of triangles. Problems commonly emerge with over-triangulated flat surfaces, making the data far heavier than necessary [20]. At the same time, insufficient triangle density is also a problem [21].

4. Methods and Materials

This paper uses a constructive research approach [22] to create an artifact and obtain answers for the research questions that focus on technical real-world problems, i.e., the creation of a simulation environment by turning a LiDAR scanned point cloud into a 3D mesh that would be suitable to be used as an environment for vehicle digital twins. The workflow is presented in Figure 2, where obtaining the point cloud is described through its transformation into a 3D model imported into the simulation platform when it's finally ready to be evaluated. The target of contribution of the research is the automotive industry, specifically in the context of digital twins and simulation. The obtained results during the development of the artifact can, in addition to the automotive applications, help in future research regarding the development of simulation environments.

4.1. Data Acquisition

We use the ZEB Horizon handheld LiDAR scanner, as shown in Figure 4. According to the manufacturer, it can scan 300,000 points per second with up to 6 mm relative accuracy and has a range of up to 100 meters. As for the hardware of the development PC used, it consists of AMD Ryzen 9 7900X, Asus GeForce RTX 4070 DUAL - OC Edition, Kingston FURY Beast - DDR5 32GB:2 x 16.

For an example environment, we choose an open outdoor space with enough features to ensure we acquire a high-quality scan while still allowing space for a vehicle to move around. For a scanned feature to be considered significant, its size-to-range ratio should be roughly 1:10 [23]. Ideally, the location should contain objects with structural variety, such as buildings, that help the scanner pick up features, which our chosen location fulfilled.

4.2. Data Preprocessing

After obtaining the data from the scan, it is processed and converted into the correct format using the LiDAR software FARO Connect Viewer. The data is constructed in a custom file format determined by the manufacturer, which means that, as is, other tools cannot use the data without parsing. In the process, there are options for the software to automatically filter noise by setting the environment type of the capture area. We chose default settings, as selecting any other area type could potentially remove some of the proper data points, which could slightly reduce the overall quality of the point

Figure 3: Image of the raw point cloud.

Figure 4: The setup with ZEB Horizon.

cloud. Minor inconveniences, such as tiny disruption points caused by bystanders, can be manually removed to preserve the point cloud quality.

Processing the data is one of the biggest bottlenecks regarding time efficiency. One flaw that slows down the process is that the images taken by the camera have to be processed by the LiDAR software, which is the most time-consuming part of the process. Using the software, it is impossible to only get the color data from the camera without stitching the images along the trajectory. This procedure can take up to 45 minutes for a single scan, depending on the point cloud size. When the application successfully processes the point cloud, it can be inspected in a 3D view. The FARO Connect Viewer software is used to automatically filter the point cloud from noise in a given environment, inspect the point cloud, and export it in the desired format.

There were only minor bits of noise around the entrance of the building, which might have been caused by the small amount of snow around that area; these are very difficult to avoid, however, as the scanning device does not show the registration of points in real-time so that the regions with weaker data could be complimented for during the scanning. In total, the point cloud contained a high number of 44,143,072 points. The chosen format for the file to be exported is E57, a compact and vendor-neutral format for storing point cloud data specified by an international standards organization ASTM [1]. This format is commonly supported across different software. It supports a wide variety of other metadata, such as panorama images captured by the ZEB Vision camera attached to the LiDAR device, in case they were needed for reconstruction while remaining moderate in file size.

5. Mesh Model Generation

We experiment with three different tools that are popularly used within the industry and have extensive documentation and community support, making them worth testing before implementing a new solution. CloudCompare, MeshLab, and Metashape each offer different meshing approaches, including Poisson Surface Reconstruction, Ball Pivoting, and Marching Cubes, which allows us to compare how different techniques affect the reconstruction quality.

5.1. Reconstruction Algorithms

Poisson Reconstruction, or more formally, the Poisson surface reconstruction method, uses a function-fitting approach to solving the reconstruction problem [24]. The central concept behind this method is that it reconstructs the indicator function, which has a value of one inside the surface and zero outside. The idea is to utilize the fact that a set of oriented points can be seen as a sample of the gradient of the indicator function. Solving the indicator function then involves finding the scalar function whose gradient best matches the vector field, which is achieved by solving the Poisson equation.

[1] http://www.libe57.org/

An adaptive, multi-resolution basis is used to represent the solution for the Poisson equation because the indicator function and its gradient only contain high-frequency information around the solid surface. This involves adapting an octree to the point samples and defining a function space by associating a tri-variate B-spline to each octree node. These B-splines are translated to the node centre, scaled by the node size, and then used to define the function space over which the Poisson equation is solved. A finite-elements approach is used to solve this equation, where the system is discretized by using the B-splines as test functions. The system is then solved by finding the function that satisfies the equation for all nodes in the octree.

Ball Pivoting Algorithm starts with a seed triangle, pivoting a ball around the edges of the boundaries of the current mesh until the ball hits a new point [18]. The edge and point then define a new triangle, which is added to the current mesh. The algorithm then considers a new boundary edge for the continued pivoting.

The BPA follows the advancing-front approach to construct an interpolating triangulation incrementally. The algorithm takes a list of surface-sample data points pi as input, each associated with a normal ni and a ball radius r. The algorithm finds a seed triangle that consists of three data points such that a ball of radius r touching them does not contain any other data points. While keeping in contact with two of the data points of the seed triangle, the ball pivots around it until it touches another point, adding one triangle to the mesh at a time. A "front" is a collection of linked lists that keep track of the edges of the triangles. Each edge in the front contains information about its endpoints, the opposite vertex, and the centre of the ball, which touches all three endpoints and links to edges next to it in the same loop. The edges can have a different status based on whether they are used for pivoting. The front is a collection of linked lists instead of a single one because it changes shape as the algorithm progresses. BPA uses join and glue operations to ensure that the front is always a set of linked lists, even as the algorithm progresses.

The marching cubes algorithm starts with a three-dimensional grid where each point has a value. The algorithm includes two steps: estimating the number and connectivity of triangles in each field cell by identifying regions within the cell where the value goes beyond a certain threshold. Then, once the algorithm has identified these regions, it predicts the vertex locations of the triangles to best present the surface within the cells, which then determines the object's geometry [16].

The algorithm was originally presented by Cline & Lorensen [17], and the paper summarizes the method in the following way: four segments are read into memory, and two of them are scanned to construct a cube, leveraging four neighboring points on one slice and four on the other one. A unique identifier is then calculated for the cube by comparing the density values at its eight vertices with the surface constant. The unique identifier is employed to reference a table containing a list of edges, and using the densities at each edge vertex, a surface edge intersection is found using linear interpolation. Finally, unit normals are calculated at each cube vertex using central differences and are subsequently interpolated to each triangle vertex, resulting in the output of triangle vertices and vertex normals.

5.2. The Reconstruction Tools

Cloud Compare [25] is a free-of-cost and open-source 3D point cloud processing software with a plugin for Poisson surface reconstruction. To apply the Poisson reconstruction algorithm, surface normals for the point cloud must be computed first. Obtaining good results from computing the normals, meaning most of the surface normals would have to be pointing in the right direction, becomes increasingly difficult as the point density increases.

The software has different settings and parameters that can be adjusted to compute the surface normals, each having their respective use cases, such as working with noisy scans, sharp edges, or curvy surfaces [26]. Cloud Compare also allows for selecting how the local neighbors are chosen and whether the normals point inside or outside the object. After several attempts to generate the mesh, the most satisfactory results were obtained plane as the local surface model when computing the surface normals. This is a setting in Cloud Compare for an estimated local surface represented by a point and its neighbors. The parameters were set with the neighboring number (k-NN) of eight and an octree

Figure 5: Reconstructed mesh model with Cloud Compare.

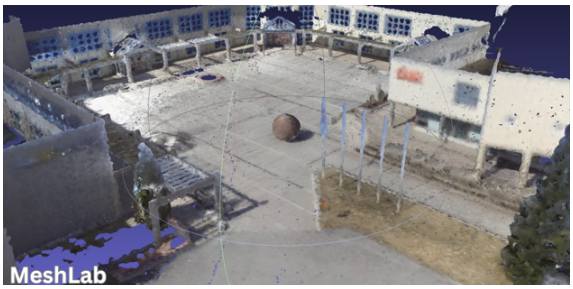

Figure 6: Reconstructed mesh model with MeshLab.

radius of 0.06, meaning that eight is the maximum number of neighbors connected to each node.

The Poisson surface reconstruction can be run after the normals are computed for the point cloud. Cloud compare allows for the output density of the mesh to be set as a scalar field, which enables the output mesh extents to match the determined input. This was useful for working with a point cloud of an outside environment, as the Poisson surface reconstruction algorithm works naturally better on closed shapes. Running the Poisson Surface Reconstruction with an octree depth of 11 produced the best results, as going any lower made the mesh overly smooth, and increasing the value, on the other hand, considerably slowed down the system. The resulting mesh was nearly watertight and sharp in detail. However, due to the sharpness of the mesh, the model still had an amorphous structure in places where bits of missing data were present, with many imperfections emerging on the ground surface, especially near the entrance of the building, which could be concluded to have been caused by a circular structure on the ground that the LiDAR Scanner did not entirely register during the scanning.

MeshLab [27] is another free and open-source software for working with meshes while also providing support for point clouds with tools for surface reconstruction. MeshLab was chosen as one of the tools as it includes the Ball Pivoting Algorithm, which gave us three different surface reconstruction algorithms to compare with each other. Like in Cloud Compare, the normals must be computed before running the reconstruction on the point cloud, for which we chose a neighboring number of 10 as a value.

Attempting to reconstruct the point cloud directly seems too demanding of an operation with the ball pivoting algorithm. This means that the point set must be optimized by creating a simplified version of the point cloud for the algorithm to be run. We chose a sample number of 1,000,000 for the simplification, which created a new point cloud of only 804,420 points. The ball pivoting algorithm was run on the model with the following parameters: ball radius auto-guessed by MeshLab, a clustering radius of 20%, and an angle threshold of 90. The chosen sample number appeared to be a reasonable value, as the reconstruction using the ball pivoting algorithm took a grand total of 49 minutes, even with the simplified cloud. Despite the sparse cloud after simplifying, the ball pivoting algorithm preserves even the fine features of the environment well. However, due to the sparsity of the point set, the model is not as smooth on its surfaces. This would likely require some adjusting of the ball radius to get the best results. The ball pivoting algorithm could be more suitable for smaller areas, as it seems to create a well-recognizable mesh model considering such a low vertex and face count, with the only significant issue being the inconsistency of the surface.

Agisoft Metashape [28] is the third and final tool we experimented with. The whole process of importing the point cloud into the software and building a mesh model out of it took slightly longer than the other tools used earlier, but the difference was insignificant. The major difference regarding the process, as opposed to Cloud Compare, was that Metashape supposedly seems to compute the surface normals during the importation, and by looking at the console logs, it can be confirmed that Metashape uses the marching cubes algorithm to generate the mesh model from the point cloud. With Metashape, the user cannot affect the point cloud reconstruction process by changing any parameters.

The marching cubes algorithm excels on ground surfaces with very little noise. However, imperfections emerged with thin objects such as poles, pillars, and trees, causing them to look amorphous and

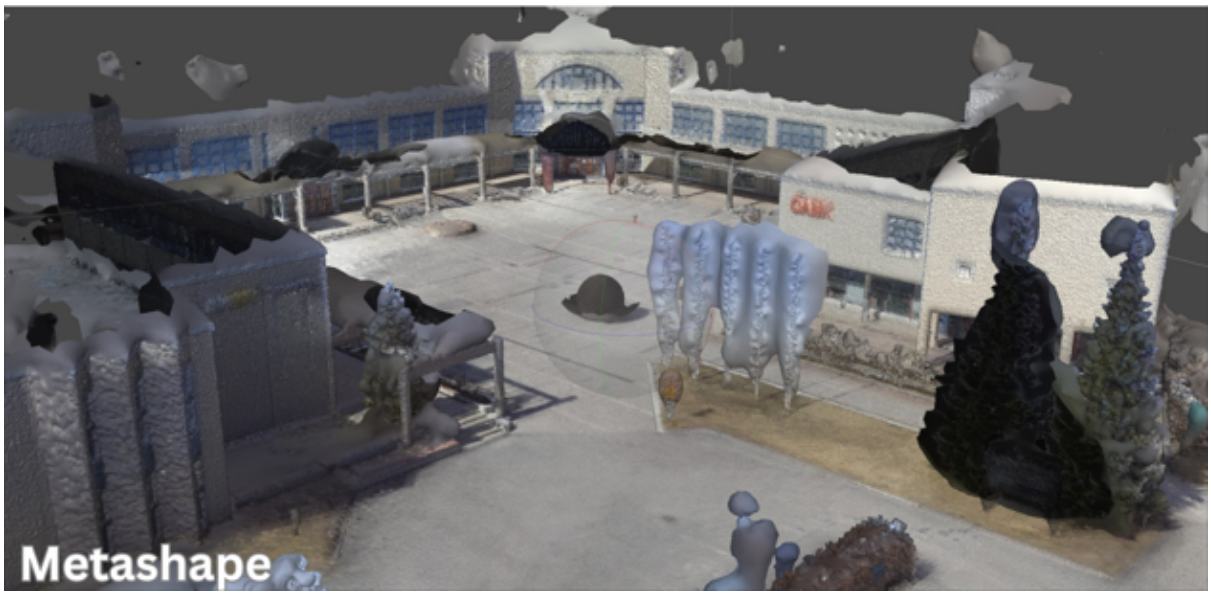

Figure 7: Reconstructed mesh model with Agisoft Metashape.

over-smoothed. This may indicate that the marching cubes algorithm may be more suitable for dense, low-noise point sets, as it greatly preserves well-registered features while having problems with more sparse areas. The marching cubes algorithm has great potential, especially if some of the imperfections were polished manually, as the ground surface is often very well registered by the LiDAR scanner, with most problems being concentrated on the thinly shaped objects around the environment, such as trees and poles. Additionally, the marching cubes algorithm proved to be very efficient in terms of reconstruction, being the fastest of the three, even with the highest face and vertex count.

6. Integration with Unreal Engine

The model created with Agisoft Metashape was chosen to test its integration with Unreal Engine, as it had the most well-preserved ground surface of the three, making it more suitable for a vehicle model to be tested within the environment. The mesh model is exported from Metashape as .fbx, a format developed by Autodesk [29]. This format was chosen due to the native support of its exportation and importation on both ends in terms of bringing the mesh model into the Unreal Editor, and the fact that the file format handles complex meshes supposedly well. To validate that the model could be imported to Unreal Editor without any issues, a blank simulation project was made to import the model into the game engine. The .fbx file was roughly 70,000 KBs and could be quickly imported into Unreal Editor without issues. Unfortunately, the vertex colors alone were insufficient to create a photo-realistic appearance regarding the mesh's texture, as seen in Figure 8. Hence, they were left out for now and would be considered in our future work.

	Reconstruction Algorithm	**Processing Time**	**Face count**	**Vertex count**	**Size**
MeshLab	Ball pivoting algorithm	49 minutes	1,435,521	804,420	143 MB
Cloud Compare	Poisson surface reconstruction	12 minutes	25,497,282	12,722,431	428 MB
Agisoft Metashape	Marching cubes	9 minutes	43,307,829	21,630,047	929 MB

Table 1: Reconstruction performance of each software.

The dimensions of the generated model were verified by comparing real-world dimensions with the 3D model from several locations around the scanned area. As the LiDAR scanner's laser pulses essentially measure distances by calculating the time-of-flight on each point, the dimensions were expected to be near a ratio of 1:1 between the real world and the mesh model. To verify this, some stone

Figure 8: The mesh inspected on Unreal Editor with vertex colors applied

Figure 9: Driving a virtual car in the Unreal environment.

Measured point	Real world (cm)	Virtual model (cm)	Difference (%)
Stone ledge 1 (vertical)	25	25	0
Stone ledge 2 (vertical)	28	28.5	~1.7
Stone ledge 3 (vertical)	70	70	0
Stone ledge 3 (horizontal)	28	27.5	~1.8
Stone ledge 4 (vertical)	45	45	0
Stone ledge 5 (vertical)	138	138	0
Stone ledge 5 (horizontal)	21	21	0

Table 2: Comparison of the measured points between the real world and the virtual environment.

ledges around the scanned environment were chosen as measuring points, as they were easy to measure horizontally and vertically both in the real world and within the virtual environment. Measurements from the real environment were taken with an ordinary measuring tape and were then compared with the measured results of a virtual ruler tool.

The results indicated that the 3D model matched the dimensions of the real world with great accuracy. The measurements were nearly identical between the real and digital environments, with minor, insignificant differences that were primarily caused by the fact that the measured points were not completely flat due to the reconstruction algorithm being unable to reconstruct geometrically perfectly flat surfaces from the given laser scan, meaning that measuring from a slightly different spot could make a slight difference in results. This validated that no visually detectable distortion regarding the dimensions that could have resulted from the scan was apparent and that the manufacturer's description of the accuracy held true.

7. Discussion and Future Work

In this paper, we have defined a preliminary workflow for using LiDAR scans to create virtual models of a real-world environment. Expectations regarding turning a LiDAR point cloud into a mesh model were delivered from previous works; however, none of them had the same setup regarding the device, target environment, and software used in the process. Existing research has not adequately assessed best practices regarding point cloud reconstruction workflows. That said, there were no clear expectations of the upcoming results regarding the artifact. Finally, mesh models were successfully made out of the point cloud, even if their quality was not entirely photorealistic. Even though the results were not perfect and the pipeline is not fully polished, our results contribute towards increased knowledge of the use of LiDAR on simulation environments and provide a baseline for future research.

We attempted to drive a vehicle model on the created artifact, as seen in Figure 9. The vehicle is perfectly drivable around the created environment without any interruption from unwanted points of noise. The car moves plausibly around the ground surface and responds to collision accordingly,

meaning that the ground surface is generated without excess noise that would cause issues with vehicle movement. It is important to note that these tests were only intended to ensure that the mesh surface could support interaction with a digital vehicle entity. More comprehensive testing should be conducted using dedicated automotive simulation tools, such as CARLA [30], to evaluate the suitability of the created environments for vehicular digital twin testing. This will require a more polished workflow and further research into integrating realistic physics and dynamic interactions, which we hope to assess in the future.

The main limitations of this work are related to the tools: LiDAR devices are generally costly, and while the ZEB Horizon LiDAR is supposedly a professional device, it would have been beneficial for the research to have access to different LiDAR devices to see whether the results could have been improved. Access to a LiDAR device that is not limited to being compatible with the manufacturer's software would open up many possibilities for customizing the data processing to be a smoother process. However, the process would be also sped up significantly with the current setup if the colors were to be left out.

Another limitation comes with evaluating the quality of the reconstructed mesh models. In general, there is a lack of assessment in systematic evaluation when it comes to LiDAR-created environments, and it is challenging to determine "how good is good enough. [21]" In our study, the original point cloud serves as the closest available reference, but it itself is subject to some noise and missing data, which naturally allows for the mesh model to be only as good as the input point cloud. While error metrics such as RMSE (root mean square error) between the point cloud and the reconstructed mesh model could offer a quantitative value regarding how the mesh surface fits as opposed to the input point cloud, visually evaluating the point cloud is also crucial, as RMSE alone can fail to capture structural inconsistencies and topological gaps. Depending on the reconstruction algorithm, some areas are better reconstructed than others, and this would have to be taken into account when considering adding more quantitative evaluation methods into the pipeline, which have to be planned carefully.

Future work: As this research provided a baseline for the results obtainable using high-level software tools available for point cloud processing, in the future, we plan to focus on polishing the mesh generation by experimenting with a more fundamental, low-level approach to point cloud-based mesh reconstruction. Instead of relying on pre-configured algorithms and more or less black-box implementations, we seek to improve the reconstruction pipeline using open-source libraries such as Open3D [31]. Additionally, we should explore the impact of point cloud registration on meshing quality, as proper alignment of scans is crucial for accurate reconstruction. This would allow for as many transparent and customizable processes as possible throughout the pipeline. As mentioned, we also plan to develop an evaluation method to assess the results more thoroughly and quantitatively.

8. Conclusion

In this paper, we presented a preliminary pipeline for turning MLS point clouds into simulation environments to test vehicular digital twins and compared three different methods for the virtual reconstruction of the point cloud environment. 3D models of the scanned point clouds were successfully created from the point cloud data using Cloud Compare, MeshLab, and Agisoft Metashape, each with their respective considerations and arguments. These tools created somewhat different models with their respective reconstruction algorithms: Cloud Compare's Poisson surface reconstruction produced watertight and detailed results. However, the Poisson surface reconstruction requires well-oriented normals, which, to get just right, may drastically increase computing time. The algorithm is also prone to noise when dealing with sparse input and would be potentially better suited for reconstructing closed shapes. Meshlab's ball pivoting algorithm produced a good quality mesh at the cost of a long processing time. The ball pivoting algorithm seems to require fine-tuning the ball radius for optimal results, which may require a lot of iteration. The ball pivoting algorithm could be more suitable for smaller point sets due to the long processing times. Metashape's marching cubes algorithm, on the other hand, proved to be very efficient in handling highly dense point sets. The marching cubes algorithm, like the Poisson

surface reconstruction, however, also seemed to be quite prone to noise with sparse inputs.

The point clouds created using the LiDAR device were of high quality, and the device's range was enough to create a mesh model of a moderately large area, such as a big parking lot. However, the density and complexity of the point cloud caused some problems in computing the surface normals for the mesh models. As specific points often have missing data, there is bound to be some distortion in the reconstructed models, which is more of a problem to be solved by the reconstruction algorithm. Despite the challenge of computing the mesh model, the LiDAR device could create a very high-quality point cloud, sufficient for creating a virtual model of a real-world environment. Regarding the suitability of the created models for simulation purposes, the algorithms could reconstruct a LiDAR point cloud to function for this purpose in Unreal Engine. As is, the created environments can be used for simple simulation purposes, but would require polishing for a more realistic appearance.

Acknowledgments

The work has been supported by the EU HORIZON projects CHIPS-JU CIA FEDERATE (grant 101139749) and CHIPS-JU RIA HAL4SDV (grant 101139789), Business Finland HAL4SDV national funding (grant 7655/31/2023), and the Finnish Research Council project Northern Utility Vehicle Laboratory Consortium GO!-RI (grant 352726). This work has been supported by FAST, the Finnish Software Engineering Doctoral Research Network, funded by the Ministry of Education and Culture, Finland.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] S. Manivasagam, S. Wang, K. Wong, W. Zeng, M. Sazanovich, S. Tan, B. Yang, W.-C. Ma, R. Urtasun, Lidarsim: Realistic lidar simulation by leveraging the real world, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11167–11176.

[2] FEDERATE, Federate, https://federate-sdv.eu/, 2023. Retrieved from FEDERATE SDV.

[3] Honda Motor Co., Honda reveals new prototype autonomous work vehicle at conexpo 2023, 2023. Retrieved from https://global.honda/en/newsroom/news/2023/c230307eng.html.

[4] Mercedes-Benz USA, Drive pilot, n.d. URL: https://www.mbusa.com/en/owners/manuals/drive-pilot#faq.

[5] L. Cheng, S. Chen, L. Xiaoqiang, H. Xu, Y. Wu, M. Li, Y. Chen, Registration of laser scanning point clouds: A review, Sensors 18 (2018) 1641. doi:https://doi.org/10.3390/s18051641.

[6] M. Jaboyedoff, T. Oppikofer, A. Abellán, M.-H. Derron, A. Loye, R. Metzger, A. Pedrazzini, Use of lidar in landslide investigations: a review, Natural Hazards (2012) 5–28. doi:10.1007/s11069-010-9634-2.

[7] J. Deems, T. Painter, D. Finnegan, Lidar measurement of snow depth: a review, Journal of Glaciology (2013) 467–479. doi:https://doi.org/10.3189/2013JoG12J154.

[8] F. Xue, W. Lu, Z. Chen, C. Webster, From lidar point cloud towards digital twin city: Clustering city objects, ISPRS Journal of Photogrammetry and Remote Sensing 167 (2020) 418–431. doi:10.1016/j.isprsjprs.2020.07.020.

[9] A. F. Chase, D. Z. Chase, J. F. Weishampel, Lasers in the jungle, Archeology 63 (2010) 27–29.

[10] FARO, What is lidar and how does it work?, https://www.faro.com/en/Resource-Library/Article/What-is-Lidar, n.d. Retrieved from FARO.

[11] T. Raj, F. Hashim, A. B. Huddin, I. Mohd, A. Hussain, A survey on lidar scanning mechanisms, Electronics 9 (2020) 741. doi:10.3390/electronics9050741.

[12] C. Cao, M. Preda, T. Zaharia, 3d point cloud compression: A survey, in: Proceedings of the 24th International Conference on 3D Web Technology, 2019, pp. 1–9. doi:https://doi.org/10.1145/3329714.3338130.

[13] F. Remondino, From point cloud to surface: the modeling and visualization problem, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV-5/W10 (2003). doi:10.3929/ethz-a-004655782.

[14] M. Dassot, T. Constant, M. Fournier, The use of terrestrial lidar technology in forest science: application fields, benefits and challenges, Annals of Forest Science 68 (2011) 959–974. doi:10.1007/s13595-011-0102-2.

[15] L. Li, R. Wang, X. Zhang, A tutorial review on point cloud registrations: Principle, classification, comparison, and technology challenges, Mathematical Problems in Engineering 2021 (2021). doi:10.1155/2021/9953910.

[16] Y. Liao, S. Donné, A. Geiger, Deep marching cubes: Learning explicit surface representations, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2916–2925. doi:10.1109/CVPR.2018.00308.

[17] H. E. Cline, W. E. Lorensen, Marching cubes: A high resolution 3d surface construction algorithm, in: ACM SIGGRAPH Computer Graphics, volume 21, 1987, pp. 163–169. doi:https://doi.org/10.1145/37402.37422.

[18] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics 5 (1999) 349–359. doi:10.1109/2945.817351.

[19] M. Bolitho, M. Kazhdan, R. Burns, H. Hoppe, Parallel poisson surface reconstruction, in: Advances in Visual Computing, volume 5875, 2009, pp. 678–689. doi:https://doi.org/10.1007/978-3-642-10331-5_63.

[20] M. Boussaha, B. Vallet, P. Rives, Large scale textured mesh reconstruction from mobile mapping images and lidar scans, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4 (2018) 49–56.

[21] C. Goodin, M. N. Moore, D. W. Carruth, Z. Aspin, J. Kaniarz, Geometric fidelity requirements for meshes in automotive lidar simulation, in: Virtual Worlds, volume 3, MDPI, 2024, pp. 270–282.

[22] K. A. Piirainen, R. A. Gonzalez, Seeking constructive synergy: Design science and the constructive research approach, in: Design Science at the Intersection of Physical and Virtual Design, 2013, pp. 59–72. doi:10.1007/978-3-642-38827-9_5.

[23] GeoSLAM Ltd., Zeb-horizon™ user's manual, 2020. Retrieved from https://geoslam.com/wp-content/uploads/2021/02/ZEB-Horizon-User-Manual-v1.3.pdf.

[24] M. Bolitho, M. Kazhdan, R. Burns, H. Hoppe, Parallel poisson surface reconstruction, in: Advances in Visual Computing: 5th International Symposium, ISVC 2009, Las Vegas, NV, USA, November 30-December 2, 2009, Proceedings, Part I 5, Springer, 2009, pp. 678–689.

[25] D. Girardeau-Montaut, Cloudcompare (version 2.x), https://www.cloudcompare.org/, 2011. Accessed: 2025-01-24.

[26] Cloud Compare, Cloudcomparewiki, https://www.cloudcompare.org/doc/wiki/index.php?title=NormalsCompute, 2016. Retrieved from Normals/Compute.

[27] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, MeshLab: an Open-Source Mesh Processing Tool, in: V. Scarano, R. D. Chiara, U. Erra (Eds.), Eurographics Italian Chapter Conference, The Eurographics Association, 2008. doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[28] A. LLC, Agisoft metashape (version x.x), https://www.agisoft.com/, 2025. Accessed: 2025-01-24.

[29] Autodesk, Adaptable file format for 3d animation software, https://www.autodesk.com/products/fbx/overview, 2024. Retrieved from Autodesk.com.

[30] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator, in: Annual Conference on Robot Learning, 2017, pp. 1–16.

[31] Q.-Y. Zhou, J. Park, V. Koltun, Open3D: A modern library for 3D data processing, arXiv:1801.09847 (2018).