Animation of Laser Scanning Process for Deep Learning-based Reactor Parts Classification

Hyeji Na*, Sungmoon Joo and Jonghwan Lee
Korea Atomic Energy Research Institute, Daedeok-daero 989-111, Yuseong-gu, Daejeon, Korea
**Corresponding author: nahyeji@kaeri.re.kr*

1. Introduction

This paper presents an efficient method to animate laser scanning process using Blensor, an open source simulation package for scanning sensors [1]. The simulation of laser scanning process generates synthetic point cloud data which is used for the training of a deep learning model for reactor parts classification [2]. The animation shows the change in the position of a scanner and the generation of the point cloud as a result of the scan.

This paper is organized into 4 sections. Section 2.1 briefly introduces the point cloud scan process animation. Section 2.2 explains the interpolation method for changing the scanner position. Section 2.3 shows a comparison of the two methods of converting point clouds into meshes. Section 3 concludes the paper.

2. Methods and Results

In this section, the interpolation method for changing the scanner position is described, and two methods of converting point clouds into meshes are compared.

2.1 Animation Overview

The point cloud scanning process animation is a means to easily convey information about the location of the scanner and resulting point cloud overlayed on the target object, so that viewers understand the scanning process and how synthetic point clouds are obtained. As shown in Figure 1, a scanner looks at the object and scans the object while moving on the surface of virtual sphere. An image, such as shown in Figure 1, is saved each time the scanner is moved or a scan is generated. The animation created by splicing all the saved images together.

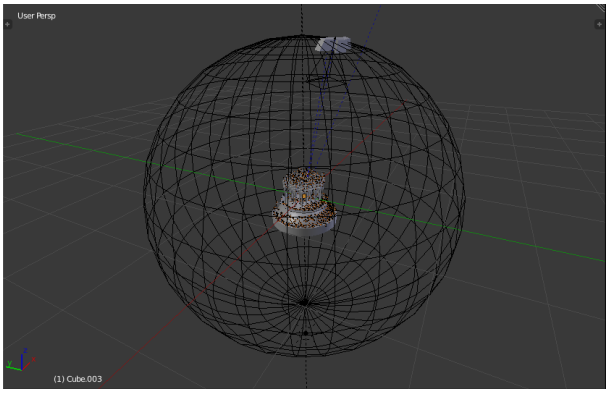

Figure 1. A scene during the scanning process

2.2 Interpolation

The animation function of Blensor uses interpolation to express the movement of an object smoothly. However, it cannot contain the creation of new objects over time. Therefore, to create an animation that shows both the position change of the scanner and the generation of synthetic point cloud, the method of creating animation from still images should be used. In case of using still images, code that performs interpolation function should be written in blensor script. This can be easily solved using the scipy module [3]. The splprep function of the scipy module was used to find the B-spline representation of the scanner's position. After finding the B-spline expression, the splev function of the scipy module was used to evaluate the B-spline and its derivatives. As a result, the movement of the scanner as shown in Figure 2 is obtained. The yellow wire-frame sphere is the sphere over which the scanner moves, and the blue line is the path of the scanner. The red points are where the scanner scans a target object.

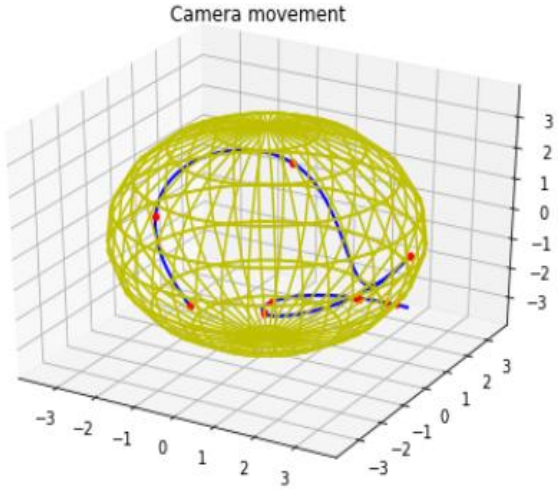

Figure 2. Movement of scanner

2.3 Methods of converting point clouds into meshes

This section describes two methods of rendering point clouds using meshes and compares the two methods in terms of computation time.

The first way to turn a point cloud into a mesh is to create a small cube and copy that cube repeatedly at each point. If it is the first point, a cube is created and the location, color, and size of the cube are specified. If

it is not the first point, copy the first cube to a new location. And if it's not the last point, it repeats the process, and if it's the last point, it gets the job done. The process is summarized in Fig.3.

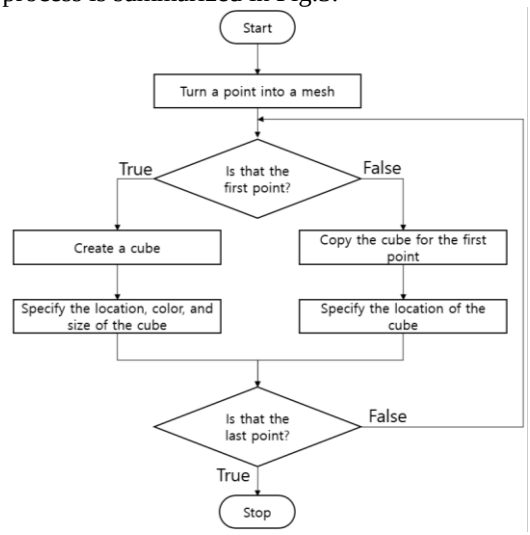

Figure 3. Flow chart (the first method)

The second way is to use the bmesh module [4]. If it is the first point, a cube is created and characteristics of the cube are specified. Then, create an empty bmesh and a mesh that will be the basis for the bmesh operation. If it is not the first point, a new vertex is created at each point. After the vertice of the last point have been created, convert the bmesh to a mesh and duplicate the child objects at all vertices of the base mesh. Finally, if the base mesh is set as the parent of the cube that was first created, all points in the point cloud have a cube corresponding to it. The process is summarized in Fig.4.

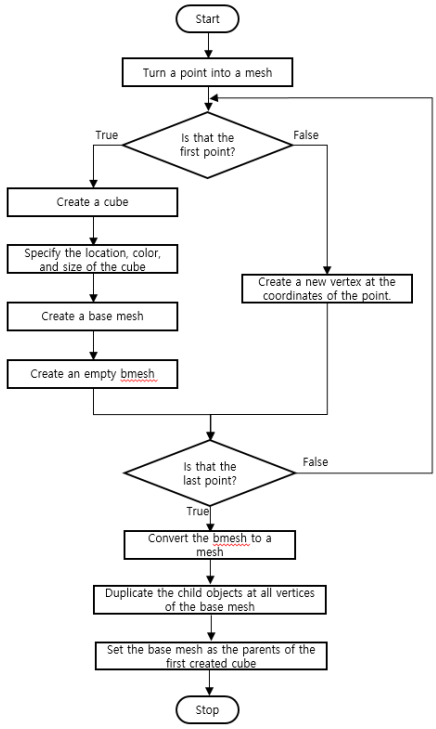

Figure 4. Flow chart (the second method)

Both methods render the same point cloud image but the computational efficiencies differ. Therefore, it is necessary to compare the two methods to determine which method to choose. For the comparison of the two methods, the time taken to convert point clouds into meshes for each method was measured. Based on the measurement results, the time taken to convert a point into a mesh was calculated. Fig.5 is a graph showing the calculation results. Numbers on the x-axis of the graph is numbers entered by the user. It takes a lot of time to perform the operation without sampling the point cloud. So, it was sampled for the quotient of the number of points in the point cloud file divided by the number entered by the user. Fig. 5 is the result of each measurement after inputting the user input number as [50, 40, … 10]. As a result, using 'bmesh' reduces the time by approximately 53% compared to using the 'cube copy' method.

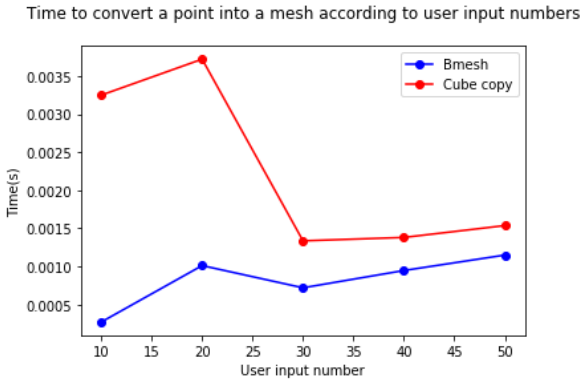

Figure 5. Time to convert a point into a mesh according to user input numbers

3. Conclusions

This study introduces the interpolation method and the point could to mesh conversion method, which are techniques required to animate the point cloud scanning process in Blensor, and compares the two methods to convert the point cloud to mesh. The result of the comparison is that the bmesh method saves 53% time compared to the cube-copy method.

REFERENCES

[1] https://www.blensor.org
[2] Sungmoon Joo et al, "Application of a Deep Learning Network to 3D Modeling of Nuclear Facilities," Waste Management Symposia, March 2019.
[3] https://docs.scipy.org/doc/scipy/reference/interpolate.html, last access on Aug. 14 2020.
[4] https://docs.blender.org/api/2.79/bmesh.html, last access on Aug. 14 2020.