**3. Methodology**

**3.1 Scope**

Point clouds were used in this study due to their popularity in representing spatial scenes, and octree and K-D tree were chosen as the primary structures for visualization due to their popularity in storing point cloud.

**3.2 Methods**

**3.3 Data Segmentation and Input Selection**

Three objects – a house, a tree, and a light pole – were segmented from a Brooklyn Army Terminal point cloud dataset and had been visualized in forms of octree and k-d tree to exemplify how different kind of files were stored in these two data structures. The point distributions among the three point cloud objects were distinctive from each other. The light pole point cloud was featured by a uniformly distributed set of points towards one vertical direction. The house point cloud had shown an uneven point distribution, with more points on the outside forming a blanket to encapsulate the empty inside space. The tree point cloud also had displayed an uneven point distribution, with more points representing leaves on the top and fewer on the bottom.



Figure 4. Original House Point Cloud

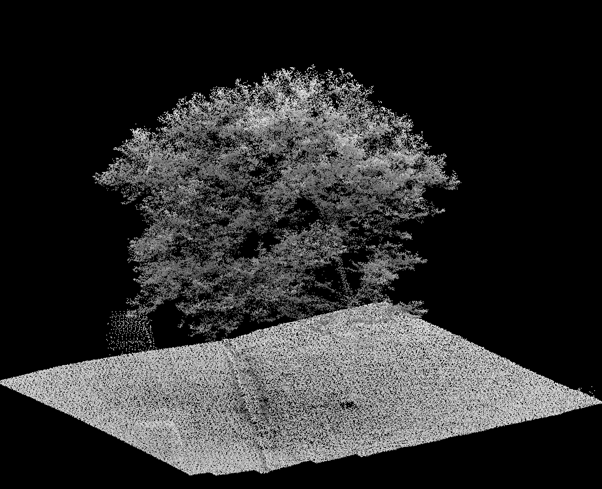


Figure 5. Original Tree Point Cloud

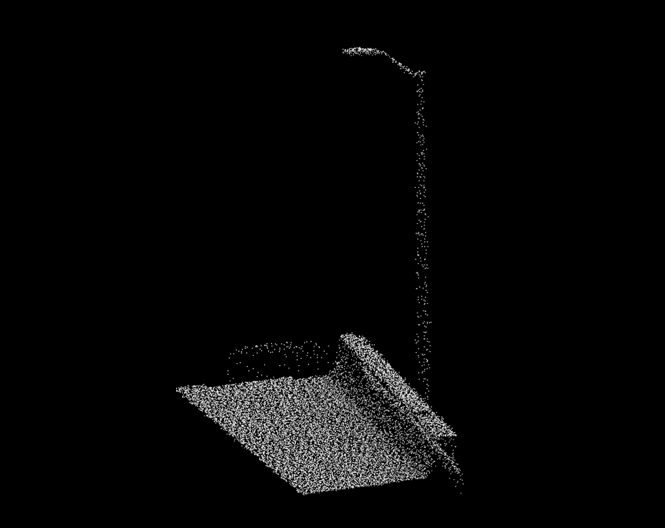


Figure 6. Original Light Pole Point Cloud

**3.4 Data Cleaning**

To reduce noise data points and computational load in generating visualization, CloudCompare was used to segment all point clouds and subsample only the house and tree point cloud. Two reduction methods, random subsampling and uniform subsampling by distance methods, were utilized for segmenting the house and the tree data points. The random subsampling method would randomly eliminate 99% of the data points of the files, and the uniform subsampling method would uniformly remove any points within 2 units. The file size of the segmented light pole was already optimal enough to produce clear visualizations, so it was not subsampled. Overall, there were 5 input files used in visualization – the randomly reduced house and tree files, the uniformly reduced house and tree files, and the light pole file.

A black background with white dots

Description automatically generated

Figure 7. House Point Cloud After Segmented and Randomly Reduced by 99%

A white dots on a black background

Description automatically generated

Figure 8. House Point Cloud After Segmented and Uniformly Reduced by Distance of 2 Units



Figure 9. Tree Point Cloud After Segmented and Randomly Reduced by 99%

A white dots in the sky

Description automatically generated

Figure 10. Tree Point Cloud After Segmented and Uniformly Reduced by Distance of 2 Units

A white object with a black background

Description automatically generated

Figure 11. Tree Point Cloud After Segmentation

**3.5** **Octree and K-D Tree Structures Visualization**

Python served as a tool for creating visualizations of octree and k-d tree structures, and NetworkX was used for projecting point cloud data into 2D plane. Time and psutil libraries were used for computing runtime and memory consumption during the program's operation with the 5 inputs files. Dots were used to represent nodes stored in the two data structures, and Gephi was employed to enhance visualization by minimizing the node-crossing situations and applying a blue-to-yellow-to-red color map, illustrating the hierarchical characteristic of the two tree structures. Blue colored node would signify the root of the tree while the more yellowish or reddish colored nodes were its children. The deeper the level a tree has, the more reddish its leaf nodes were. Runtime (in seconds) and memory usage (in Megabytes) were calculated alongside the visualization.

A diagram of a cloud

Description automatically generated

Figure 12. Flowchart of This Study

**4. Results**

**4.1 Comparative Analysis of Data Structure Visualizations**

**4.1.1 House**

Both of the octree visualizations of house point cloud had a group of reddish nodes or "tip" in the top left representing the flashing at the house roof. However, in the k-d tree visualizations, the distribution of the nodes was more uniform, and there was no one node that stood out among the whole visualizations. It was worth noting that due to the uniformity of the k-d tree structure, more nodes had been utilized to form a subtree. Furthermore, since color of the nodes changed from blue to yellow to red along with the tree's depth, with k-d tree visualizations having redder colored nodes, it indicated that the k-d tree used more depth in storing the house point cloud. This made sense since only two subdivisions were allowed among each node of the k-d tree, compared to the allowance of eight subdivisions in an octree, meaning that the k-d tree needed more depth to store the same amount of point cloud data.

A colorful network of dots and lines

Description automatically generated with medium confidence

Figure 13. Octree Visualization of Randomly Reduced House Point Cloud

A colorful network of dots and lines

Description automatically generated with medium confidence

Figure 14. Octree Visualization of Uniformly Reduced House Point Cloud

A colorful network of dots and lines

Description automatically generated

Figure 15. K-D Tree Visualization of Randomly Reduced House Point Cloud

A network diagram of dots and lines

Description automatically generated

Figure 16. K-D Tree Visualization of Uniformly Reduced House Point Cloud

**4.1.2 Tree**

A prominent yellowish tip on the top right was observed among three of the tree's octree visualizations, likely representing the top of the tree where leaves, absorbing more sunlight, grow more extensively (Figure 17, 18, 19). Notably, only octree visualizations of the house point cloud had the tip while both the octree and k-d tree visualizations of the tree point cloud had the tip. However, the feature was absent in the k-d tree visualization of the uniformly reduced file, perhaps due to the manner in which that reduction method removed any leaves that were closely packed in a set distance (Figure 20). Again, the property that k-d tree visualizations having more reddish colored leaf nodes due to the tree having deeper depth and the choice of reduction method influenced the uniformity of the node distribution was present here.

A network diagram of dots and circles

Description automatically generated

Figure 17. Octree Visualization of Randomly Reduced Tree Point Cloud

A network of dots and lines

Description automatically generated

Figure 18. Octree Visualization of Uniformly Reduced Tree Point Cloud

A network of dots and lines

Description automatically generated

Figure 19. K-D Tree Visualization of Randomly Reduced Tree Point Cloud

A network of dots and lines

Description automatically generated

Figure 20. K-D Tree Visualization of Uniformly Reduced Tree Point Cloud

**4.1.3 Light Pole**

The light pole octree visualization also featured a tip, which represented the light as one could observe from the point cloud image, though only octree visualization had it (Figure 11, Figure 21). The colors of both outer layers of the octree and k-d tree visualizations of light pole were more reddish than those of the tree's and the house's, possibly due to the structure of the light pole being more concentrated in a straight vertical direction, so more depths were utilized to store this kind of structure (Figure 21, Figure 22).

A colorful network of dots

Description automatically generated with medium confidence

Figure 21. Octree Visualization of Segmented Light Pole Point Cloud

A network of dots and lines

Description automatically generated

Figure 22. K-D Tree Visualization of Segmented Light Pole Point Cloud

**4.2 Comparative Analysis of Program Runtime and Memory Usage**

Note that the original point cloud files were utilized in calculating the program's runtime and memory usage. Given that the house point cloud contained the greatest number of points, its octree and k-d tree visualizations required more time and memory to process. Additionally, since octrees accommodate more child nodes, their visualizations took longer to generate. Interestingly, there was a direct correlation between the number of points in a point cloud and the runtime and memory usage for processing its octree and k-d tree visualizations. For instance, creating an octree visualization of the house point cloud was approximately 1.544 times faster and used 1.391 times more space than that of the tree point cloud. This closely aligns with the fact that the house point cloud had about 1.375 times more points than the tree point cloud. Despite octrees inherently requiring more space due to their capacity for more child nodes, the memory usage for generating both octree and k-d tree visualizations remained comparable.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of Points | Runtime: Octree (s) | Runtime: K-D Tree (s) |
| House | 374,328 | 7.422 | 1.764 |
| Tree | 272,332 | 4.806 | 1.089 |
| Light Pole | 18,895 | 0.343 | 0.078 |

Figure 23. Table of Program Runtimes for Generating Data Structures Visualizations

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of Points | Memory Usage:  Octree (MB) | Memory Usage:  K-D Tree (MB) |
| House | 374,328 | 240.984 | 221.484 |
| Tree | 272,332 | 173.234 | 143.547 |
| Light Pole | 18,895 | 13.922 | 9.844 |

Figure 24. Table of Memory Usages for Generating Data Structures Visualizations

**5. Discussion**

**5.1 Libraries and Software Choice**

Python was chosen to extract libraries and do visualization. NumPy was used to import point cloud files as matrices for the convenience in converting them into a tree structures. NetworkX was a package for creating network visualizations and tree data structure can be visualized as a network of nodes. Initially, Matplotlib was employed for displaying visualizations, but due to the complexity of the point clouds in this study, which led to node overlapping (Figure 25), it became challenging to address this issue through coding. Therefore, Gephi was adopted to refine and enhance the visualizations. On the other hand, the time and psutil libraries were imported to measure program performance, namely its runtime and memory usage in generating visualizations.

A blue circle with a white background

Description automatically generated

Figure 25. Light Pole Point Cloud's Octree Visualization using Matplotlib

**5.2 Motivation for Data or Noise Reduction**

The purpose for data segmentation and dimension reduction was not only to alleviate computational burden in generating visualizations, but also to remove unnecessary data such as the ground area in the original tree point cloud (Figure 5). That way, the resulting visualization would be more accurate in reflecting the way specific tree structures stored various types of point cloud objects.

**5.3 Limitation**

Both reduction methods used in this study may have the potential to deteriorate the original structure of the point cloud object, as shown in the loss of tip in some of the visualizations (Figure 15, Figure 16, Figure 20, Figure 22). Note that among the visualizations without the tips, all of them were k-d tree visualizations and most of them were reduced uniformly.

The scope of this study was limited as only two data structures were inspected, and the variety of objects being visualized was small – all of them have a bulk of area on top. Other data structures such as R-tree might be a good choice to be incorporated in the future study. Additionally, the speed of data extraction and the memory requirements for storing point cloud into these data structures should also be considered in the future study.

**6. Conclusion**

This study involved visualizing two commonly used data structures for point cloud analysis using three distinct point clouds, two of which were further reduced via methods from CloudCompare. The study also compared the performance of generating these visualizations. The K-D tree emerged as a preferable choice for those preferred uniform distribution of point cloud data in memory or a faster method for storing point cloud files. However, the octree proved more effective in maintaining the distinctive features of the point cloud objects.