Behind the Scenes of Point Cloud Storage

Anni Zheng

**Abstract**: Effective organization of big data are crucial in urban science. The data processing and storage system must preserve the original data structure while minimizing the computational load. Data structures help facilitate this process. In this study, two commonly used data structures, the octree and the k-dimensional tree, would be analyzed and visualized as networks for how they handle point distributions within point cloud data. Additionally, there would be a side study that explores how random and spatial subsampling methods potentially change the structure of point clouds and affect their storage. While the study does not provide direct guidance, it introduces concepts and uses visualizations and statistical techniques to help select appropriate data structures for point cloud storage.

**1.** **Introduction**

Point cloud is a collection of data points defined in a three-dimensional (3D) coordinate system, where each point encodes positional information of the surface of an object captured using Light Detection and Ranging (LiDAR) technologies. Point clouds are often used in the urban studies field for 3D modeling and mapping, providing detailed information about the shape and structure of a particular object or an environment. As the scanned area increases, the point cloud becomes denser with more data points. Consider the recent 3D Elevation Program managed by the U.S. Geological Survey, which provides high-quality data representing the nation's features, would have an excessive number of points in their point cloud file, it is, therefore, crucial to consider using data structures in managing large point cloud files while using subsampling methods in reducing their size and noise. This study would introduce two tree data structures, the octree and the k-dimensional tree (k-d tree), along with random and spatial subsampling methods due to their popularity among urban science community.

Tree data structures are organized and hierarchical ways of storing data. To better demonstrate how point cloud data objects are stored, the study would first build octree and k-d tree data structures storing three kinds of point cloud objects then visualized those data structures as 2D networks, with nodes representing portions within the data structures and lines connecting those nodes as linkage between the portions. Together they marked the hierarchical connections of how each part of the point clouds were organized by the data structures in the database. In addition to the visualizations, there would be an examination made based on how the data structures decomposed by a set of density thresholds. The decomposition process would be encoded as distributions for giving insights such as whether two point clouds of the same kind, but subsampled differently, would be stored in the same way by a data structure.

The study also hopes to provide inspirations on how one could understand the octree and k-d tree based on the visualizations and distributions it provided, so people may have a more comprehensive awareness in choosing appropriate data structures and subsampling methods for handling point cloud data.

**2.** **Background**

Data structures plays a crucial role in organizing and storing data for efficient processing and retrieval. This includes tree structures, where there is a node acting as the root of the tree, and the data handled by this data structure would be broken down into several portions or child nodes within them in simulating the growth of a tree and with the ending nodes as the leaves of the tree. One may also think of a tree data structure as a nested lists with the root node being the outer list and its child nodes being the lists inside it, nesting each other. Basically, the root node holds all the data information while the child node shares part of them. The depth of the tree data structure is determined by how many nested lists are produced.

In this study, tree data structures such as octrees and k-d trees were examined for their effectiveness in managing point cloud. To recall, points in point cloud are derived from the LiDAR technology and contain spatial information (i.e., location information such as x, y, z for each point) of the surface of an object. That is to say, a collection of points in point cloud forms the basic shape of an object. Three kinds of point cloud data were utilized in study for inspecting how the two data structures store point cloud—they are house, tree, and light pole, each drawn from a large Sunset Park LiDAR dataset published in 2019. All three objects, commonly found in urban settings, embody distinct shapes allowing more detailed analysis of how data structures handle them.

On the other hand, the study drew inspiration from a concept called persistent homology, a tool used by mathematicians to examine how the original shapes formed the data points formed and destroyed with a set of point radii—as points' radii expanded, they would start to touch each other, and a chain of points touching each other would form either a ring, a tunnel, or other interesting shapes signaling the destruction of their original shapes. By examining the frequency of their shape changes, insights could be drawn about the persistence of certain points in maintaining a shape, highlighting the significance of shapes hold by certain points—shapes that are quickly destroyed are potentially not significant. The study took a different approach in examining the persistence of the depth of a tree data structure with a set of density threshold. A complete tree data structure would create child node and expand to a certain point. The further it could expand; the less data points would be captured by its child node but the more details they could capture. The concept of density threshold would be—as the number of points captured by a child node do not past the threshold, that child node would not form. In inspecting the persistence of a node with a set of density thresholds, the study hoped to draw insights on how the two data structures held point cloud differently (i.e., which portion of the data were more persistently, and whether the shape of the data or the subsampling method affected the persistence of them).

**3. Methodology**

**3.1 Scope**

This study was driven by two research objectives. The first was to visualize how various data structures stored urban object data, through which the potential strengths and weaknesses of each data structure could be analyzed. The second objective was to examine the effects of two subsampling methods—random and spatial subsampling—on the distribution of points within these data structures. The scope of this work included providing two visual mechanisms for comparison, as well as a traditional quantitative basis, specifically focusing on the investigation of three data structures.

A diagram of a data analysis process

Description automatically generated

Figure 1. The Study's Workflow

**3.2 Data and Data Structure Selection**

Using LiDAR data and its positional attributes, this study focused on octree and k-d tree data structures. These structures exhibit some differences in making the partition for organizing point cloud data. Octree would iteratively divide the space into eight portions and would keep only the portion with points in it while k-d tree would iteratively divide the data along a certain axis, which would break it down to at most two portions. To better evaluate the characteristics of these data structures in handling real-world objects, three different urban point clouds were used in giving insights to their robustness in processing and managing them. The three specific objects were a house, a tree, and a light pole, with their distinct structural characteristics being: (1) the house point cloud, with its discernible geometric features, had a dense exterior layer that mimicked walls and roofs, enclosing an empty interior space; (2) the tree point cloud, characterized by its irregular structure, exhibited varying densities of points—denser at the top with foliage and thinner towards the bottom trunk area; (3) the light pole point cloud, defined by its directional geometry, predominantly displayed a vertical and uniform distribution of points, with denser concentrations at the top and bottom to represent the light fixture and base, respectively.

A house with a black background

Description automatically generated A tree on a black background

Description automatically generated A white object with a black background

Description automatically generated

Figure 2A. House Point Cloud Figure 2B. Tree Point Cloud Figure 2C. Light Pole Point Cloud

**3.3 Data Cleaning**

To reduce some noise data and allow each point cloud to better represent the objects' characteristics, segmentation was performed in CloudCompare before applying two subsampling methods. The specific segmentation for each type of point cloud was included. Specifically, the house point cloud had its fences, vegetation areas, and some of its roof areas removed as noise data. The tree point cloud had mostly its ground area and some of its foliage areas removed. The light pole point cloud had the car object behind it removed. The yellow areas indicated in figure 3 were the removed sections.

A house with a yellow border

Description automatically generated A tree growing out of a field

Description automatically generated A white object with a black background

Description automatically generated

Figure 3A. House Point Cloud Figure 3B. Tree Point Cloud Figure 3C. Light Pole Point Cloud

In this study, two subsampling methods were employed—spatial subsampling and random subsampling, facilitated by CloudCompare. To ensure a consistent basis for comparison, each urban point cloud object was reduced to precisely 4096 points. Maintaining an equal number of points across different objects enabled consistent comparative analysis. The spatial subsampling method involved reducing the number of points within a predetermined distance. It is important to note that the automated subsampling process provided by CloudCompare did not always result in the exact desired number of points; therefore, manual adjustments were needed to achieve 4096 points across all samples. Specifically, the spatially subsampled house point cloud was configured with the distance set to 1.41-unit distance, with 1 point being removed manually afterward. The spatially subsampled tree point cloud was created with the distance set to 1.05-unit distance, with 12 points being removed manually afterward. The spatially subsampled light pole point cloud was created with the distance set to 0.25-unit distance, with 11 points being removed manually. Following spatial subsampling, a random subsampling method was applied, which was simpler and quicker. This process involved specifying the desired number of remaining points, after which points were removed randomly.

A white dots in the air

Description automatically generated with medium confidence A house made out of small white dots

Description automatically generated

Figure 4A. Randomly Subsampled House Figure 4B. Spatially Subsampled House

A white tree with many dots

Description automatically generated with medium confidence A white dots on a black background

Description automatically generated

Figure 4C. Randomly Subsampled Tree Figure 4D. Spatially Subsampled Tree

A white object with a black background

Description automatically generated A white object with a black background

Description automatically generated

Figure 4E. Randomly Subsampled Light Pole Figure 4F. Spatially Subsampled Light Pole

**3.4 Octree and K-D Tree Creation**

After the segmentation and subsampling steps, six CSV files of point clouds with positional information (x, y, z of each point) were exported to Python for data structure creation. In this study, two data structures were created with equal depth, 3, for comparison and visualization. The octree was created by first determining the span of the points in each point cloud along the x, y, and z axes, then dividing the space into at most eight nodes, with each node potentially holding a certain number of points. Iterative division was then performed until either a set depth was reached or there were no points in that node, thus eliminating the need for further division in storing point data. The k-d tree was created by first sorting the points in sequential order along the x-axis, then making three divisions at the center point along the x-axis, followed by the y-axis, and then the z-axis. In this study's implementation, the octree was more space aware as it was created with more focus on the initial span of the object but less on the balance of point density inside those equally large eight nodes. In contrast, the k-d tree was more point-aware as it divided at the center point, making the balance of point distribution more prominent. During the creation of the two data structures, hierarchical information such as which node settles at which depth and how many points are contained inside, along with the connection information among nodes, was stored in two CSV files, node.csv and edge.csv. These were inputted into Gephi to produce network visualizations of the structures of the data structures.

**3.5 Data Structure Visualization**

The exported csv files would be inputted into Gephi for automate generation of network visualization. Each node having a label showing its density. Size and color map were employed for highlighting the point distribution and density among data nodes of the two data structures with darker and larger nodes containing more points and lighter and smaller nodes containing less points. Nodes were also dragged using tools from Gephi to prevent crossing, and so to make the visualization better.

A screenshot of a network

Description automatically generated

Figure 5. Part of Network Visualization made using Gephi

**3.6 Data Analysis with Density Threshold**

An analysis was then conducted using a density threshold to explore how the two tree data structures stopped producing more child nodes when a range of density thresholds, from 0 to 4096 (the total number of points), was established. An algorithm was integrated into the tree structure creation process to keep track of the number of child nodes that were not formed due to them not passing the density threshold. For instance, in a k-d tree shown in figure 6A below, setting the density threshold to 520 would halt the creation of all eight nodes at the deepest level. The distribution made by the density thresholds could be used to visualize the destruction of two data structures storing different point clouds; therefore highlighting their characteristics. To further quantify this comparison, Wasserstein distance, which measures the transferability of a data to another, was calculated, z-score normalized, and compared between data structures holding the same type of data but sampled differently. This comparison helped demonstrating the differences in data structure destruction patterns, explaining whether the effects made by the random and spatial subsampling methods significantly altered the shape of the data. Note that a data structure with fewer nodes simply indicated fewer specifications of data division. This aspect of the study was focused solely on the nature of the data structure.

**4. Results**

**4.1 Visualizations**

A diagram of a network

Description automatically generated A diagram of a network

Description automatically generated

Figure 6A. K-D Tree of Randomly Subsampled House Figure 6B. K-D Tree of Spatially Subsampled House

A diagram of a network

Description automatically generated A diagram of a network

Description automatically generated

Figure 6C. K-D Tree of Randomly Subsampled Tree Figure 6D. K-D Tree of Spatially Subsampled Tree

A diagram of numbers and circles

Description automatically generated A diagram of a network

Description automatically generated

Figure 6E. K-D Tree of Randomly Subsampled Light Pole Figure 6F. K-D Tree of Spatially Subsampled Light Pole

A network of connected lines and dots

Description automatically generated with medium confidenceA network of connected spheres

Description automatically generated with medium confidence

Figure 7A. Octree of Randomly Subsampled House. Figure 7B. Octree of Spatially Subsampled House

A network of blue dots and circles

Description automatically generatedA network of dots and circles

Description automatically generated

Figure 7C. Octree of Randomly Subsampled Tree Figure 7D. Octree of Spatially Subsampled Tree

A network diagram with circles and dots

Description automatically generated A diagram of a network

Description automatically generated

Figure 7E. Octree of Randomly Subsampled Light Pole. Figure 7F. Octree of Spatially Subsampled Light Pole

**4.2 Wasserstein Distance**

**House**

|  |  |
| --- | --- |
| Data | Wasserstein Distance (Rounded) |
| Octree | 1.395 |
| K-D Tree | 1.111 |

Figure 8A. Wasserstein Distance of the Distribution

made by Density Thresholds among House Data

**Tree**

|  |  |
| --- | --- |
| Data | Wasserstein Distance (Rounded) |
| Octree | 0.899 |
| K-D Tree | 0.202 |

Figure 8B. Wasserstein Distance of the Distribution

made by Density Thresholds among Tree Data

**Light** **Pole**

|  |  |
| --- | --- |
| Data | Wasserstein Distance (Rounded) |
| Octree | 0.496 |
| K-D Tree | 1.313 |

Figure 8C. Wasserstein Distance of the Distribution

made by Density Thresholds among Light Pole Data

**4.3 Distribution Graphs created by Density Thresholds**

**A graph of data on a white background

Description automatically generated**

Figure 9A. Distributions made in Octrees by Density Thresholds

**A group of blue lines

Description automatically generated**

Figure 9B. Distributions made in K-D Trees by Density Thresholds

**5. Discussion**

**5.1 Subsampling Methods in Preserving Data Shapes**

Eyeballing the way points were displayed in CloudCompare in figure 4, one could observe that the spatial reduction method preserved the structure of the house better than random reduction—both the roof and the walls of the spatially subsampled house were preserved, allowing one to still perceive the house's encapsulating structural characteristics to the internal empty space. In contrast, the randomly subsampled house had mostly its roof preserved and with only a few points unevenly distributed on the wall parts—here, the encapsulating structure was not maintained; instead, the covering structure of the roof was more prominent. On the other hand, both reduction methods effectively preserved the data for the tree and the light pole. One could easily discern the represented objects in the point clouds produced by both methods. Interestingly, the random reduction method might be deemed better at hinting the uneven leaf distribution of the tree data, rendering the tree data more realistic compared to that produced by the spatial reduction method. However, the random reduction method was generally more unstable than spatial reduction in this case, as it could discard points critical to the structure of objects, yet it was also faster, which could be advantageous for reducing the data size of highly dense point cloud files.

**5.2 Space Management using Data Structure**

**5.2.1 Noise Data Handling**

Regular data structures used in this study did not have algorithms implemented to handle noisy data. Consequently, one issue with directly feeding data into these structures was that the noise became incorporated throughout the division process. For example, as shown in figure 6E and 10, half of the k-d tree might have stored noise data unrelated to the light pole. Therefore, it was essential and advisable to clean the noisy portions of the data before introducing them into the data structure. Similarly, caution was recommended when using network visualization to represent unfamiliar data. Like the data structure, if noise was not managed correctly, the visualization might not have accurately represented the typical storage of point cloud objects.

A blue and yellow object with a long tail

Description automatically generated with medium confidence

Figure 10. Light Pole Data Stored After the First Split in K-D Tree

**5.2.2 Space Awareness vs. Point Awareness**

Inspecting the network visualization produced from the two structures, one could see how the octree visualization might have better represented the point distribution of the original data, as shown in figure 7C, in which the denser part with nodes holding denser node being the leaf part of a tree displayed in figure 4C, while the lower areas are the trunk part of the tree. This was because the octree implemented here was more space aware. It divided the space of the object's span into an equal number of chunks and was indifferent to how many points each chunk stored (unless there were no points, in which case the octree would not generate further child nodes). Thus, the nodes represented by the octree, if projected into network visualization, would have reflected the density of points in a portion of the space. The visualization resembled a snapshot taken from one side of the whole three-dimensional object.

However, the k-d tree was less space aware. It divided points based on the axis determined by the algorithm and the median of the points, focusing less on the logical orientation of the points, as one could not discern a tree from figure 6C. Moreover, as in figure 6E and figure 11, the light of the light pole was placed in the same portion as the base of the pole, which was illogical if viewing the pole as a vertically oriented object but made sense if viewed from the side of the pole since the split considered the front or protruding side of the pole as one portion while the back side as another. Visually speaking, the octree network visualization was more human-readable as its splitting method could seem arbitrary, and the point density shown in the visualization was so uniform for all objects that it became hard to distinguish them.

A blue and white object with a white background

Description automatically generated with medium confidence

Figure 11. Spatially Unaware K-D Tree Split

**5.3 Density Threshold and Data Structure**

Examined the distribution produced using a density threshold and the Wasserstein distance derived from it, one might have noticed that the Wasserstein distance measured in both data structures was high for house data. This made sense, as house data was the only dataset mentioned above with a relatively different structure preserved by the two subsampling methods, so a larger difference between them was expected. The Wasserstein distance measured in the octree, produced by randomly reduced tree data, was also large, potentially because, although the two subsampling methods preserved the basic characteristics of a tree structure, the imbalance of tree leaves still made the two tree point clouds distinguishable. Surprisingly, the Wasserstein distance of the k-d tree, which appeared more uniform in visualization, was larger than that of the octree. This would be investigated further in the future.

**6. Limitation and Future Works**

Both reduction methods used in this study might have compromised the original structure of the point cloud object, as evidenced by the loss of tips in some visualizations. Notably, among the visualizations lacking tips, all were produced using k-d tree methods, and most underwent uniform reduction. The scope of this study was limited, as it only examined two data structures and visualized a small variety of objects, all of which had a significant bulk of area on top. Future studies could have benefited from incorporating other data structures, such as R-trees. Additionally, there were various ways to implement octrees and k-d trees, each potentially impacting data storage differently. This study provided insights only into the specific data structures it implemented. Future investigations should have also considered the speed of data extraction and the memory requirements for storing point clouds in these data structures, beyond just how points were distributed.

**7. Conclusion**

This study involved visualizing two commonly used data structures for point cloud analysis—octree and k-d tree—using three distinct point clouds. Two of these point clouds were further reduced using methods provided by CloudCompare. Network visualizations then depicted how each data structure organizes point clouds hierarchically. The analysis focused on how subsampling methods might degrade the structure of the point cloud, noting that the data structures were indifferent to the presence of data noise. In this study's implementation, the octree was found to be more space aware, while the k-d tree was more point-aware. Additionally, it was observed that in the k-d tree, child nodes at deeper depths decayed more slowly than those in the octree. The study also measured Wasserstein distances to evaluate how the same type of objects, when subsampled differently, decayed differently with density thresholds. Notably, two out of three larger differences in the octree indicated that objects, despite being the same kind, were stored quite differently in this octree if the point distributions produced by the two methods varied. This occurred because the octree, being space aware, was more susceptible to imbalances in point distribution, unlike the k-d tree, which consistently split the data by first finding the median. In the future, the study would examine how data retrieval speed changes when data are stored in these or more than these two data structures.

**8.** **Reference**

3D Elevation Program | U.S. Geological Survey. https://www.usgs.gov/3d-elevation-program

Dulal, Saurab. Octree Construction and Nearest Neighborhood Search(NNS). https://dulalsaurab.github.io/computerscience/octree-construction-and-nns/

Laefer, Debra F. & Vo, Anh-Vu. 2019 LiDAR Data Collection for Sunset Park, Brooklyn, NY. FDA. https://archive.nyu.edu/handle/2451/60458

Rieck, Bastian. Topological Data Analysis for Machine Learning. https://bastian.rieck.me/talks/ecml\_pkdd\_2020/