3D point cloud processing: Deep Learning-Based Point Cloud Compression using Compressed Geometric Arrays

The demand for 3D modeling has increased due to the emergence of immersive applications and advancements in virtual reality-related technologies. As a result, 3D visual representation formats have become essential for 3D image and video processing in various fields, including medicine, autonomous driving, robotics, remote sensing, and urban planning. A point cloud is a set of three-dimensional points without structure that consists of two components: geometry and feature. Geometry represents the position of each point, while features are additional information attached to each point that describes their appearance. Point clouds have become an important 3D structure format for displaying objects and complex scenes due to their advantages, such as simpler representation for storage and processing, storing the original geometry information without discretization, immersion, and providing high interaction to the user. To capture point clouds, multiple methods are employed, including camera arrays, LiDAR sensors, and RGBD cameras like Kinect, RealSense, and Apple's depth cameras. Point clouds generated by camera arrays are typically dense and uniformly sampled. RGBD cameras generate point clouds that are dense, uniform, and prone to obstruction, and can be displayed in a 2D image format with color and depth components. LiDAR sensors commonly used in self-driving cars capture point clouds that can be roughly displayed in 2D with depth and reflectance components. LiDAR point clouds are generally sparse and unevenly sampled in Euclidean space but more regular and dense in spherical space, which represents the sensing mechanism and is uniformly sampled. These advanced 3D capturing techniques enable realtime, cost-effective, and easy acquisition of 3D point cloud data with high accuracy and quality. In the past, classical methods were used to process point clouds. However, with the expansion of artificial intelligence algorithms and the development of neural networks, deep learning methods have replaced classical methods. These deep learning methods offer more accuracy and less execution time, resulting in significant progress in various fields. Many datasets have been developed as a result of deep learning's remarkable results in processing point clouds, including ShapeNet, ModelNet, ScanObjectNN, PartNet, Semantic3D, ScanNet, ApolloCar3D, the KITTI Vision Benchmark Suite and S3DIS. These datasets provide many tasks for processing point clouds, such as point cloud segmentation, point cloud classification, and object detection.
This project begins with a thorough analysis of several point cloud processing methods, such as tracking, segmentation, detection, classification, and compression. Then a new method for point cloud compression is proposed. The main contributions of this method are as follows:

1) Making Use of the Compressed Geometric Arrays (CGA) structure: In the project, the CGA structure is utilized as a point cloud representation, which minimizes memory usage and improves search performance over more complex formats. Moreover, CGA simplifies operations related to finding the closest point, calculating chamfer distance, and choosing the centroid, which results in faster patching procedures and lower energy consumption.
2) New Patching Technique: The project presents a cutting-edge technique for creating patches out of a point cloud. A point cloud's points are unrelated to one another and do not exhibit regular voxel-space correlation or connectivity, in contrast to two-dimensional images. The method carefully calculates the total number of patches and the number of points in a patch in order to better capture local structure. A reliable method for maintaining

local information is ensured by the generation of patches through the use of point sampling and KNN (k-nearest neighbors) operation.

Compressed Geometric Arrays Structure (CGA)

CGA consists of five arrays with. Zindex vector, which represents the z coordinates of the existing points in the point cloud. Yindex vector, indicating the y coordinates of the corresponding points in the Zindex array. Ypointer vector, which stores the cumulative number of existing points with the y coordinates equal to Yindex[i], is defined recursively as follows:

- Ypointer[0] = 0
- Ypointer[i + 1] = Ypointer[i] + the number of points with y coordinates equal to Yindex[i]

An Xindex vector, showing the x coordinates of the points from the Yindex array. An Xpointer vector, which similarly stores the cumulative number of points with the x coordinates equal to the index of the Xpointer vector, following this recursive relation:

- Xpointer[0] = 0
- Xpointer[i + 1] = Xpointer[i] + the number of points with the x coordinate equal to i and distinct Y values.