REDUCING DATA REDUNDANCY IN 3D LIDAR POINT CLOUD USING OCTREE-BASED CODEC

Dr. PL. Chithra[1]*, S. Lakshmi Bala[2]

Abstract:
Efficient data compression techniques are increasingly necessary for processing LiDAR (Light Detection and Ranging) data. To address this need, a 3D point cloud data compression methodology based on Octree coding is presented in this paper. This codec involves quantizing the point clouds and compressing the 3D point cloud data using arithmetic encoding. Decompression is the reverse process used to reconstruct the original point cloud data. 3D LiDAR point cloud compression using Octree codec is a lossless technique that reduces the size of the data by identifying and removing less relevant data points while preserving the accuracy of the point cloud data. This approach enables easier storage and processing of the data, and the compression time for 3D point cloud data is calculated during the compression process. The proposed method is compared with the run-length compression method and Huffman coding method. It is found that the Octree codec with quantization and Arithmetic encoding outperforms existing Huffman coding and run-length coding.

Keywords:- Arithmetic coding, Decoding, Encoding, Huffman coding, LiDAR, Octree, Point Cloud Data, Quantization, Run-length coding.

[1]*Department of Computer Science, University of Madras, Chennai, India,
E-mail:- chitrasp2001@yahoo.com
[2]Department of Computer Science, University of Madras, Chennai, India,
E-mail:- lakshmibala.sj@gmail.com

***Corresponding Author:** - Dr. PL. Chithra
*Department of Computer Science, University of Madras, Chennai, India,
Email:- chitrasp2001@yahoo.com

DOI: - 10.48047/ecb/2023.12.si5a.0153

I. INTRODUCTION:

Point clouds are a valuable data representation that accurately and comprehensively captures spatial information about the surrounding environment [1]. To reduce the size of point cloud data without sacrificing its essential information, the technique of 3D LiDAR point cloud compression is employed [1]. For compressing static and dynamic point clouds, a two-stage deep learning system called Voxel Context-Net is used, with voxel-based and octree-based methods. In this methodology, the octree codec is utilized to compress the octree-structured data using voxel context [2]. While the file-based octree is powerful for sizable 3D point clouds, it slows down significantly when processing longer objects such as tunnels and corridors, compared to a memory-based approach [3]. To address this issue, a novel method for predicting the probability of geometry coding is used, which employs a denoising neural network on a "noisy" context cube containing both surrounding decoded voxels and uncoded voxels. A convolution-based technique is used to up-sample the decoded point cloud on the decoder side at a coarse resolution [4]. A "truncated octree" structure, which replaces the deep octree with a collection of shallow sub-octrees, can improve the compression ratio while maintaining the original structure. This approach reduces storage requirements without compromising the quality of the data. A variable length addressing technique is involved, which allows for the selection of the node address length of an octree based on the degree of truncation [5]. The embedded wavelet image compression method has been discussed. It tracks the significant pixels in multiple directions within the LL sub-band and uses quad-tree partitioning to locate the scattered significant clusters in other sub-bands [6]. The truncated octree is (T-Octree) used for encoding unstructured point clouds. A variable length addressing scheme is used for providing efficient query performance and storage savings for sparse data has been described [7]. In image compression method for nonlinear analysis transformation, uniform quantizer, and nonlinear synthesis transformation. The convolution linear filters and nonlinear activation functions are applied in three stages. It performs in the visual quality of all images has been used [9]. The methodology for performance assessment in the compression of dynamic point clouds for streaming and broadcast in mixed and virtual reality applications has been used in point cloud coding to enable consistent comparison [10]. The deep compression algorithm is used for reducing the memory footprint of LiDAR point cloud data. It obtains the sparsity and structural redundancy of points to achieve a higher compression rate. In the encoding process, the point cloud is into an octree and employs a tree-structured conditional entropy model is discussed [11]. Color image compression algorithm uses a Multilayer perceptron neural network parallelism with compression techniques for improving the image quality by the algorithm that divides the image into blocks and applies either DCT or DWT for lossy or lossless compression. The methods are normalized by coefficients that are processed by the MLP utilizing the Backpropagation neural network described [12]. A method called ScanNet for limited datasets for supervised deep learning in RGB-D 3D scenes like object classification, voxel, and CAD model retrieval has been discussed [13]. An effective framework for video-based person identification is discussed [14]. A new geometry partition and coding scheme is used in point cloud compression using Octree representation. In this approach, the quad-tree and binary-tree partitions are combined for asymmetric kd-trees with a symmetric Octree-based coding framework. It enables the fitting of 3D scenes with asymmetric bounding boxes and allows for skipping unnecessary bits during encoding [15,16]. O-CNN is an Octree-based convolution neural network designed for 3D shape analysis. It utilizes the octree representation of 3D shapes normal vectors from leaf octants as inputs and performs 3D convolution neural network operations on occupied octants of the shape surface [17].

This work is focused on compression for 3D LiDAR point cloud based on Octree-codec. Thus, using an octree codec is used for reducing the data redundancy based on quantizing the input data and performing an Arithmetic coding algorithm to encode. The rest of the paper is organized as follows: Section II describes the Methodology for the proposed work. Section III discusses the Performance Analysis of the proposed work. Section IV explains the Experimental Results compared with existing methods. And finally, the conclusion is discussed in Section V.

II. METHODOLOGY:

This paper presents Reducing data redundancy in a 3D LiDAR point cloud using an Octree-based codec has been proposed. Octrees are widely used data structures in 3D computer vision to efficiently manipulate and represent large collections of point cloud data. Octree-based point cloud compression recursively divides 3D

space into smaller regions, storing only the points that cannot be accurately represented by a simpler representation within each region. This significantly reduces data size, which is then encoded and decoded using Arithmetic coding. Fig. 1 illustrates the compression process, during which the elapsed time is calculated. Arithmetic coding is a lossless compression method used to represent and compress the point cloud data. In octree compression, arithmetic encoding is used to encode the data stored in each octree node, including the node's size, value, and pointers to its child nodes. The reverse process is then used to reconstruct the 3D point cloud data.

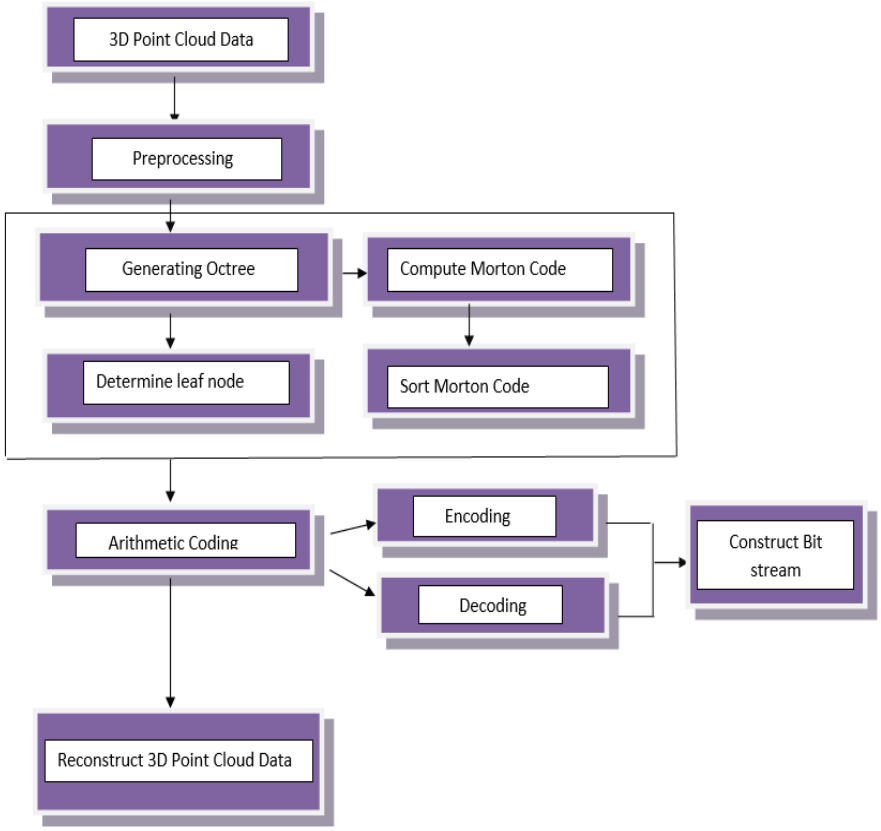


Fig.1. Block diagram for 3D point cloud compression based on Octree codec

A. Description of the Octree-Codec:

Octree compression is a powerful technique used to compress 3D point cloud data. Point clouds are collections of 3D points representing the surface of an object, generated by LiDAR systems, and often contain millions of points. Octree compression works by exploiting the spatial coherence of point cloud data and reducing the number of points that need to be stored. To construct the octree, the 3D space is recursively subdivided into smaller cubes until each cube contains a small number of points. The octree is traversed in a depth-first manner, and each node is processed. If a node contains a small number of points, it is replaced with a single point located at the centroid of the points. This reduces the number of points in the octree and removes some of the spatial redundancy in the point cloud. After constructing the octree, arithmetic coding is used to compress the resulting tree structure using a lossless compression algorithm. The compression is performed recursively, starting at the root of the tree and encoding the children of each node. The compressed data can be stored or transmitted more efficiently than the original point cloud. To decompress the compressed octree, the points are reconstructed by traversing the tree in a depth-first manner and decompressing the nodes as they are encountered. The result is a compressed 3D point cloud with reduced storage requirements and improved efficiency in transmission and processing.

Let n be the number of points in the point cloud, and let s be the size of the largest cube in the octree. The number of cubes in the octree, N, can be expressed as:

$N = (n / s\text{^}3) + 1$

After the octree has been constructed, each node that contains a small number of points is replaced with a single point located at the centroid of the points. Let k be the number of points in a leaf

node, and let p be the position of the centroid. The position of the new point, q, can be expressed as:
q = (k*p) / k
After the octree has been constructed and the number of points has been reduced, the resulting tree structure is compressed using a lossless compression algorithm. L.

Let B be the number of bits required to represent a node in the octree, and let C be the number of bits required to represent the compressed octree. The compression ratio, R, can be expressed as:
R = (B * N) / C

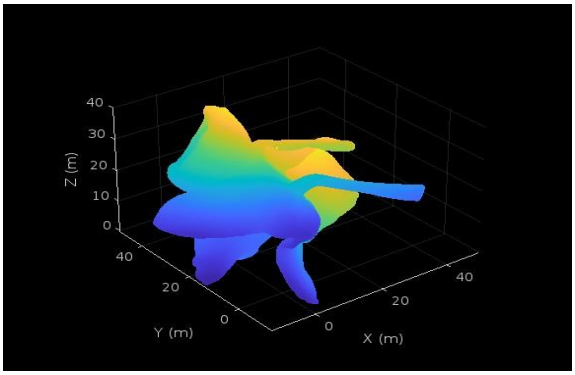
When the compressed octree is decompressed, the points are reconstructed by traversing the tree in a depth-first manner and decompressing the nodes as they are encountered. Let D be the number of bits required to represent a decompressed node, and let M be the number of nodes in the octree. The size of the decompressed point cloud, S, can be expressed as:
S = (M * D) / 8

Initialize the range [l, u] to [0, 1).
For each node in the octree, compute the probability p of the node using the number of points in the node and the total number of points in the octree.
For each bit of the binary code representing the node, update the range [l, u] using the probability p of the bit. If the bit is 0, set u = l + p * (u - l). If the bit is 1, set l = l + p * (u - l).
After all, bits have been processed for a node, output the lower bound l.

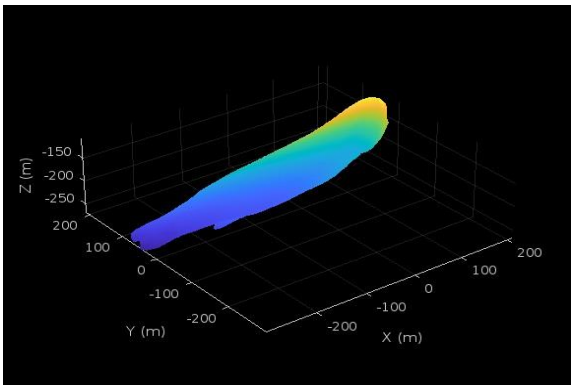
POINT CLOUD COMPRESSION ALGORITHM:
Input: Point cloud data (points, color)
Output: Compressed octree representation of the point cloud
Step: 1 Create the initial structure of the octree, which starts with a root node that represents the entire 3D space
Step: 2 For each point in the point cloud:
2.1 Each point in the point cloud is traversed from the root node to the smallest node that contains the point.
2.2 The threshold value for point cloud t = 150
2.3 If the node is a leaf node, and if the number of points in the node exceeds the threshold value, then the node is subdivided into 8 smaller nodes, and the points in the node are assigned to the corresponding child nodes.
Step: 3 Removes all the empty leaf nodes that do not contain any points.
Step: 4 Compressing the points in each non-empty leaf node using a lossless compression method. Arithmetic encoding is used for encoding the 3D point cloud data, and the resulting bitstream is used for decoding during the decompression process. The octree is traversed to retrieve the original point cloud data during decompression.
Step: 5 Stores the compressed octree structure and the compressed data for each leaf node
Step: 6 The point cloud is in a more compact form using the resulting tree structure of the octree, with only the leaf nodes containing the actual point data.

III. PERFORMANCE ANALYSIS:

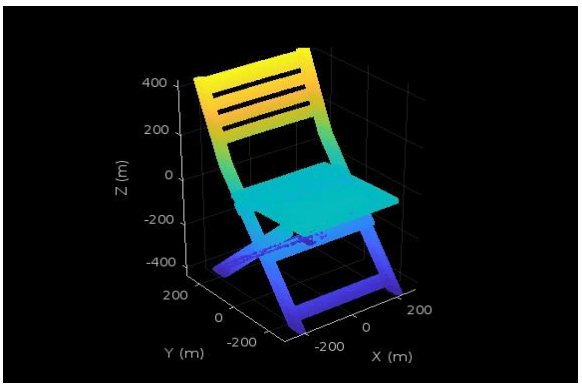
The proposed work aims to compress 3D color point cloud data using the Octree-codec method. The input data consists of five unstructured ply format point clouds, including Fig.2 (a) Flower, (b) Fish, (c) Chair, (d) Tiger, and (e) Gemstone, with a size ranging from size of 194818 × 3, 247503 × 3, 249773 × 3, 976257 × 3, 900506 × 3, taken from [8]. The compression process focuses on location attributes such as X-axis, Y-axis, and Z-axis.
To adjust the level of detail in the compression, the maximum depth and the minimum number of points per node parameters can be modified. A deeper octree with more levels will result in a more detailed compression. During the decompression process, the compressed octree structure and compressed data for each leaf node are decoded and decompressed. The decoded octree structure is used to reconstruct the original 3D space partitioning, and the decompressed data for each leaf node is used to reconstruct the original points in the point cloud data.



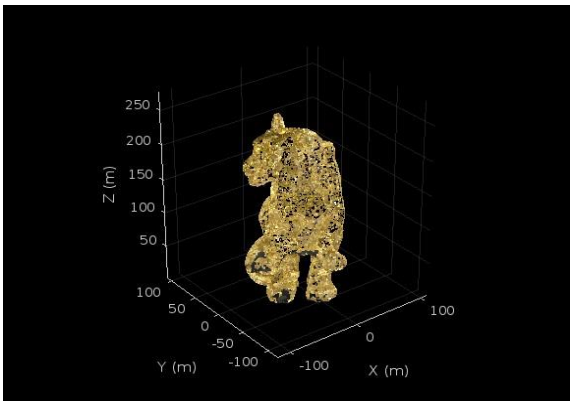
(a)



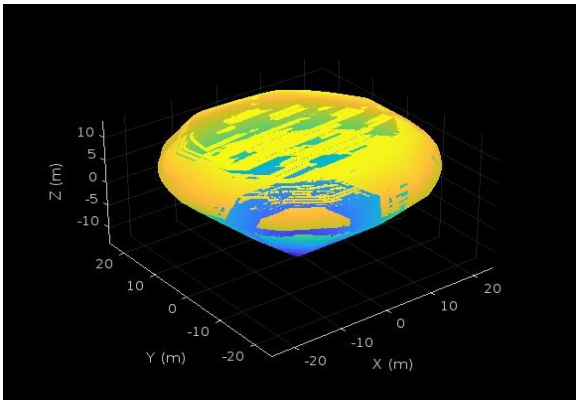
(b)



(c)

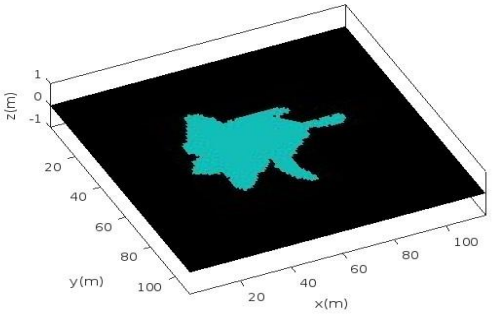


(d)

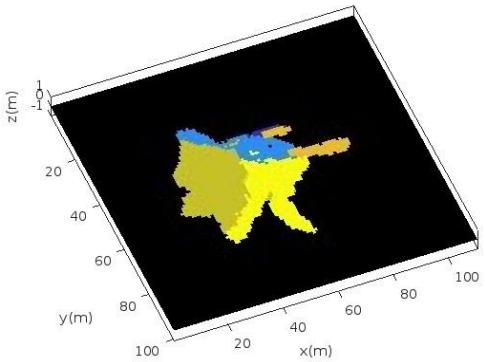


(e)

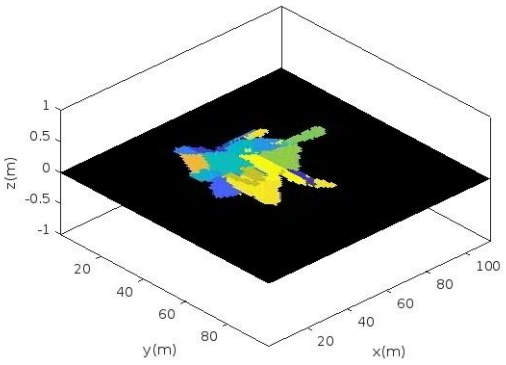
Fig.2: Original and synthetic 3D point cloud data (a) Flower, (b) Fish, (c) Chair, (d) Tiger (e) Gemstone.



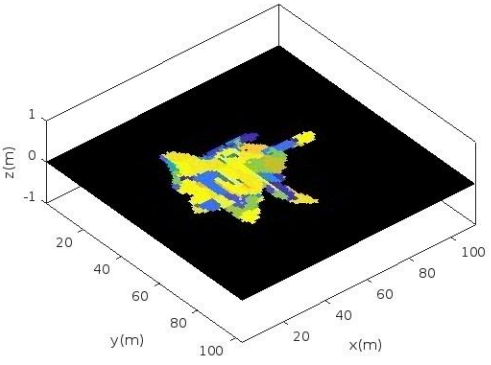
(f)



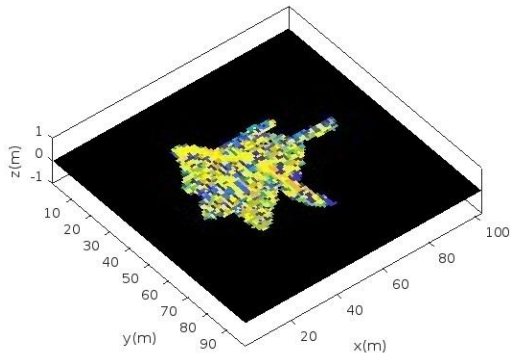
(g)



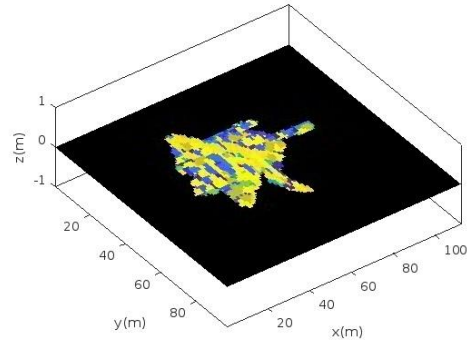
(h)



(i)



(j)



(k)

Fig:3 Levels of the compressed point cloud using Octree codec compression (f) Level 1 (g) Level 2 (h) Level 3 (i) Level 4 (j) Level 5 (k) Level 6 for the input 3D point cloud data.

Figure 3 illustrates the Octree codec works by recursively dividing the 3D space into smaller and smaller cubes called nodes until each node contains a small number of points or no points at all. The process starts at level 1, where the 3D point cloud data is quantized and the octree generation begins. As the octree is generated, the data is divided into nodes and leaves at level 2. At level 3, the data is split into eight child leaves, which increases the resolution of the compression. At level 4, all the leaf nodes that do not contain any points are removed, which further reduces the size of the compressed data. At level 5, the compressed octree structure and the compressed data for each leaf node are stored. Finally, at level 6, the resulting tree structure of the Octree represents the point cloud in a more compact form, with only the leaf nodes containing the actual point data.

Table:1 Performance efficiency of octree-codec

PCD Data	No. of Points Original	No. of Points Compressed	File size (MB) Original	File size (MB) Compressed using Arithmetic coding	File size (MB) Compressed using Huffman coding	File size (MB) Compressed using Run-Length coding
Fish	194818	175802	30.4	25.1	28.9	26.7
Flower	247503	216578	15.4	13.2	16.3	14.1
Chair	249773	208792	9.05	7.1	8.7	7.5
Tiger	976257	967812	38.2	30.9	36.8	33.4
Gemstone	900506	853264	32.6	27.4	32.9	29.1

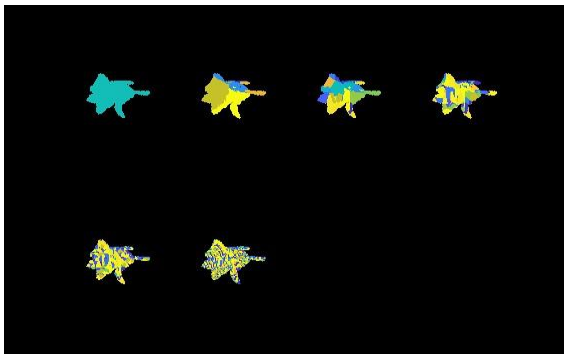
The performance of octree compression includes the number and distribution of points in the input point cloud, the depth of the octree, and the size of the nodes. The number and distribution of points can affect the efficiency of the traversal algorithm used for generating the octree, while the depth of the octree and the size of the nodes can affect both the compression ratio and the time required for compression and decompression.

To evaluate the performance of octree compression, various metrics have been used, such as compression ratio, compression time, decompression time, and distortion and error metrics. The octree compression is provided a balance between compression ratio and accuracy for 3D point cloud data, parameters have a significant impact on the performance of the compression and decompression operations. The level of detail and complexity in the point cloud data can also affect the compression and decompression speed and accuracy.

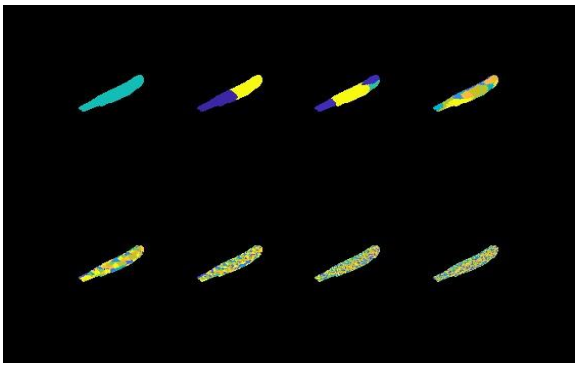
IV. EXPERIMENTAL RESULTS:

The Analysis of the performance of the proposed work to determine the effectiveness of the compression algorithm. Table 1 presents the results of the performance evaluation, including the original and compressed size of the input 3D point cloud data, calculated in megabytes using octree-codec. The table also shows the number of original points and compressed points, which can be used to calculate the compression ratio achieved by the algorithm. By analyzing the data in this table, it is possible to determine how effective the proposed octree compression

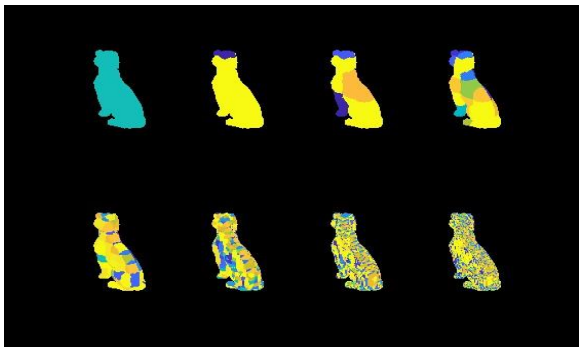
algorithm is at reducing the size of the input point cloud data while maintaining a high level of accuracy in the decompressed data.



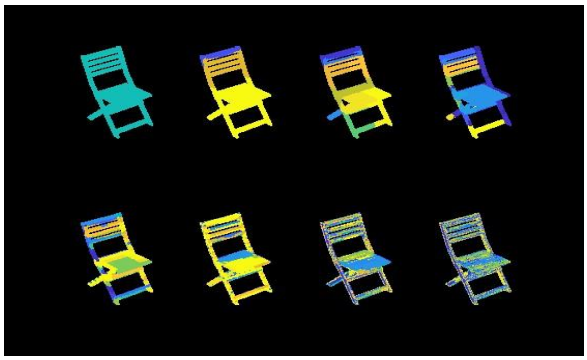
(l)



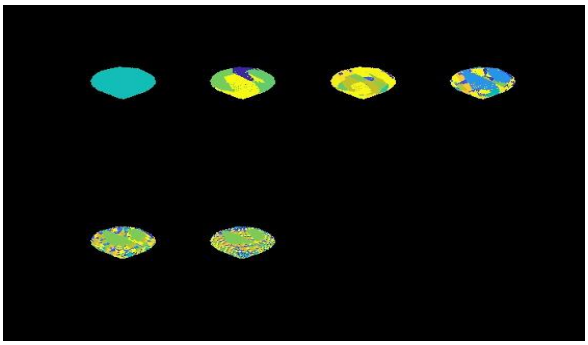
(m)



(n)



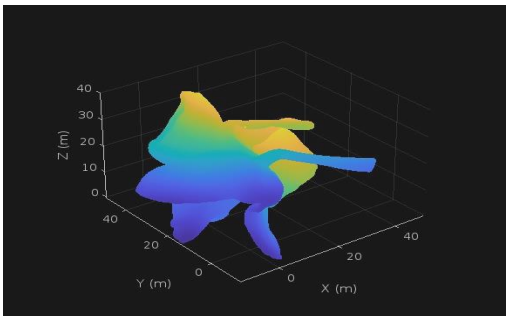
(o)



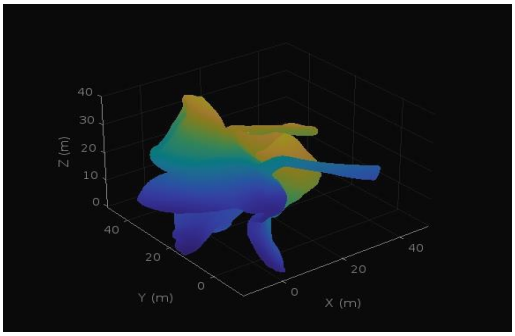
(p)

Fig. 4. Levels based on an octree codec compression

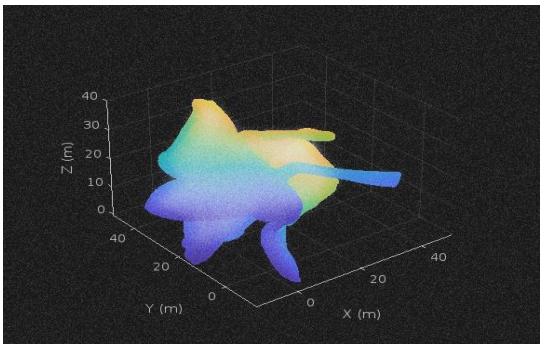
Figure 4 (l), (m), (n), (o), and (p) represents the levels based on the octree codec structure. The different levels of the decomposition for the input 3D point cloud data. As the density of the input data increases, more levels of decomposition are required to achieve the desired compression ratio. This is because a denser point cloud contains more points in a smaller area, which makes it more difficult to achieve a high compression ratio without losing significant information. Therefore, the octree codec structure will decompose the input data into more levels to ensure that the compressed data retain the necessary level of detail.



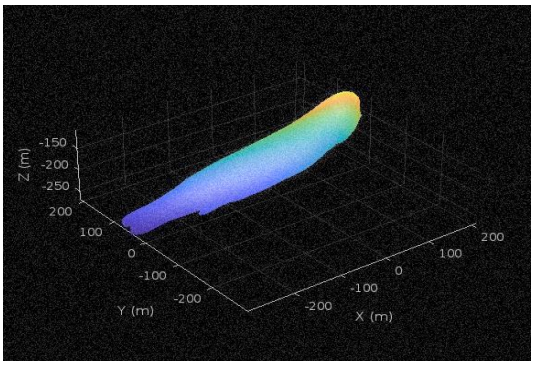
(q)



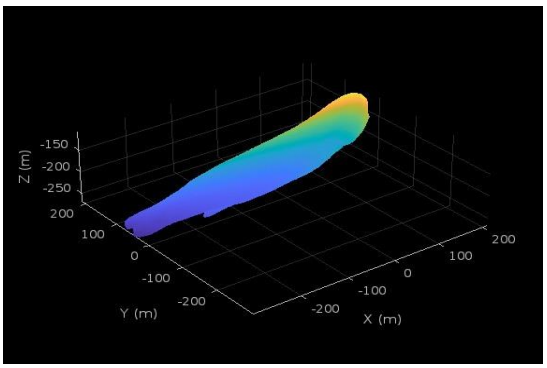
(r)



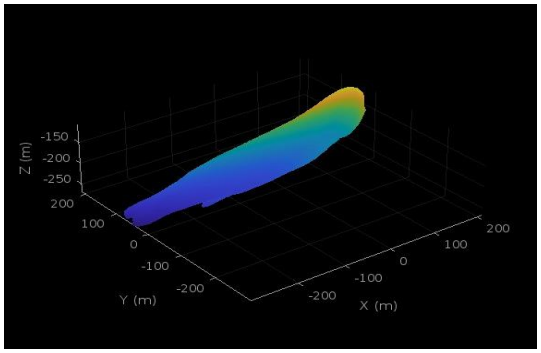
(s)



(v)



(t)



(u)

Fig: 5 Reconstruction of Input 3D Point Cloud

Figures 5 (q) and (t) depict the reconstructed 3D point cloud data using Arithmetic coding, while Figures 5 (r) and (u) illustrate the reconstructed data using the run-length coding algorithm. Figure 5 (s) and (v) explains the reconstructed data using the Huffman coding algorithms. The method employs Lossless compression, meaning that the reconstructed data retains all of the original information. The reconstructed data has been compared with the original 3D data to evaluate the effectiveness of the compression algorithm.

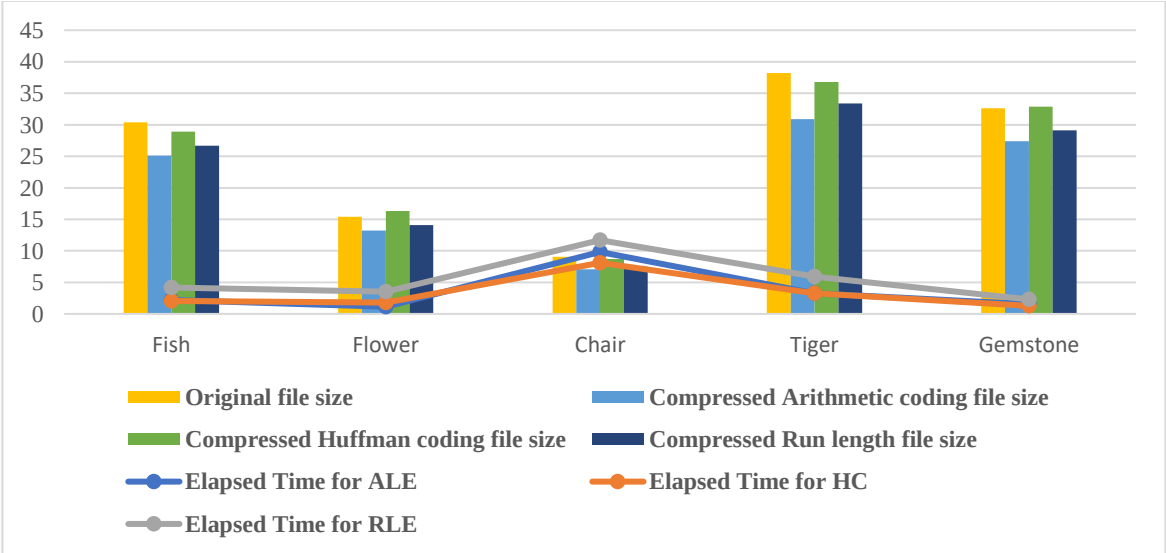


Fig:6 Graphical Representation of Performance analysis between Arithmetic encoding with Huffman coding algorithm and Run-length coding algorithm

The graph presented above in Fig. 6. displays the performance evaluation of three lossless compression algorithms, arithmetic encoding, Huffman coding, and run-length coding, for compressing 3D input point cloud data.

V. CONCLUSION:

In this paper, an Octree codec has been presented for compressing 3D point cloud data using Arithmetic coding, which has been compared and contrasted with Huffman coding and run-length coding. The experimental results indicate that Arithmetic coding offers the best compression ratio while minimizing the time required by the algorithm for 3D point cloud data. The use of Octree-based compression has proved to be an effective technique for compressing 3D point cloud data, achieving high compression ratios while maintaining the spatial coherence of the point cloud. This makes it a suitable method for applications that require efficient storage and transmission of 3D data. In the future, it may be possible to extend 3D point cloud compression using deep learning techniques within an Octree to enhance the point cloud data further.

REFERENCES:

1. P. L. Chithra and C. Tamilmathi A., "3D Color Point Cloud Compression with Plane fitting and Discrete Wavelet Transform," 2018 Tenth International Conference on Advanced Computing (ICoAC), Dec. 2018, doi: https://doi.org/10.1109/icoac44903.2018.8939106.
2. Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An Octree-based Framework for Point Cloud Compression," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2021, doi: https://doi.org/10.1109/cvpr46437.2021.00598
3. S. Han, "Towards Efficient Implementation of an Octree for a Large 3D Point Cloud," Sensors, vol. 18, no. 12, p. 4398, Dec. 2018, doi: https://doi.org/10.3390/s18124398.
4. Y. Mao, Y. Hu, and Y. Wang, "Learning to Predict on Octree for Scalable Point Cloud Geometry Coding," 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), Aug. 2022, doi: https://doi.org/10.1109/mipr54900.2022.00024
5. N. Koh, P. K. Jayaraman, and J. Zheng, "Parallel Point Cloud Compression Using Truncated Octree," 2020 International Conference on Cyberworlds (CW), Sep. 2020, doi: https://doi.org/10.1109/cw49994.2020.00009
6. P. L. Chithra and P. Thangavel, "A new efficient embedded wavelet image codec based on Multidirectional Traversal Algorithm," International Journal of Signal and Imaging Systems Engineering, vol. 5, no. 3, p. 196, 2012, doi: https://doi.org/10.1504/ijsise.2012.049852.
7. N. Koh, P. K. Jayaraman, and J. Zheng, "Truncated octree and its applications," The Visual Computer, vol. 38, no. 4, pp. 1167–1179, Apr. 2021, doi: https://doi.org/10.1007/s00371-021-02130-5.
8. Point Cloud Data set from: https://www.artec3d.com/3d-models/ply Accessed on December 18 2022.
9. Johannes Balle, Valero Laparra, and Eero P. Simoncelli. ´ End-to-end optimized image compression. In 5th International Conference on Learning Representations, ICLR, 2017.
10. Rufael Mekuria, Sebastien Laserre, and Christian Tulvan. Performance assessment of point cloud compression. In 2017 IEEE Visual Communications and Image Processing (VCIP), pages 1–4. IEEE, 2017.
11. Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020. 1, 2, 3, 5, 7, 8 [16] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In Proceedings of the 27th ACM International Conference on Multimedia, pages 890– 898, 2019.
12. PL.Chithra, A.Christoper Tamilmathi, " Effective lossy and lossless color image compression using Multilayer Perceptron", International Journal of Engineering and Technology, vol.7, pp. 9–14, 2018.
13. Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5828–5839, 2017.
14. D. Ouyang, Y. Zhang, and J. Shao, "Video-based person re-identification via spatiotemporal attentional and two-stream fusion convolutional networks," Pattern Recognition Letters, vol. 117, pp. 153–160, Jan. 2019, doi: https://doi.org/10.1016/j.patrec.2018.05.009.
15. X. Zhang, W. Gao, and S. Liu, "Implicit Geometry Partition for Point Cloud

Compression," Data Compression Conference, Mar. 2020, doi: https://doi.org/10.1109/dcc47342.2020.00015.
16. Zhang, X., et al. "Implicit geometry partition for point cloud coding." ISO/IEC JTC1/SC29/WG11 input document M49231 (2019).
17. P.-S. Wang, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," doi: https://doi.org/10.1145/3072959.3073608.