Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 3, March 2014, pg.351 – 358

RESEARCH ARTICLE

REAL-TIME COMPRESSION STRATEGY ON VARIOUS POINT CLOUD STREAMS

Miss R.SARANYA*[1]
M.Tech Student
Department of Computer Science and Engineering
PRIST University Pondicherry, India.
saranrya@gmail.com

DR.S.THIRUNIRAI SENTHIL*[2]
Head of the Department
Department of computer science and engineering
PRIST University Pondicherry, India.
razvi_zen@rediffmail.com

ABSTRACT

The Lossy based compression technique is used to compress the 3d image. These techniques exploit the spatial and temporal redundancy within the point data. To design an effective compression algorithm for point cloud computation by increasing its efficiency and space vector. so we perform a spatial decomposition based on octree data structures. By encoding their structural differences, we can successively extend the point clouds at the decoder. Another approach reduces coding complexity and coding precision. Our experimental results show a strong compression performance of a ratio of 14 at 1 mm coordinate precision and up to 40 at a coordinate precision of 9 mm.

KEYWORDS: ***Point Cloud, Coordinate Precision, Spatial Decomposition***

I. INTRODUCTION

Point clouds consist of huge data sets describing three dimensional points associated with additional information such as distance, color, normal, etc. Additionally, they can be created at high rate and therefore occupy a significant amount of memory resources. Once point clouds have to be stored or transmitted over rate-limited communication channels, methods for compressing this kind of data become highly interesting. So it provides point cloud compression functionality. It allows for encoding all kinds of point clouds including ―unorganized‖ point clouds that are characterized by non-existing point references, varying point size, resolution, density and/or point ordering. Furthermore, the underlying octree data structure enables to efficiently merge point cloud data from several sources. Compression algorithms can broadly be categorized in lossless (i.e. all information contained in the original data set can be reconstructed from the compressed representation) and lossy. For the latter, details are sacrificed — to a certain degree — in exchange for significantly higher compression ratios. Since in many applications the raw data is inherently

© 2014, IJCSMC All Rights Reserved

noisy due to the physical process involved in the acquisition system, a lossy compression scheme is preferable as long as the introduced distortion is kept below or close to sensor noise levels.

The main contribution of our system is its ability to efficiently detect and encode spatially and temporally redundant signal information from a stream of point clouds. In this context, we propose a modified octree data structure (double- buffering octree) that enables detection and differential encoding of spatial changes within temporarily adjacent unorganized point clouds. Another important contribution is the ability to reduce coding complexity by applying additional point detail encoding.

A. Octree Based Point Cloud Compression

An octree is a tree-based data structure for organizing sparse 3-D data. We will learn how to use the octree implementation for detecting spatial changes between multiple unorganized point clouds which could vary in size, resolution, density and point ordering. By recursively comparing the tree structures of octree, spatial changes represented by differences in vowel configuration can be identified.

An octree is a tree data structure suitable for sparse 3D data, where every branch node represents a certain cube or cuboids bounding volume in space. Starting at the root, every branch has up to eight children, one for each sub-octant of the node bounding box. An example is shown in Fig. 1 and in the left half of Fig. 2 . Common implementations create an object in memory for each node, holding 8 pointers to other nodes. If a child node does not exist, it is represented by a NULL pointer.

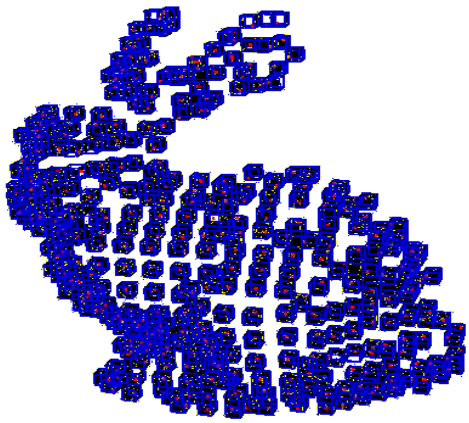
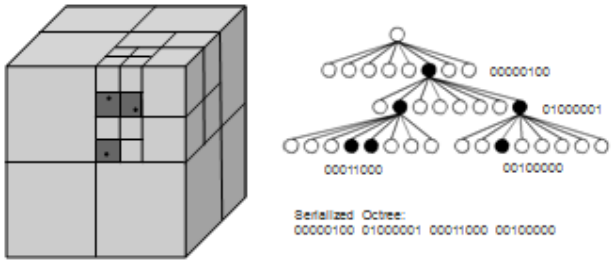


Figure 1. Sample octree for the Stanford bunny data set

The depth of the octree is limited either directly or by specifying a target leaf box size. When constructing the octree, every point is added iteratively by traversing the tree in depth first order starting at the root, and determining the correct child octant in every branch, creating new children as necessary until the point can be appended to the list of points stored in each leaf node. If only the leaf node occupancy is of interest, each leaf node contains a single Boolean variable.

If one sets a single bit for every existing child of a branch node, this child node configuration can be efficiently represented in a single byte, assuming some consistent ordering of the eight octants. This is the underlying idea behind the octree-based point cloud compression. By traversing the tree in breadth-first order and outputting every child node configuration byte we encounter, we are able to efficiently encode the point distribution in space. Upon reading the encoded byte stream, the number of bits set in the first byte tells the decoder the number of consecutive bytes that are direct children.

© 2014, IJCSMC All Rights Reserved

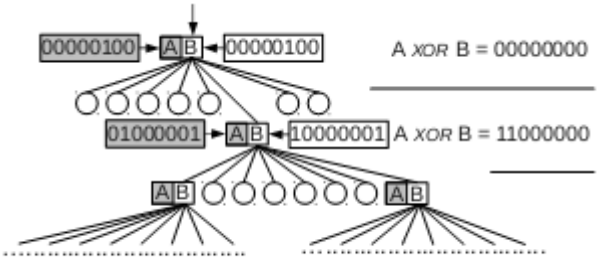


The bit positions specify the vowel/child in the octree they occupy. The right hand side of Fig.2 illustrates this byte stream representation for the octree in the left half. Note that this stream encodes the whole octree structure, and the size of this stream is n bytes, where n is the number of branch nodes. To fully reconstruct the point cloud, however, some additional parameters such as the root bounding volume and the octree depth need to be encoded.

The octree implementation provides efficient nearest neighbor search routines, such as "Neighbors within Voxel Search", "K Nearest Neighbor Search" and "Neighbors within Radius Search". It automatically adjusts its dimension to the point data set. A set of leaf node classes provide additional functionality, such as special "occupancy" and "point density per vowel" checks. Functions for serialization and deserialization enable to efficiently encode the octree structure into a binary format. Furthermore, a memory pool implementation reduces expensive memory allocation and deallocation operations in scenarios where octree needs to be created at high rate.

B. Temporal Compression

The output byte stream of the octree serialization is already a compact description of the spatial point distribution. Due to the lack of direct correspondences between points of different point clouds in the stream, the detection of changes within the point distribution is challenging. To this end, we propose a novel double-buffering octree scheme which extends our octree capabilities and enables comparison and differential coding of consecutive octree data structures.



(b) Differential encoding of a consecutive point cloud using pointer buffer *B*. Buffer *A* contains the previously processed octree structure.

Figure 2. Illustration of the differential encoding technique of consecutive octree data structures

This is achieved by adding a second set of eight child pointers to every branch node now containing a total of $2 \cdot 8 = 16$ pointers to child nodes. Initially, the collection of all pointers in all branches belonging to the first pointer set are dubbed the first child node pointer buffer A and all pointers belonging to the second pointer set constitute the second pointer buffer B. From there on, the two child pointer buffers are alternately assigned to every incoming point cloud. This interleaved storage of two consecutive point clouds in one octree data structure allows us to spatially match reference and analyze the structural changes within consecutive octree structures.

Initial to every incoming point cloud, the root node of the currently assigned pointer buffer is initialized with zeros. Whenever new child nodes need to be created, we perform a look up to the preceding pointer buffer within the current branch node. If no corresponding reference pointer can be found, the corresponding vowels did not exist in the previously processed octree structure and a new child node is created. If, however, a corresponding child node pointer can be found in the preceding

© 2014, IJCSMC All Rights Reserved

pointer buffer, we simply adopt its reference and initialize the selected child pointer buffer with zeros. Furthermore, we expand the size of the octree whenever incoming points violate its bounding box. This way, we continuously ensure structural consistency to the preceding point cloud and corresponding octree data structure by adaptively matching spatially related vowels/octree nodes.

However, in order to obtain and output the structural changes for every branch, we apply an exclusive disjunction operation (XOR) on the two bytes representing the child node configurations of the child pointer buffers A and B.This way, the serialized output for all regions in the scene that have not changed in occupancy is zero which leads to reduced entropy of the output stream.

Implementation

Each uncompressed point is described with 3 float values for x, y, z, for a total of 12 bytes per point (bpp). The symbol frequencies of the range coder are encoded with 4 byte integer resolution. . In our experiments, we use a single threaded implementation of the proposed compression technique running on a standard consumer PC.

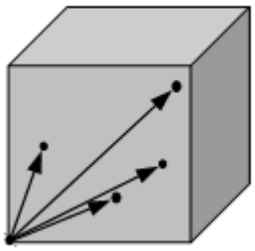


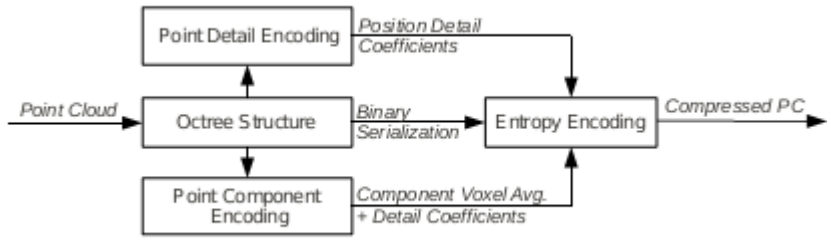
Figure 3. Illustration of Point Details Encoding

The principle of point detail coding can be applied to different kinds of point component information that is assigned to each point, e.g. color, normal's, etc. Instead of referencing the vowels origin coordinate, we compute and transmit the average color and perform an encoding of the color differences analogously to the Cartesian point detail differences. For other information, e.g. surface normal vectors, different approaches have been shown to perform well, e.g., converting the normal's from Cartesian space to polar space first. Assuming similar component information of neighboring points, the differentially encoded point components are characterized by reduced entropy which is exploited during subsequent entropy coding. Since point clouds can be enriched with arbitrary additional dimensions that can exhibit largely varying properties, it makes sense to analyze and fine tune these encoding details as needed instead of relying on one generic approach**.**

II. COMPRESSION ARCHITECTURE

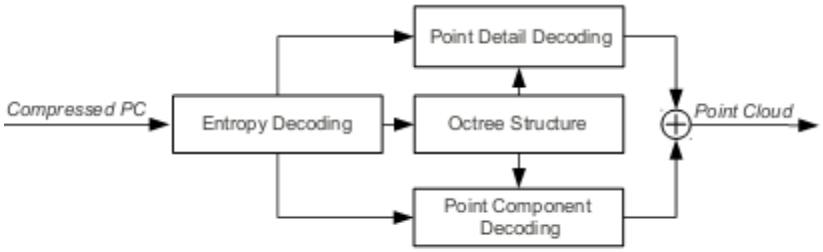
The resulting compression and decompression architecture is illustrated in Fig. 4. In Fig. 4(a), the encoder builds the doubled buffered octree structures and extracts their XOR encoded binary sequence during serialization. Voxel-local point details are extracted and encoded separately during the performed serialization process, which generates a stream of position detail coefficients (all relative to their respective leaf vowels center). Similarly, additional point components such as color and normal's are serialized into a separate stream each, containing the component vowel averages and corresponding detail coefficients.

© 2014, IJCSMC All Rights Reserved



(a) Encoder

Since their respective symbol frequencies are possibly different, these streams are consumed by the entropy encoder separately and multiplexed into a joint output data stream, which constitutes the compressed point cloud data to be written to e.g. a network stream or file. Note that for speed considerations, we use an integer-arithmetic variant of an arithmetic coder for entropy coding, a so called range coder. Prior to every entropy encoded bit stream, corresponding symbol frequency tables are additionally en- coded.



(b) Decoder

Figure 4. Schematic Overview of the Encoding and Decoding Architecture

The decoder (illustrated in Fig. 4(b)) performs demultiplexing followed by entropy decoding of the individual data streams. The serialized binary octree description is fed to the respective deserialization routines used to decode and reconstruct a coherent double buffered octree structure. Whenever leaf vowels are constructed during the deserialization process, their absolute position is fed to the point detail decoding and point component decoding routine. This enables us to recover the original point set of the encoded point cloud together with its respective point component information.

III. PERFORMANCE ANALYSIS OF POINT DETAIL CODING

Fig. 5 and Table I show the compression performance for the dynamic point cloud test sequence as a function of coordinate precision for the static octree encoding (frames compressed independently) and the proposed double buffering differential octree encoding, both without point detail encoding. Here, only the octree structure of the first point cloud is intra encoded. Shown in gray bars, the static octree compression reaches on average about 1.75 bpp at a point precision of 0.5 mm. By encoding the stream with the differential octree compression method, we significantly reduce the data rate to1.15 bpp, which is an improvement in performance of 34 %. This is mainly due to an increased amount of zero symbols in the serialized octree byte stream (up to 70%) reflecting unchanged branch nodes.

© 2014, IJCSMC All Rights Reserved

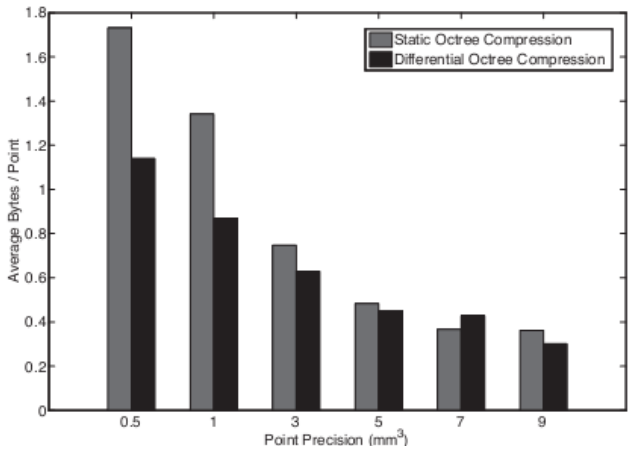


Figure 5. Comparison of the compression performance using static octree and the differential double-buffering octress, both without point detail encoding

With reduced compression precision, the compression ratio further increases and reaches a rate of about 0.5 bpp at 5 mm and of about 0.3 bpp at 9 mm. interestingly; the difference in performance between the static and the differential octree compression approach also decreases with reduced coding precision.

We further analyzed the compression performance when encoding additional point detail information (see Fig.12). In this experiment, we fixed the octree vowel resolution to 9 mm and used point detail encoding to achieve the different target resolutions.

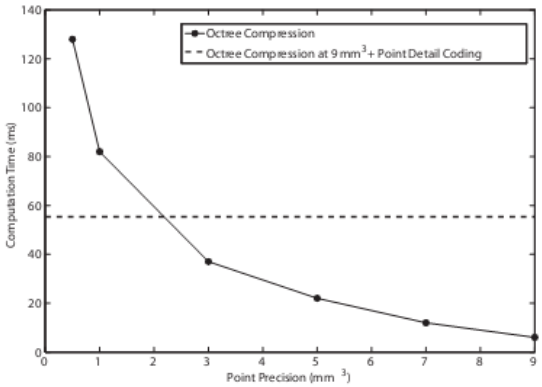


Figure 6. Computation time of differential octree coding per frame (point cloud) as a function of point precision is shown by the solid line. The computation time of differential octree coding at a fixed point precision of 9mmcombined with point detail coding is visualized by the dotted line

This offers two benefits. First, the octree structure occupies considerably less memory because of the reduced number of leave nodes. Second (and closely related), the computation time using point detail encoding becomes independent of the coding precision. This can be seen in Fig.6 (a). Here, the solid line represents the computation time for pure octree-based compression, which follows an exponential progression with increasing point resolutions. The combination of fixed octree compression and point detail coding (dashed line) on the other hand is constant in run time. At 9 mm resolution, we can observe the computation overhead of about 50ms when performing the additional point detail compression consisting of the calculation of point distances to vowel origin with subsequent entropy coding. As the complexity of the discretization of extracted vowel- local coordinates is independent from the applied quantization levels, this computation overhead stays constant regardless of the applied coding precision. While for lower resolutions, this constant time is higher than the time required for differential octree compression, there is a ―break-even point at around 2 mm. In addition, Fig. 6(b) shows the corresponding bpp rates as a function of coding precision. Black bars show the performance of the differential octree coding and the gray bars represent the overhead involved in

© 2014, IJCSMC All Rights Reserved

using the point detail coding approach. As can be seen in Figures 5 and 6 (b), for our test system, the gain in compression performance by using differential octree encoding are roughly canceled out when applying the point detail coding scheme, at the advantage of reaching constant computation times per frame.

The obtained results are to be understood as being qualitative, as they depend on the run-time behavior of the implementation that may vary between parameterization and different hardware setups (core processor, cache sizes, bus, memory, etc.). Note that additional performance could be also achieved by parallelizing the proposed approach. For instance, the compressed frame rate can be doubled

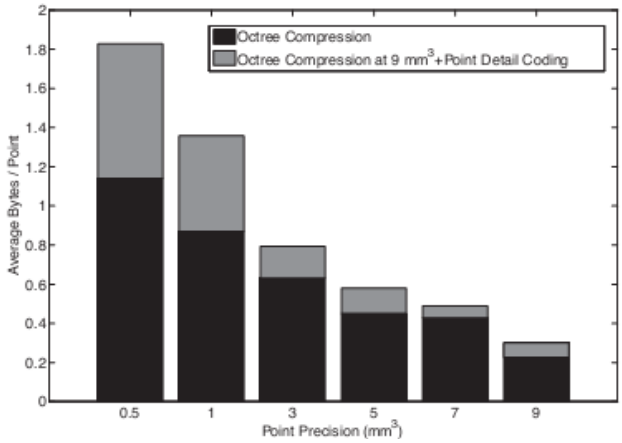


Figure 7. Compression performance of differential octree coding at different point precision levels is shown in black. The overhead in compression performance of differential octree coding at a fixed point precision of 9mm combined with point detail coding is shown in gray.

Relatively easily by creating two compression instances, one for even and one for odd frame numbers. This way, every frame is encoded relative to the second-to-last frame. Both streams can then be processed in parallel, as they are completely independent of each other, and the point differences will likely be only slightly bigger than differences between directly adjacent frames. Alternatively, one could prepartition the point space into subareas, rendering our approach suitable for e.g. multi- core architectures.

TABLE I. EXPERIMENTAL RESULTS

Compression Method	Precision	Bytes / Point	Compression
uncompressed	∞	12 bytes	**1:1**
static octree	0.5 mm	1.73 bytes	**1:7**
	1 mm	1.34 bytes	**1:9**
	3 mm	0.75 bytes	**1:16**
	5 mm	0.48 bytes	**1:25**
	7 mm	0.37 bytes	**1:32**
	9 mm	0.36 bytes	**1:33**
differential octree	0.5 mm	1.14 bytes	**1:11**
	1 mm	0.87 bytes	**1:14**
	3 mm	0.63 bytes	**1:19**
	5 mm	0.45 bytes	**1:27**
	7 mm	0.43 bytes	**1:28**
	9 mm	0.30 bytes	**1:40**

IV. CONCLUSION

In this paper, a novel online compression framework for unordered point cloud streams is presented. By differentially encoding changes within octree data structures, we can exploit spatial and temporal redundancies between consecutive point clouds. Furthermore; we introduce vowel-local point detail coding which enables online compression with increased point precision at reduced computation and memory requirements. Our experimental evaluation of the proposed compression scheme shows promising performance results at varying coding precision levels.

© 2014, IJCSMC All Rights Reserved

REFERENCES

[1] R. B. Rusu and S. Cousins, ―3D is here: Point Cloud Library (PCL),"in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, May 9-13 2011

[2] R. Schnabel and R. Klein, ―Octree-based point-cloud compression, "in Proceedings of the Euro graphics Symposium on Point-Based Graphics, Zurich, Switzerland, 2006, pp. 111–120.

[3] M. Waschbu¨ sch, M. Gross, F. Eberhard, E. Lamboray, and S. Wu¨ rmlin,―Progressive compression of point-sampled models,‖ in Proceedings of the Euro graphics Symposium on Point-based Graphics, Zurich,Switzer- land, 2004, pp. 95–102.

[4] B.Merry, P.Marais,and J.Gain,―Compression of dense and regular point clouds, "Computer Graphics, vol. 25, no. 4, pp. 709–716, 2006

[5] J.Zhang and C. Owen, ―Octree-based animated geometry compression, "in Proceedings of the Data Compression Conference, Snowbird, UT, USA, 2004, pp. 508–517.

[6] J.Zhang and C.Owen,―Octree-based animated geometry compression, "in Proceedings of the Data Compression Conference, Snowbird, UT,USA, 2004, pp. 508–517.

[7] G.Nigel and N.Martin, ―Range encoder range encoding: An algorithm for removing redundancy from a digitized message,‖ in Proceedings of the Video & Data Recording Conference, Southampton, UK, July 1979,p. 2427.

[8] R.Rusu, Z.Marton, N.Blodow, M.Dolha, and M.Beetz,―Towards 3d point cloud based object maps for household environments,‖ Robotics an Autonomous Systems, vol. 56, no. 11, pp. 927–941, 2008.

[9] J.Lengyel,―Compression of time-dependent geometry, in Proceedings of the 1999 Symposium on Interactive 3D Graphics, Atlanta, GA, USA,1999, pp. 89–95.

[10] S.Fleishman,D.Cohen-Or,M.Alexa, and C.Silva,―Progressive point set surfaces,ACM Transactions on Graphics (TOG), vol. 22, no. 4, pp.997–1011, 2003.

AUTHORS PROFILE

Miss R.SARANYA, Presently Pursuing Final Year M.TECH CSE, In PRIST University, Puducherry Campus, Puducherry, India

Dr.S.THIRUNIRAI SENTHIL,M.C.A,M.Phil.,Ph.D., Sri Vinayaka Mission University Salem, in 2010, Presently he is a Working Professor, Head Of Department in Computer Science and Engineering at PRIST University, Puducherry Campus, and Puducherry, India. The area of specialization is a Data Mining, Computational Biology and Bioinformatics about the professional analysis

© 2014, IJCSMC All Rights Reserved