COMPRESSION STRATEGIES OF 2D POINT CLOUDS

Marian Stan [1*]
Ionut Spataru [2]
Ion Bucur [3]

ABSTRACT

The purpose of this article is to describe the concept of point clouds and to present the main idea and the important steps made in this research domain. This article also presents methods that aim to solve the point clouds compression problem.

KEYWORDS: *point clouds, Delaunay triangulation, alpha shapes, marching cubes, quad-tree, KD-tree, Huffman.*

INTRODUCTION

Point clouds are just large collections of coordinates stored in a set. They can be stored in a bi-dimensional or a three-dimensional coordinate system, depending of the case, we may have X, Y or X, Y and Z coordinates. They are the basic structure in every new design process, and can be created using 3D scanners. When an object is scanned with such a device, a very large number of points is generated, the surface data file.

The usage of point clouds is getting bigger and bigger every day, few examples are 3D CAD models, metrology, animations and medical imaging.

The main problem with point clouds is that they cannot be used as they are, first they have to be converted to a polygon mesh, Nurbs surface etc.

The main techniques of converting these point clouds are for now Delaunay triangulation, alpha shapes and marching cubes.

THE NEED FOR POINT CLOUD COMPRESSION

We are living in a world where every bit counts, even with the great speed that we achieved on transfer between machines, we are still highly concerned about making every data-package smaller without any data loss.

Point clouds make no exception from this rule, their use increases every day, and new ways of making their size smaller and smaller emerge.

[1*] corresponding author, Engineer, ”Politehnica” University of Bucharest, Romania, marianstan7@gmail.com
[2] Engineer, ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, spataru.ionut91@yahoo.com
[3] Associate Professor PhD Eng., ”Politehnica” University of Bucharest, 060042 Bucharest, Romania, ion.bucur@cs.pub.ro

How can this data be compressed, how to make its space complexity smaller, because a big data set can be problematic. The goal is to achieve a smaller data set without any loss when compressing or uncompressing the data.

For this, we have used a variety of methods, each of them with different results. We've started with a bi-dimensional data, aiming to achieve new and better ways of compressing point cloud data.

PREVIOUS WORK

Up until now, the main techniques of manipulating point clouds where Delaunay triangulation, alpha shapes and marching cubes.

Delaunay triangulations

Both in geometry and mathematics, the Delaunay triangulation is, for a set of points (P in this example) a triangulation under the rule that no point from P is contained in a circumcircle of a triangle in DT(P). So this method, named Delaunay triangulation, maximizes the minimum angle of all the angles, of the triangles, inside the triangulation. Boris Delaunay is the one that named this method in 1934. There cannot be a Delaunay triangulation for a set of points that are on the same line. There is no unique triangulation for more than 4 points stored on the same circle.

If you consider circumscribed spheres, then the Delaunay triangulation can be extended to more than three dimensions. There can be generalizations, even other than Euclidean, but in these cases it is not a certainty that a Delaunay triangulation exists, or even be unique.

Alpha Shapes

In geometry, the alpha shapes are a group of piecewise linear simple curves in the Euclidean plane, group that is associated with the shape of a finite set of points. The alpha shapes were initiated by Edelsbrunner, Kirkpatrick and Seidel and derived from the convex hull, so that each convex hull is an alpha-shape but not every alpha shape is a convex hull.

For each real number R, define the concept of a generalized disk of radius 1/R as follows:

- If R = 0, it is a closed half-plane;
- If R > 0, it is closed disk of radius 1/R;
- If R < 0, it is the closure of the complement of a disk of radius -1/R.

After this, an edge is drawn between two of the members of the finite set of points. This edge will be drawn only if there is a generalized disk with the radius 1/R that contains the entire set of points and which has a property that the two points lie on its boundary.

Marching Cubes

First published in 1987 by Lorensen and Cline, the "Marching cubes" algorithm is used for extracting a polygonal mesh of an object, from a three-dimensional surface.

This algorithm works by taking eight neighbor locations at a time, this eight location are forming a cube. After this, the polygons that are needed to represent this cube are determined, merging the individual polygons in a surface is the final step.

THE COMPRESSION STRATEGIES

Quad-tree algorithm

One of the most successful algorithms is the quad-tree algorithm. It is aimed to compress a bi-dimensional array with very good results.

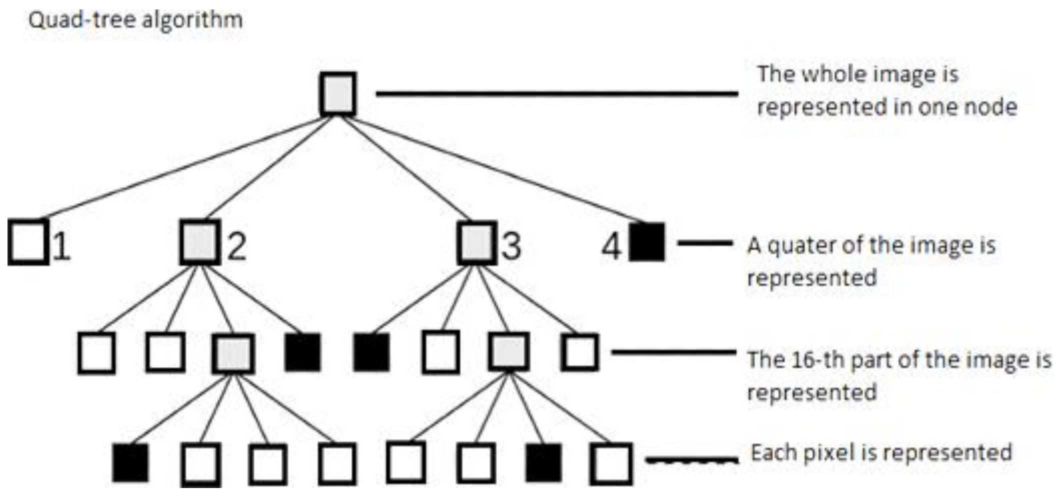
Quad-tree algorithm
The whole image is represented in one node
1
2
3
4
A quater of the image is represented
The 16-th part of the image is represented
Each pixel is represented

Figure 1. A quad-tree.

In figure 1 it is represented a quad-tree that contains information about the pixels of a black and white image, with a dimension of 8 * 8 pixels. The first node represents the whole image, the next nodes represent a quarter of the image and so on until each node represents a pixel in the image.

To represent a node, each zone is enumerated, from which that node is a part of, from the root to the leaf node. This would mean 6 bits, and it is the same as to represent a point in a space of 8 *8. In case it is wanted to represent all of the points, it is enough to enumerate the nodes and a number of bits will be obtained. This can be much more efficient than to represent each point through coordinates.

In figure 1, each node from the last level could be represented with 21 bits like so: 4 (level 1) + 8 (level 2) + 8 (level 3) = 21 bits, which is less than 48 bits (8 * 6) (3 bits for each coordinate), that is a compression rate of 57%.

For the whole picture of 64 points, it would only need 81 (4 + 16 + 64) bits, compared to 384 (64 * 6) bits that are now, this representing a compression rate of 79%. The more, the better, with each bigger test data, the compression rate improves.

To sum up, this is one of the best algorithms implemented, it has a space complexity of O(n + n log(n) + n / 2) and a time complexity of O(n + n log(n)) for encoding and a space complexity of O(n) and a time complexity of O(n) for decoding.

Overall, it manages to obtain a compression rate of about 71%.

KD-Tree algorithm

Another approach used for compression is the KD-Tree algorithm. This algorithm divides the space from one of the axis; usually this division is made through one of the points that are already in existence.

It should stop when it reaches a minimum dimension or when it reaches the minimum number of points per cell.

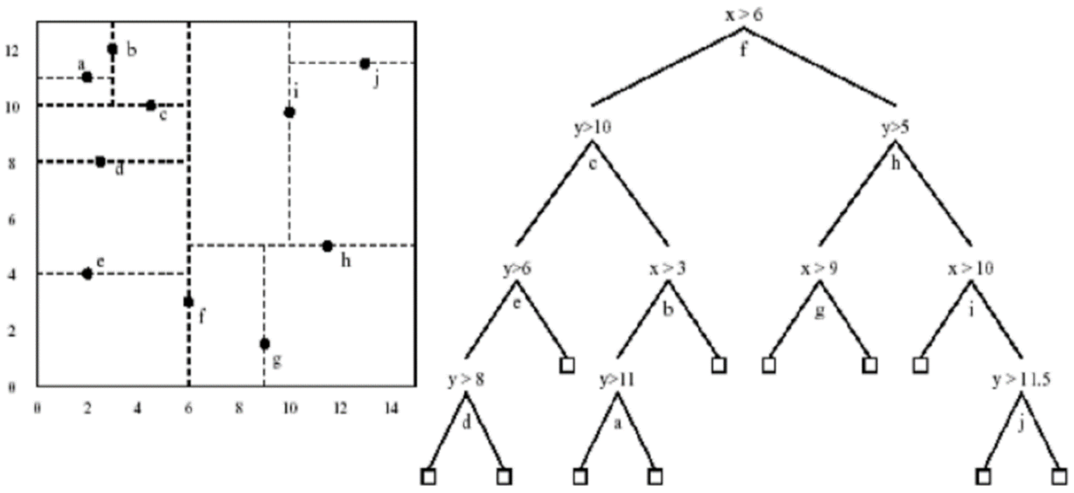

Figure 2. A KD-tree.

For example, in figure 2, it is a picture of a KD-tree with the dimension of $2^b * 2^b$, where b represents the number of bits. The most significant bit on the left is taken for this algorithm, which is 0, along with the most significant bit on the right which is 1.

It will stop when it reaches an empty cell or when it reaches the highest level, meaning 2*b (number of bits).

First, the interval of the parent node is divided in two. Each node will represent the number of points that it contains, it will keep only one of its ‘sons’, because the other one can be deduced afterwards.

The number of bits needed to write the number of points, is equal to the number of points read from the root node, so to represent a child node it is needed a number of bits equal to: log (number of points the ‘parent’ contains). This is represented in figure 3.

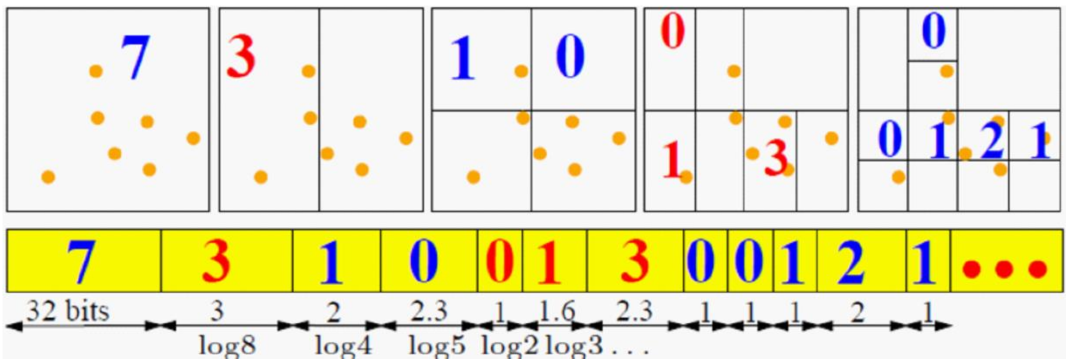

Figure 3. A KD-tree split scenario.

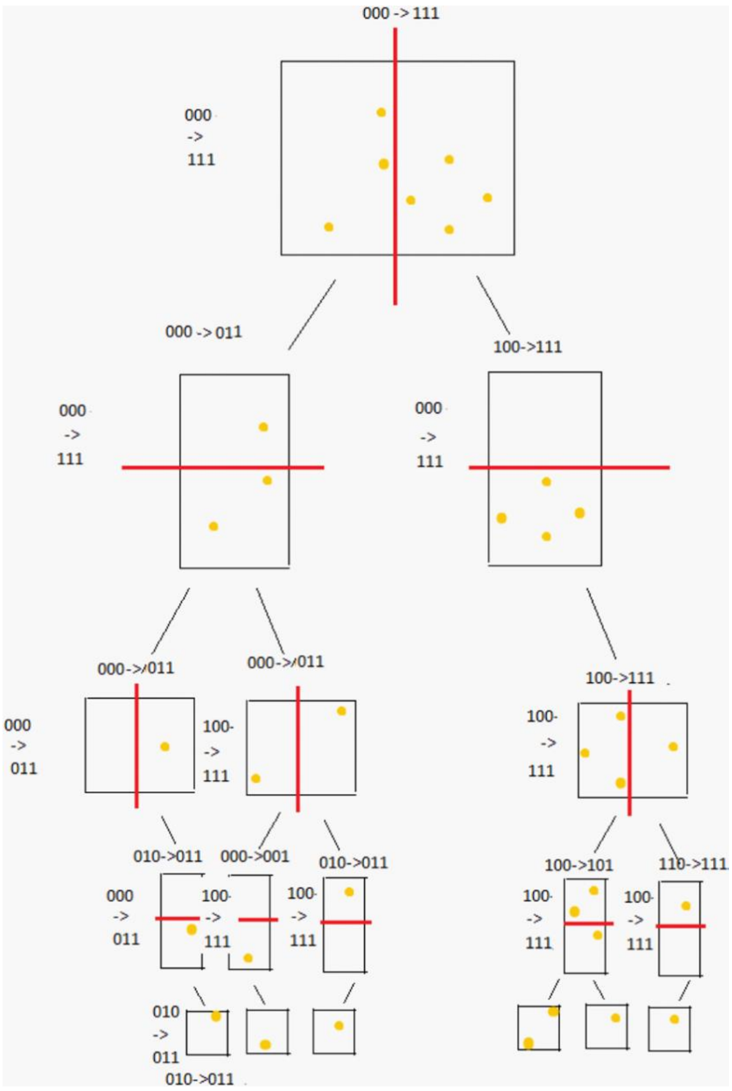

Figure 4. The representation of the resulted tree.

This algorithm has a compression rate of about 71% on small files and 78% on bigger ones.

The space complexity of the encoding of this algorithm is $O(n^2)$. The time complexity is O(n * log(n) + log(n)) * (dimension of the KD – Tree > n).

Huffman algorithm with move-to-front transform

Another algorithm used is composed from the following two steps:

1) Encoding the coordinate’s fractional part of each 2D point;
2) Compressing the result from step one with the use of Huffman algorithm and a move-to-front transformation.

The first step was introduced because the fractional part is mostly a random number, number that can’t be compressed with great success.

First step is sorting in ascending order each point by the fractional part of the X coordinate, fractional part of the Y coordinate, the whole part of the X coordinate and whole part of the Y coordinate. After this, the fractional part of each coordinate, is replaced with a vector of 216 elements, then the whole parts are put in place.

We will treat the fractional part and the whole part as a string, after concatenating them, the result will be used in key-value pair system where the resulting string is the key, and the number of occurrences is the value.

After sorting this vector, the result is represented by a number of elements that repeat themselves. To represent this vector only 123Kb are needed in our main compression scenario.

To decode this vector, it is expanded first, then the fractional part is repeated by the number that was initially present in the vector, regrouping each element, will return the initial vector.

In the second part of this algorithm, a move-to-front strategy is applied against the result obtained in the first part, followed by a Huffman encoding.

The move-to-front strategy is a method of encoding the data designed to improve the entropy encoding techniques. In this algorithm, the symbols used are inserted in a list, initially this list containing all possible symbols in a lexicographical order. Each processed symbol is replaced by its index from the list, and that symbol is moved in front of the list.

So the symbols that appear more often will be replaced with smaller values; this transformation is reversible without the need of any additional data.

The Huffman encoding is a method of compression data without any loss, using prefix codes. The most frequent symbols are replaced with smaller codes, and the symbols that show up less often are replaced with longer codes. The length of each symbol is inversely with the frequency of the symbol. Canonical Huffman codes are used to ease the encoding and decoding.

The time and space complexity of this algorithm are O(n * log(n)) and O(n) respectively for encoding and O(n) and O(n) respectively for decoding.

OBTAINED RESULTS

Table 1 summarizes the results of the aforementioned compression methods. The Quad-tree algorithm seems to be the best one, having the best compression rate, the best memory usage and some good compression and decompression times.

The KD-tree algorithm seems to be the worst one, because it has bigger execution time and memory usage, and the Huffman MTF algorithm has a slightly worse compression rate.

Table 1. Compression results

Algorithm	Test	Compression
Quad-tree	64c1	0.6757
	64c4	0.7395
	64c16	0.8022
	128b1	0.6436
	128b4	0.7088
	128b16	0.7715
	256a1	0.6476
	256a4	0.713
	256a16	0.7761
KD-tree	64c1	0.671
	64c4	0.7336
	64c16	0.7961
	128b1	0.6409
	128b4	0.7036
	128b16	0.7662
	256a1	0.646
	256a4	0.7087
	256a16	0.7712
Huffman with move-to-front transformation	64c1	0.6056
	64c4	0.6439
	64c16	0.7022
	128b1	0.5564
	128b4	0.5935

Algorithm	Test	Compression
	128b16	0.6631
	256a1	0.5164
	256a4	0.5764
	256a16	0.6525

FUTURE WORK

Introducing multithreading operation is one of the biggest improvements that can be achieved.

The compression may be improved by using a key-value pair system with a variable number of bits and using a Dynamic Huffman Algorithm.

The Huffman algorithm can be used on smaller chucks of data so that the local distribution of points can be exploited.

Stepping up to compressing 3D point clouds will be the next challenge.

CONCLUSIONS

Point clouds are, without any doubt, very important in medical and CAD systems. Creating, storing, and compressing them will always be a challenge, new algorithms and techniques will arise for each of these steps. We will continue our work on this subject because of the vast improvements that can be done.

We will end with an interesting fact, that proves the actual necessity to transfer fast and accurate large amounts of raw data: on December 29, 2014 the first-ever hand tool 3d model was emailed up to the International Space Station so that it can be printed there. It was the first object designed on Earth and printed in space. The 11.4 centimeter wrench can now be downloaded from everywhere because N.A.S.A made it public.

ACKNOWLEDGEMENTS

The authors would like to thank students Emanuel Ungureanu, Sabina Batranu, Ciprian Apetrei, Gabriel Ivanica and Dorian Dogaru for their great support and assistance with this paper.

REFERENCES

[1] Yvinec, Mariette. "2D Triangulation". Retrieved April 2010

[2] Pion, Sylvain; Teillaud, Monique. "3D Triangulations". April 2010

[3] Hornus, Samuel; Devillers, Olivier; Jamin, Clément. "dD Triangulations"

[4] Hert, Susan; Seel, Michael. "dD Convex Hulls and Delaunay

[5] T N. Akkiraju, H. Edelsbrunner, M. Facello, P. Fu, E. P. Mucke, and C. Varela. "Alpha shapes: definition and software". In Proc. Internat. Comput. Geom. Software Workshop 1995, Minneapolis

[6] Edelsbrunner, Herbert (1995), "Smooth surfaces for multi-scale shape representation", Invited Talk, Foundations of Software Technology and Theoretical Computer Science, Volume 1026 of the series Lecture Notes in Computer Science pp 391-412, 31 May 2005

[7] Lorensen, W. E.; Cline, Harvey E. (1987). "Marching cubes: A high resolution 3d surface construction algorithm". ACM Computer Graphics 21 (4): 163–169

[8] Nielson, G. M.; Hamann, Bernd (1991). "The asymptotic decider: resolving the ambiguity in marching cubes". Proc. 2nd conference on Visualization (VIS' 91): 83–91

[9] Montani, Claudio; Scateni, Riccardo; Scopigno, Roberto (1994). "A modified look-up table for implicit disambiguation of Marching cubes". The Visual Computer 10 (6): 353–355

[10] Rusinkiewicz, S. and Levoy, M. 2000. QSplat: a multiresolution point rendering system for large meshes. In Siggraph 2000. ACM , New York, NY, 343–352. DOI= http:// doi.acm.org/ 10.1145/ 344779.344940, june 2016

[11] Meshing Point Clouds A short tutorial on how to build surfaces from point clouds, http:// meshlabstuff.blogspot.ro/ 2009/ 09/ meshing-point-clouds.html, june 2016

[12] Costin-Anton Boiangiu, “The Beta-Shape Algorithm for Polygonal Contour Reconstruction”, CSCS14 – “The 14th International Conference on Control Systems and Computer Science”, Bucharest, Romania, July 2003, pp. 29-33

[13] Costin-Anton Boiangiu, Mihai Zaharescu, “Beta-shape using Delaunay-based triangle erosion”, in Proceedings of the 18th International Conference on Applied Mathematics (AMATH '13), Budapest, Hungary, December 10-12, 2013, pp.97-101

[14] Costin-Anton Boiangiu, Ion Bucur, Andrei Iulian Dvornic. “The Beta-Star Shape Algorithm for Document Clipping”. Annals of DAAAM for 2008, Proceedings of the 19th International DAAAM Symposium, Trnava, Slovakia, October 22-25, 2008, pp. 0127–0128

[15] Costin-Anton Boiangiu, Mihai Zaharescu, Ion Bucur - „Building Non-Overlapping Polygons for Image Document Layout Analysis Results”, Journal of Information Systems & Operations Management (JISOM), the Proceedings of Journal ISOM, Vol. 6 No. 2 / December 2012, pp. 428-436

[16] Costin-Anton Boiangiu, Bogdan Raducanu, Serban Petrescu, Ion Bucur, “Ensure Non-Overlapping in Document Layout Analysis”, 20th DAAAM World Symposium, Austria Center Vienna (ACV), 25-28th of November 2009, pp.327-328.