ISSN 2395-8618

Meta-heuristic Coefficient for Indexing Geometric Objects in Hierarchical Spatial Data-Structures

Gerdys E. Jiménez Moya and Jairo Rojas Delgado

***Abstract*—Hierarchical spatial data structures are usually employed for indexing geometric objects and are characterized by recursively decomposing the space. Due to this recursion process is necessary to define a decision criteria to determine when to stop the process of spatial decomposition. In this paper, a new recursion threshold for indexing hierarchical spatial data structures that is independent of the nature of the data is introduced. The objective is to reduce the execution time of space searches that arise in various applications of modern computing systems such as mining, solid modelling, simulation and others. Results indicate that the proposed recursion threshold reduces the execution time of space searches respect to others criteria of general purpose reported in the literature, however, RAM consumption is increased considerably.**

***Index Terms*—execution time, recursion threshold, spatial data-structure.**

I. Introduction

MODERN software systems are increasingly important to the development of human activity. Nowadays these systems implements several complex operations from a computational point of view related to spatial and time consumption indicators. Among these operations are the diminution of the computational cost when processing a geological block model[1], the estimation of mineral resources and the calculation of the mineral concentration located within two surfaces which are common operations of any mining software system.

In the problem of reducing the computational cost of processing a block model [1], a calculation of the mineral concentration and economic cost for each block is made. This is done as part of an optimization process to design an open pit mine. Here the goal is to maximize profits and to select the proper blocks for mineral extraction while the physical boundaries of the pit mine, usually with the shape of a regular inverted cone, are defined according to geometric constraints as illustrated in Figure 1a.

The time complexity required for finding the best combination of blocks using brute force is 2^n where n is the number of blocks and since there is a spatial relation between blocks and a geometric constraint in the physical boundaries of the mine pit, this problem can be seen as a spatial search problem.

G. E. Jiménez was with the University of Informatics Sciences, Geoinformatics and Digital Signals Center, La Habana, Cuba. e-mail: (gejimenez@uci.cu).

J. Rojas was with the University of Informatics Sciences, Geoinformatics and Digital Signals Center, e-mail: (jrdelgado@uci.cu).

[1]Block model: structure for modelling mine pits that represents the space through a set of regular blocks.

In the estimation of mineral resources operations, Figure 1b, there is an interest in calculate the value of the mineral concentration in a spatial region from which there is not any explicit measure. In order to obtain the mentioned value, a spatial search is performed around the region of interest and all measurements are considered as inputs for a spatial interpolation method that makes an estimation according to the values of the k-nearest neighbours. According to [2] the time complexity required to perform a k-nearest neighbours search using a naive algorithm for n query point is in the order of n^2 which is expensive.

Another example of the operations performed in modern mining software is showed in Figure 1c and is a special case of a spatial range search. In the figure, a calculation of the value of the mineral concentration within two surfaces represented by polygonal meshes is required. In order to do this, mining systems computes the intersection between meshes components and a search volume. This operation has a linear time complexity but still expensive since the intersection tests usually require a lot of computational effort.

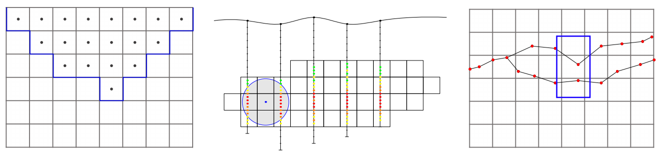

Fig. 1: Common spatial operations performed in the actual mining software.

In addition to the previous mentioned issues, there are other operations performed by modern software systems focused in tridimensional visualization that can be treated as spatial search problems. In order to achieve a higher performance of software and hardware, several methodologies have been developed, but usually the great amount of data and the complexity of the implemented operations leads to computational burden [3]. An open issue in this research field can be identified and it states as follows: how to reduce the execution time of spatial queries?

II. Spatial search operations

Spatial queries can be classified according to its basement: query based in phenomena and query based in location [4].

ISSN 2395-8618

Spatial queries based in phenomenon makes use of a semantic criterion associated to data to satisfy some constraint, while spatial queries based in location only makes use of geometric descriptions of the spatial data. In the latest years, the effort related to the diminution of the time complexity of spatial queries based in location have been intense because this kind of spatial query are context-independent[2]. In recent investigations two fields can be identified: parallel computing and spatial data-structures.

Despite parallel computing has showed good results in many application fields, there are several limitations related to the mandatory requirement of specific hardware and the difficulty of writing concurrent code. Until now, several spatial data-structures with the objective of performing spatial queries have been proposed: the Binary Space Partitioning trees (BSP), the Kd-trees, the R-tree and the quadtrees.

III. Hierarchical spatial data-structures

A spatial data-structure organizes geometric objects located in a dimensional space M according to some geometric and spatial constraint. According to [5] the organization of a spatial data-structure is usually hierarchical and its construction is based on recursive decomposition of space. Recursive decomposition of space refers to a systematic subdivision of a space in two or more subsets M_i according to Equation 1.

$$M = M_1 \cup ... \cup M_n i \neq j, 0 < i, j \leq n \tag{1}$$

Given the previous definition, a set of geometric objects can be partitioned in a set of discrete adjacent elements. These elements are usually primitives and can have different shape, size, position and orientation. For tridimensional spaces, the most common of these primitives are boxes and when are used, it's a regular decomposition process because boxes are uniform in shape, size and orientation [6]. Figure 2a shows a bidimensional space regularly partitioned through squares.

An example of non-regular space partitioning process is the resulting from the indexation of a BSP tree like Figure 2b. On its simplest form, a BSP tree use a line to divide the space in two subregions and then reorganize the geometric objects according to its position respect that line [4]. The process is repeated recursively until a condition is met. The previously mentioned condition is also known as stopping criteria or indexation criteria of the recursive space decomposition process and determines when to continuing subdividing the space or not.

The recursive space partitioning process defined by a Quadtree is an example of a regular one. With a Quadtree , space is subdivided by squares in four subregions of equal size and then geometric objects are reorganized according to its position respect to the corresponding portion of space [7]. Like BSP trees, Quadtrees recursively continue subdividing the space until a condition is met.

As can be expected, there are more complex variations for BSP trees [8], [9] and Quadtrees [10], [11], [12]. BSP

[2]Context-independent means that is not necessary to use semantic information associated to the spatial data, but only their geometric description

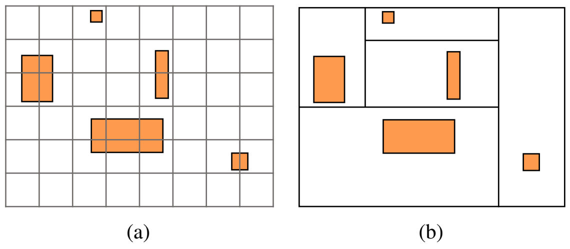

Fig. 2: Regular and no regular recursive space particioning process.

trees can be used for dimensional spaces higher than two by using planes in R^3 and hyperplanes in R^k for $k > 3$ also known as Kd-trees. Additionally, BSP trees can be Axis-Aligned and Polygon-Aligned when the subdivision line or plane is aligned to the axes of the coordinate system or not respectively. Quadtrees also have several variations used in R^3 named Octree and Hyperoctree in R^k for $k > 3$. In all cases a sort of recursion threshold or indexing criteria must be defined and according to [13] the setup of this recursion threshold has a direct impact in the ability of these spatial data-structure for solving specific problems.

For precise mathematical description, an Octree exhaustive definition will be given and in the following section, common studied recursion thresholds for Octrees will be enunciated although it can be generalized for BSP, kd-trees and others hierarchical spatial data-structures. Octrees were proposed for first time by [14] and extensively studied by [15]. On it's general form, an Octree is a tridimensional approximation of an object by a set of boxes. According to [16], given an Octree O, a set of geometric objects S contained in a restriction volume V associated to O and $\gamma(\{O, S, V\}) \to 0, 1$ a function that determines if the Octree O should be partitioned or not; the formal definition of an Octree is as enunciated in Equation 2.

$$\begin{array}{lcc} 1 & O & \text{if } |S| = 0 \\ 2 & O & \text{if } \gamma(O, S, V) = 1 \\ 3 & \{\{O_1, S_1, V_1\}, ..., \{O_8, S_8, V_8\}\} & \text{in other case} \end{array} \tag{2}$$

In Equation 2 the construction process of an Octree O stops if their restriction volume does not contain any geometric object or if the threshold function have been satisfied. In any other case space is recursively partitioned into eight subregions O_i calculating V_i and S_i.

A. *Recursion threshold for hierarchical spatial data-structures*

According to [13] the setup of the recursion threshold has a direct impact in the ability of spatial data-structure for solving specific problems. The recursion threshold has a direct influence in the amount of nodes generated in the hierarchical structure that at the same time determines the amount of recursions needed to process a spatial query. The behaviour for the time complexity in these structures is usually logarithmic while the search executes in the branches of the

ISSN 2395-8618

hierarchical structure and is linear when search is inside a specific node. Despite the importance of this configuration parameter, according to [13] very few have been written about this. Some applications can be seen in [17], [18].

In [19] for example a stopping criterion for an Octree construction is defined based on a priory knowledge in radiative transfer simulation but this stopping criteria can not be generalized to other fields. In [20] an Artificial Neural Network (ANN) is proposed to calculate an *information value* index that evaluates if to continue subdividing an Octree box given a specified threshold. However, very little information is given about this process, or about how to train the ANN or what kind of test were executed with this novel methodology. A review of the state of the art about recursion threshold functions reveal that there are three main criteria:

- **Recursion level**(α). The generation of nodes will continue until the level of a node O in the hierarchical structure gets higher than α.
- **Volume of the region**(β). The generation of nodes will continue until the volume of a region assosiated to a node O gets under β.
- **Number of geometric objects**(δ). The generation of nodes will continue until the number of geometric objects assosiated to a node O gets under δ.

Each of the previously mentioned indexation criteria has their own advantages and disadvantages and their adjust is context-dependent. This means that depending of the structure of the data to index, the spatial data-structure will be sensitive to formation of clusters and over generation of nodes. In the following section a novel indexation criteria will be introduced.

B. Meta-heuristic coefficient for indexing geometric objects in hierarchical spatial data-structures

Meta-heuristic refers to the use of some sort of experience or prior knowledge to approximate an acceptable solution to a given problem. Actual knowledge allows deducing that decision about when to stop the space partitioning is related to the size of the region to subdivide, the amount contained geometric objects and to the times that the space has been subdivided. These factors match with indexation criteria seen in Section III-A.

In Equation 3 function γ from Section III is defined as a new indexation criteria that establish a direct relation between the volume of a region associated to Octree O and number of geometric objects associated to O. At the same time the new indexation criteria establish an inverse relation with the recursion level of O.

$$\gamma(O,S,V) = \begin{cases} 1 & \text{if} \quad \epsilon < \phi \times \frac{|S| \times \sigma(V)}{\theta(O)} \\ \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In Equation 3, $\sigma(V)$ calculates the volume value for the restriction volume V associated to Octree O, S is the set of geometric objects contained in V. Index ϵ represents the recursion threshold and its value depends of factors: restriction volume, amount of geometric objects and depth of the node O.

Small values of ϵ increments the chance of subdivision of a node. When a node is subdivided, its children have less volume, less number of geometric objects and a higher value for their level in the hierarchical tree structure. Because of that, the chance of subdivision is smaller. The values for ϵ factor can be normalized between $0 <= \epsilon <= 1$ with Equation 4 with $0 < \epsilon < 1$. In Equation 4 V_{root} is the restriction volume associated to the root node and S_{root} is the set of geometric objects contained in V_{root}.

$$f(\epsilon) = |S_{root}| \times \sigma(V_{root}) \times \epsilon \tag{4}$$

Factor ϕ controls the influence of factors $|S| \times \sigma(V)$ and $\theta(O)$. For values of $0 < \phi < 1$ importance to factor $\theta(O)$ is granted, causing that the node has fewer chances of subdivision and the generation of children gets decreased. For values of $\phi > 1$ factor $\theta(O)$ gets less importance and the generation of children is bigger. Factor ϕ can be used to balance the effects of time and spatial complexity.

IV. Results and discusion

The following section describes a set of tests conducted to measure the time and memory consumption of spatial queries while using an Octree and a specific indexation criteria. Input data consist in point clouds randomly generated using a normal distribution with mean of 500 thousand and typical deviation of 200 thousand. A point cloud is a set of tridimensional points in the space, defined by X, Y, and Z coordinates, and often are intended to represent the external surface of an object. The generated point clouds has different sizes, starting with 100 thousand points until 2 millions points.

An Octree pointer variation data-structure was implemented in C++11 programming language [21]. Tests were executed in Xubuntu 14.04 operating system using a *Core i3* microprocessor and 4 Gb of RAM. The implemented Octree can construct an index given a point cloud using the indexation criteria specified in Section (III-A, III-B). An orthogonal spatial search was also implemented as described in Algorithm1 using a box as V parameter.

Algorithm 1 Orthogonal spatial search.

```
1:  function SEARCH(V: Search volume, O : Octree)
2:      R: Geometric Objects List;
3:      if INTERSECTION(V, O.V) = $True$ then
4:          if O.ISLEAF = $True$ then
5:              $R \leftarrow R+$ INTERSECTION(O.S, V);
6:          else
7:              for i = 1..8 do
8:                  $R \leftarrow R+$ SEARCH(V, $O.O_i$);
9:              end for
10:         end if
11:     end if
12:     return R;
13: end function
```

ISSN 2395-8618

Algorithm 1 takes two parameters, the search volume V and an Octree O. Procedure first determines if the restriction volume associated to O intersects with V. If this condition is not met then no other computation is performed and there is not other geometric object contained in V. Otherwise, if O is leaf, procedure computes the intersection of each geometric object with V, if O is not leaf, spatial search procedure is called for each node of O.

First thing to do is to configure the threshold for indexation. The time consumption for a spatial search using Octree has a direct relation with the indexation criteria and the volume of data, so this parameter must be adjusted to each specific data volume. Using this adjustment, a calculation of the time consumption of an orthogonal search for each indexation criteria is made in different scenarios. All measures are calculated as the average of 100 thousand orthogonal searches. At the same time, the number of generated nodes is calculated as a measure of RAM consumption for each indexation.

The setup of the recursion threshold is made using indexation criteria previously defined. For each indexation criteria the threshold value is considered as follows: $10^{14} < \beta < 10^{16}$ with step of 2×10^{14} volumetric units, $0 < \delta < 50000$ with step of 500 geometric objects and $0 < \epsilon < 1$ with step 0.01 and $\phi = 100$.

For example, using a point cloud of 100 thousand objects, the best scenario for δ is $\delta = 500$ geometric objects and the worst scenario is $\delta = 12000$ as can be seen in the red line of Figure 3. However, this configuration does not stand for 2 millions objects where the best scenario for δ is $\delta = 1000$ and the worst scenario is $\delta = 43500$ as can be seen in the blue line of Figure 4.

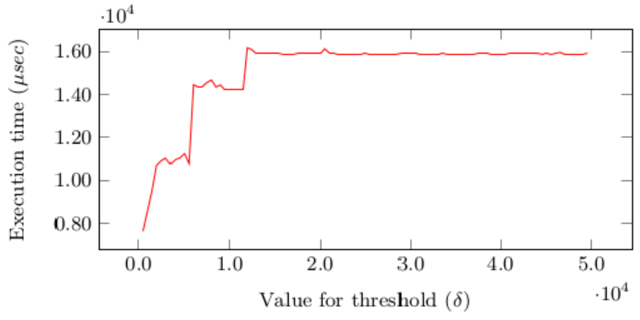

Fig. 3: Time consumption for different threshold (δ) using 100 thousand points (red)

A. *Performance for best scenario*

The objective of the following test is to measure the time and spatial consumption for best scenario of defined recursion thresholds. Figure 5 shows the time consumption for threshold (β) red, threshold (δ) blue and threshold (ϵ) black for each volume of data.

As can be seen in Figure 5 and 6 threshold ϵ performs better than other indexation criteria for each data volume tested in the best possible scenario. Considering threshold β threshold ϵ decreases the time consumption in 1.6×10^4 microseconds

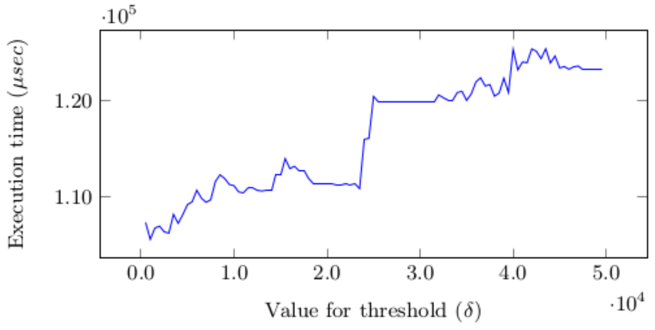

Fig. 4: Time consumption for different threshold (δ) using 2 millions points (blue).

as average and 8.7×10^3 microseconds considering threshold δ . Figure 5 also shows the number of generated childs for each indexation criteria for different data volumes in the best scenario. Results indicate that the use of the new indexation criteria in the best scenario needs more memory space in RAM than others recursion thresholds.

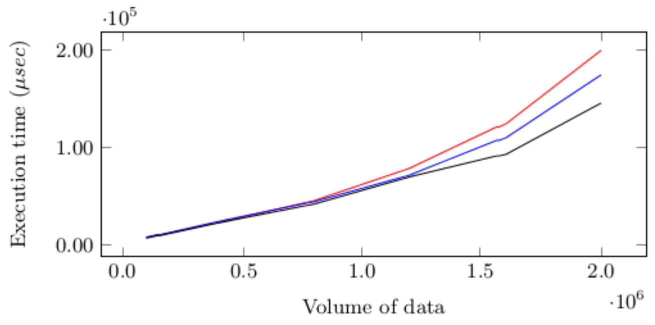

Fig. 5: Time consumption and children generation for different data volume using threshold (β) red, threshold (δ) blue and threshold (ϵ) black.

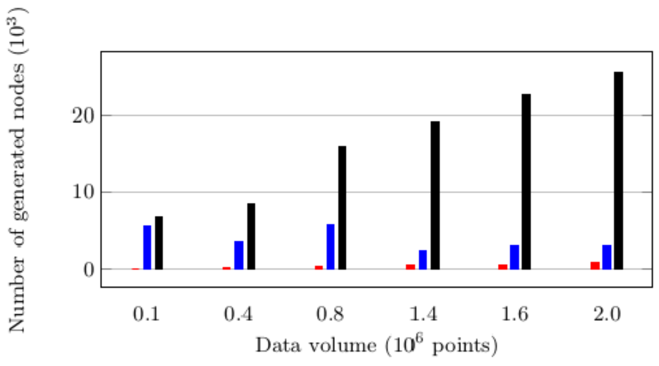

Fig. 6: Another view of time consumption and children generation for different data volume using threshold (β) red, threshold (δ) blue and threshold (ϵ) black.

ISSN 2395-8618

B. Performance for worst scenario

The objective of the following test is to measure the time and spatial consumption for worst scenario of defined recursion thresholds. Figure 7 shows the time consumption for threshold (β) red, threshold (δ) blue and threshold (ϵ) black for each volume of data.

As can be seen in Figure 7 threshold ϵ performs better than other indexation criteria for each data volume tested in the worst possible scenario. Considering threshold β threshold ϵ decreases the time consumption in 2.99×10^4 microseconds as average and 18.49×10^3 microseconds considering threshold δ.

In Figure 7 and 8 also shows the number of generated nodes for each indexation criteria for different data volumes in worst scenario. Results indicate that the use of the new indexation criteria in the worst scenario also needs more memory space in RAM than others recursion thresholds.

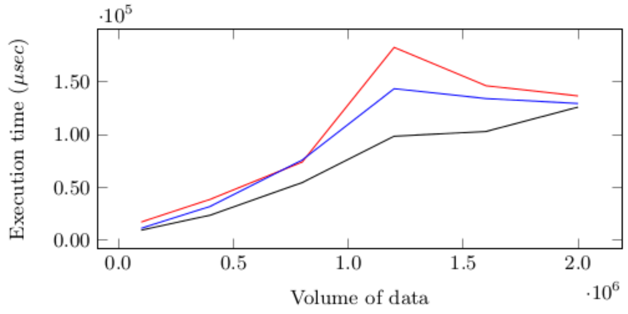

Fig. 7: Time consumption and child generation for different data volume using threshold (β) red, threshold (δ) blue and threshold (ϵ) black.

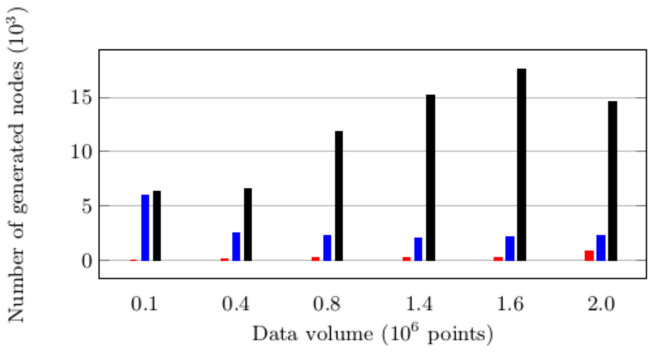

Fig. 8: Another view of time consumption and child generation for different data volume using threshold (β) red, threshold (δ) blue and threshold (ϵ) black.

V. Conclusion and future work

In this paper a new recursion threshold is proposed for indexing geometric objects using hierarchical spatial data-structures. The main contribution is a novel heuristic based threshold that performs better than other indexation criteria proposed for general data. Experiments show that considering threshold β threshold ϵ decreases the time consumption in 1.6×10^4 microseconds as average and 8.7×10^3 microseconds considering threshold δ when configuring parameters to the best possible scenario. The results remain positive even when the worst scenario is considered: considering threshold β threshold ϵ decreases the time consumption in 2.99×10^4 microseconds as average and 18.49×10^3 microseconds considering threshold δ. However the proposed threshold tends to generate more nodes in the hierarchical structure.

As future work, an investigation of the ϕ factor is required in different scenarios with the objective of reducing the space complexity of the proposed indexation criteria and more experiments have to be carried out in order to define new scenarios. We also pretend to investigate the effect of the re-indexation when considering our proposed method.

Acknowledgment

We thank colleagues Signal Processing and Geoinformation Research Group. This work was partially supported by, the University Cooperation Network "*Strengthening of ICT role in Cuban Universities for development of the society*", funded by VLIR-UOS.

References

[1] C. Quintana, "Algoritmo para la creación de volúmenes de restricción para búsqueda mínima en modelos de bloques geológicos," Centro Geoinformática y Señales Digitales, Universidad de las Ciencias Informáticas, Tech. Rep., 2013.

[2] J. Chen, "Computational geometry: methods and applications," *Computer Science Department Texas University*, 1996.

[3] Y. Valle Martínez and J. Rojas Ortiz, "Sistemas de información. representación de superficies de terrenos para su visualización en tres dimensiones," *Ciencias de la Información*, vol. 42, no. 3, pp. 57–64, 2011.

[4] D. P. Mehta and S. Sahni, *Handbook of data structures and applications*. CRC Press, 2004.

[5] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-time rendering*. CRC Press, 2008.

[6] C. M. Hoffmann, *Geometric and solid modeling*. Morgan Kaufmann, 1989.

[7] J. Bai, X. Zhao, and J. Chen, "Indexing of the discrete global grid using linear quadtree," *Proceedings of the XXXVIth ISPRS*, pp. 267–270, 2005.

[8] B. F. Naylor, "Binary space partitioning trees," *Handbook of Data Structures and Applications*, pp. 20–1, 2005.

[9] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time kd-tree construction on graphics hardware," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 5, p. 126, 2008.

[10] S. F. Frisken and R. N. Perry, "Simple and efficient traversal methods for quadtrees and octrees," *Journal of Graphics Tools*, vol. 7, no. 3, pp. 1–11, 2002.

[11] H. Sundar, R. S. Sampath, and G. Biros, "Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel," *SIAM Journal on Scientific Computing*, vol. 30, no. 5, pp. 2675–2708, 2008.

[12] T. Lewiner, V. Mello, A. Peixoto, S. Pesco, and H. Lopes, "Fast generation of pointerless octree duals," in *Computer Graphics Forum*, vol. 29, no. 5. Wiley Online Library, 2010, pp. 1661–1669.

[13] B. Aronov, H. Bronnimann, A. Y. Chang, and Y.-J. Chiang, "Cost-driven octree construction schemes: an experimental study," in *Proceedings of the nineteenth annual symposium on Computational geometry*. ACM, 2003, pp. 227–236.

[14] C. L. Jackins and S. L. Tanimoto, "Octrees and their use in representing three-dimensional objects," *Computer Graphics and Image Processing*, vol. 14, pp. 249–270, 1980.

[15] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

ISSN 2395-8618

[16] K. Yamaguchi, T. Kunii, K. Fujimura, and H. Toriya, "Octree-related data structures and algorithms," *IEEE Computer Graphics and Applications*, vol. 4, no. 1, pp. 53–59, 1984.

[17] T. Boubekeur, W. Heidrich, X. Granier, and C. Schlick, "Volume-surface trees," in *Computer Graphics Forum*, vol. 25, no. 3. Wiley Online Library, 2006, pp. 399–406.

[18] W. Saftly, P. Camps, M. Baes, K. Gordon, S. Vandewoude, A. Rahimi, and M. Stalevski, "Using hierarchical octrees in monte carlo radiative transfer simulations," *Astronomy & Astrophysics*, vol. 554, p. A10, 2013.

[19] W. Saftly, M. Baes, and P. Camps, "Hierarchical octree and kd tree grids for 3d radiative transfer simulations," *Astronomy & Astrophysics*, vol. 561, p. A77, 2014.

[20] P. K. Gupta, "Using neural networks for octree generation," *Journal of Multi Disciplinary Engineering Technologies Volume*, vol. 8, no. 1, p. 28, 2014.

[21] N. M. Josuttis, *The C++ standard library: a tutorial and reference*. Addison-Wesley, 2012.