Crisp Clustering Algorithm for 3D Geospatial Vector Data Quantization

Suhaibah Azri, François Anton, Uznir Ujang, Darka Mioc and Alias A. Rahman

Abstract In the next few years, 3D data is expected to be an intrinsic part of geospatial data. However, issues on 3D spatial data management are still in the research stage. One of the issues is performance deterioration during 3D data retrieval. Thus, a practical 3D index structure is required for efficient data constellation. Due to its reputation and simplicity, R-Tree has been received increasing attention for 3D geospatial database management. However, the transition of its structure from 2D to 3D had caused a serious overlapping among nodes. Overlapping nodes also occur during splitting operation of the overflown node N of $M + 1$ entry. Splitting operation is the most critical process of 3D R-Tree. The produced tree should satisfy the condition of minimal overlap and minimal volume coverage in addition with preserving a minimal tree height. Based on these concerns, in this paper, we proposed a crisp clustering algorithm for the construction of a 3D R-Tree. Several datasets are tested using the proposed method and the percentage of the overlapping parallelepipeds and volume coverage are computed and compared with the original R-Tree and other practical approaches. The experiments demonstrated in this research substantiated that the proposed crisp clustering is capable to preserve minimal overlap, coverage and tree height, which is advantageous for 3D geospatial data implementations. Another advantage of this approach

S. Azri (✉) · U. Ujang · A.A. Rahman
3D GIS Research Group, Department of Geoinformation, Faculty of Geoinformation and Real Estate, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia
e-mail: norsuhaibah@gmail.com

U. Ujang
e-mail: mduznir@utm.my

A.A. Rahman
e-mail: alias@utm.my

F. Anton · D. Mioc
Department of Geodesy, National Space Institute, Technical University of Denmark, Elektrovej 328, 2800 Kongens Lyngby, Denmark
e-mail: fa@space.dtu.dk

D. Mioc
e-mail: mioc@space.dtu.dk

M. Breunig et al. (eds.), *3D Geoinformation Science*,
Lecture Notes in Geoinformation and Cartography,
DOI 10.1007/978-3-319-12181-9_5

is that the properties of this crisp clustering algorithm are analogous to the original R-Tree splitting procedure, which makes the implementation of this approach straightforward.

Keywords 3D spatial data management · 3D spatial data clustering · 3D Geo-DBMS · 3D spatial indexing

1 Introduction

The uses of three-dimensional (3D) data input, 3D analysis and 3D visualization in geospatial applications, that are designed for managing and organizing geographic data (i.e., such as surface and subsurface objects), have recently become more widespread. This is due to their capability in handling efficiently 3D spatial information with non-spatial attributes. At the same time, various spatial problems could be solved efficiently (such as heat and sunlight spreading, calculation of wave propagation and noise spreading prediction models) in 3D rather than in 2D. 3D applications could be seen in various disciplines such as navigation, city planning and management, natural risks prevention, physical modeling and crime analysis.

The trend of 3D geospatial data is influenced by two main factors: users demand and data acquisition technology development such as Light Detection and Ranging (LiDAR), Interferometric Synthetic Aperture Radar (InSAR), and Unmanned Aerial Vehicles (UAV). Users demand on 3D geospatial application is due to the realistic view of environment which allows users to easily comprehend the real world problems straightforwardly (Izham et al. 2010; Jazayeri 2012; Uznir et al. 2013b). With the aid of data acquisition technologies, acquiring 3D data is getting much more convenient for 3D geospatial applications. Users could have various choices of data formats and resolutions depending on the tools and techniques used. On the other hand, the size of these 3D datasets is commonly very large due to the geometric details and in certain cases, it is incorporated with semantic datasets. For instance, by using laser scanning devices, millions of 3D points are captured for a single building or object. The same issue could be seen in 3D data interpretation applications such as CityGML. In CityGML environments, hundreds or thousands of coordinates and information are coded in XML format and stored for each urban object. In addition, in Building Information Modeling (BIM) applications, detailed building geometries and their descriptions are stored for each building model and at the same time, it escalates the data storage size.

Due to the large sizes of detailed information within 3D geospatial data, an efficient approach of 3D data management is needed. The management of 3D geospatial data is important with the aim to ensure the 3D data is efficiently managed and the produced information is efficiently retrieved. Due to these challenges, experts and spatial professionals are still working with the management of 3D geospatial information for geospatial applications. Therefore, in this paper, a method for improving the management of 3D geospatial information is presented. This method is useful for improving the efficiency of spatial access methods in order to manage clustering and indexing of 3D spatial datasets and reduce processing time during data retrieval process.

This paper is organized as follows. Problems and motivations regarding 3D geospatial information management are reviewed in the next section. In Sect. 3, the concept of the proposed indexing method is explained with its implementation. In Sect. 4, we present the analysis and results for the experiment conducted in this paper. Finally, in Sect. 5, we present the conclusions of this research.

2 Research Problem and Motivation

The increasing number of 3D geospatial applications could be seen in various fields and areas such as urban planning and management, natural risks modeling and prevention, cadaster and land administration, terrorism modeling and prevention. In the next few years, it is expected that 3D will be an intrinsic part of the core geospatial data infrastructure rather than a distinctive add-on as it is now. However, there are several issues that need to be resolved before utilizing 3D as a core geospatial data infrastructure. One of the challenges is the optimal management of 3D geospatial data, with optimization functions that combine several factors such as disk space consumption, data model, data structure and spatial access method. These factors are related to one another in order to achieve and optimize the efficiency of data management. However, 3D data come with a costly storage size, due to detailed geometrical features and detailed non-spatial descriptive information as discussed in the previous section. For an example the size of 3D geospatial data produced from laser scanning techniques, an interactive editing of large point clouds for a number of synthetic and real-world examples would take up to 63 GB (Wand et al. 2007). Another example of storage size for 3D geospatial data could be seen in (Uznir et al. 2013a). From the test in (Uznir et al. 2013a), 10,084 kb of disk storage is required for 2,500 buildings in CityGML.

Information carried and attached with 3D data varies and is amalgamated with other useful information such as the object's geometry (e.g., line, point and polygon) and other semantically based information such as roof, wall, color, object's id, material, etc. These geospatial data and information are very complex to handle in spatial database management systems. Thus, an enhanced spatial data model with spatial indexing methods is employed in order to manage the complexities and the large quantities of geospatial information. Based on this data model, 3D data are interpreted and transformed into a set of records or spreadsheets. To boost up the added value of these records, a 3D geospatial database is utilized for an efficient data constellation. In this platform, records will be stored, analyzed and manipulated efficiently. However, the efficiency of this platform towards a massive 3D dataset is slightly affected due to data complexity and disk storage consumption. In conjunction to this issue, the performance of data retrieval is also affected and decreased. The time of data retrieval is important in most critical applications, especially for real time applications or service based applications such as emergency response centers which require the retrieval of precise information at the right time in the fastest possible way.

In order to boost up the efficiency of data retrieval, a spatial index structure is utilized together with query operators. By using an index structure, the search area is minimized within a certain range without having a thorough inspection for the whole records. By implementing this approach, processing time is reduced as well as information retrieval time. For real time applications, spatial index structures are best to be utilized with data visualization and data updating (Liu et al. 2010; Zhu et al. 2007). However, in most of commercial spatial database management systems, supported index structures are mainly 2D. 2D index structures are not the best fit solution to be used for 3D geospatial information since the data types and relationships between objects are defined differently than in 2D. Until now, a well-established index structure for 3D spatial information is still an open research problem. Thus, a dedicated index structure for 3D geospatial information is significant for efficient data analysis and data retrieval.

There are various types of spatial indices invented for 2D application such as *k*-d-tree and Quad-tree. Among these structures, the R-Tree structure reported in Guttman (1984) is the most widely used index in spatial databases. This could be seen from the increasing number of R-Tree variants since it was invented in 1984 in Manolopoulos et al. (2006, Balasubramanian and Sugumaran (2012). Due to the reputation and the simplicity of its structure, many researchers (Zlatanova 2000; Arens et al. 2005; Zhu et al. 2007; Wang et al. 2010) agreed that the transition of R-Tree from 2D to 3D would be a starting point towards a promising 3D spatial index structure.

The implementation of the 3D R-Tree at geospatial database level was initiated by several commercial database management system producers, such as Oracle Spatial. Since then, several issues have been addressed due to the dimensionality expansion from 2D to 3D. In general, the features and properties of R-Trees would be inherited by 3D R-Trees, since the only modification is its dimensionality expansion. However, the dimensionality expansion from 2D to 3D did not address all the issues on overlapping nodes and splitting operations. According to Zhu et al. (2007), the parallelepipeds in 3D R-Trees will frequently overlap one another and the parallelepipeds of nodes could even contain one another. In R-Trees, overlapping among nodes should be minimized to avoid redundancy or replicated entries in a different node.

Another issue of R-Tree construction is a node splitting operation for overflown node N with $M + 1$ entries. According to Fu and Teng (2002), Sleit (2008), Liu et al. (2009), Korotkov (2012), the splitting operation is the most critical process of R-Trees, since tree structure is altered in this phase. Each overflowing node N needs to be split into two nodes and at the same time, it should produce minimal tree height, reducing overlapping area among rectangles and minimal coverage area of the node. These issues become critical when it comes to 3D. The minimization of overlap coverage of parallelepipeds is more complex and the splitting operation requires a different approach than in 2D.

The classical R-Tree structure (Guttman 1984) proposed three different approaches of splitting method i.e. quadratic, linear and exhaustive. However, these approaches have been optimized from time to time by scientists and researchers in order to produce minimum non-overlapping nodes (Paar 2006; Zhu et al. 2007; Sugihara and Shen 2012; Xu and Coors 2012).

3 Crisp Clustering Algorithm for 3D R-Tree

3.1 Splitting Algorithm for R-Tree Construction

The splitting operation is the most crucial process during the R-Tree construction. At this phase, the tree structure is altered either at the child or parent level or both. Splitting happens when the node N holds more than M entries, where M is the maximum number of entries for each node. Each node N is only allowed to hold $n \leq M$ entries, where n is number of entries in N. As shown in Fig. 1, the parallelepiped P denotes the overflown node N that requires a splitting operation with the maximum number of entries $M = 4$.

The original splitting approach in Guttman (1984) suggests to apply either exhaustive, quadratic or linear splitting algorithm. The exhaustive approach splits the overflown node by exhaustively visiting all the possible groupings. Then, the best group is chosen with respect to the smallest coverage area. However, based on the reviews from Manolopoulos et al. (2006), Al-Badarneh et al. (2013), Sleit and Al-Nsour (2014), the exhaustive approach is not practical due to the exponential time required.

Meanwhile for the quadratic approach, two entries will be selected to initiate two new nodes. Then, the remaining entries are selected one by one into the new nodes. The entry is added to the node whose coverage is minimized by the addition to the

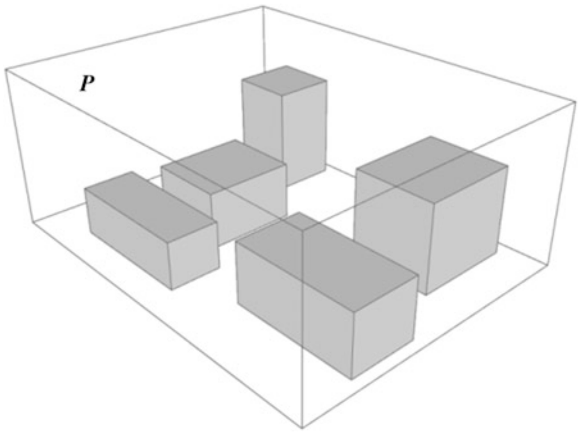


Fig. 1 Parallelepiped P with an overflown node N

new nodes. All remaining entries will go through this process until one of the new nodes reached the maximum number of entries ($M - m + 1$). According to Guttman, the quadratic approach makes compromises to achieve reasonable retrieval performance. However it does not guarantee to find the minimal or smallest coverage area compared with exhaustive approach.

The linear split algorithm has a linear time complexity. Two objects that are separated farthest apart are chosen as new nodes. Then remaining objects will be assigned one by one to the nearest node. All of these suggested methods are proven to work correctly for the construction of a R-Tree structure. However, since the main concern of R-Trees is to have very minimal areas of overlapping nodes, several researches have been conducted to improve the original methods. For example, the new linear algorithm in Ang and Tan (1997) is the improved version of Guttman's linear algorithm. In order to minimize overlap between nodes, this approach calculated the distributions of objects along all main axes to produce farthest distance towards the boundary of the overflown node. Another approach of splitting method is proposed in Kamel and Faloutsos (1994). In their approach (Kamel and Faloutsos 1994), they use the technique of space filling curves for nodes ordering and grouping all the nodes to minimize the area and perimeter of the minimum bounding rectangle.

3.2 Crisp Clustering Partition Based Algorithm for 3D R-Tree

Splitting and clustering are two different topics. However, the possibility of merging these two different topics within the construction of 3D R-Trees seems possible and obvious. The main idea of this paper is to cluster 3D geospatial objects based on clustering classes. Objects are then split according to their classes. One of the possible clustering methods to be utilized in this case is the crisp clustering algorithm. The crisp clustering algorithm has been used ubiquitously from web mining, spatial data analysis, business, prediction based on groups and many more. This approach considers non-overlapping partitions and each data vector either belongs to a class or not. This characteristic is suitable with the 3D R-Tree condition which is that an object will be clustered and appear only once in an index node so that the index nodes and their total number will be minimized. In the construction of a 3D R-Tree, replicated objects in several nodes will cause a larger structure than necessary.

In this paper, a partition based algorithm for crisp clustering will be utilized to construct the structure of 3D R-Tree. *k*-means (MacQueen 1967), is one of the most popular clustering algorithm used to solve various fundamental problems in data mining, data analysis and machine learning. The *k*-means algorithm is a simple and fast clustering algorithm although it offers no approximation guarantee at all. The *k*-means algorithm for a set of 3D vector data is described as follows:

Input	A set of 3D vector data $P = \{p_1, p_2, \ldots, p_n\} \in \mathbb{R}^d$
Step 1	Initial k cluster center $C = \{c_1, c_2, \ldots, c_k\} \in \mathbb{R}^d$ is randomly picked
Step 2	For each $i \in \{1, \ldots, k\}$, the cluster C_i is set to be the set of points that are closer to C_i than they are to C_j for all $j \neq i$
Step 3	For each $i \in \{1, \ldots, k\}$, c_i is set to be the center of mass of all points in C_i $C_i = \frac{1}{\lvert c_i \rvert} \sum_{p \in C_i} P$
Step 4	Steps 2 and 3 are repeated until C no longer changes

The main objective of choosing k centers C is to minimize the potential function:

$$\phi = \sum_{p \in P} \min_{c \in C} [d(p, C)]^2 \tag{2}$$

However, finding cluster center C to minimize the potential function is Non-deterministic Polynomial-time (NP) hard problem. This is due to the usual iterative method of Lloyd's algorithm in this clustering which is unlikely to get the minimum error for the cluster centers. In general, this may work but there is no theoretical guarantee of getting close to the minimum. As a solution, random values are chosen in this clustering and logically, this approach may work but without assurance of minimizing the error. Another problem with the random initialization is the risk of having two centers c_i and c_j in the same cluster $C_i = C_j$, that will produce over-lapping parallelepipeds. The description of this issue could be seen in Fig. 2. From Fig. 2, three cluster centers C_1, C_2 and C_3 are randomly picked. Based on the result, we could see that the cluster centers C_1 and C_2 are actually appointed from the same cluster.

Thus, with the new edition of this algorithm the issue of random picked of cluster center initialization is solved by spreading out the centers. In this approach,

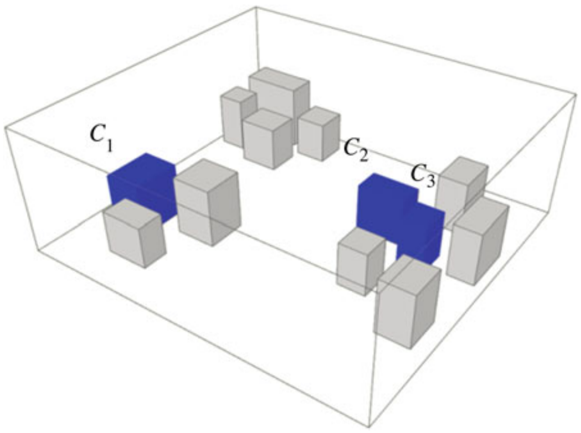


Fig. 2 Random initialization of *k*-means crisp clustering

one center C_1 is chosen uniformly from P. Then, a new center C_i is defined based on the probability of

$$\frac{D(x)^2}{\sum_{x \in X} D(x)^2} \tag{3}$$

The new edition of *k*-means algorithm is known as *k*-means++ in Arthur and Vassilvitskii (2007). *k*-means++ improved the original algorithm by defining the initial cluster center. Initial seeds are defined based on the farthest apart. Then, remaining data are clustered based on the nearest distance to the initial seeds. The initial seeding in *k*-means++ is proven to yield the improvement from its original algorithm. The algorithm of 3D *k*-means++ is described as follows.

Input	a set of 3D vector data $P = \{p_1, p_2, \ldots, p_n\} \in \mathbb{R}^d$
Step 1	Choose initial center C_1
Step 2	Choose a new center C_i, by choosing $p \in P$ with probability $$\frac{D(p)^2}{\sum_{p \in P} D(p)^2} \tag{4}$$
Step 3	*Step 2* is continued until k centers $C_1, \ldots, C_k$ are chosen
Step 4	Proceed with the standard *k*-means approach

The same dataset as depicted in Fig. 2 has been tested using the new edition of *k*-means algorithm. From the result, the cluster centers C_i are uniformly spread (see Fig. 3) and the probability of having two or more center in one cluster is reduced. Based on the results from Figs. 2 and 3, the parallelepipeds produced for each method of 3D R-Tree are described in Table 1. Based on these results, 3D *k*-means++ produced a better 3D R-Tree structure with no overlapping parallelepiped.

Based on its simplicity and lower risk of choosing center from the same cluster, we adopted 3D *k*-means++ crisp clustering to be utilized in 3D R-Tree construction

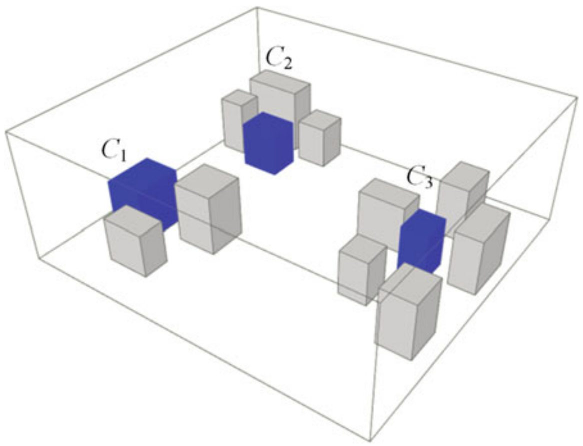
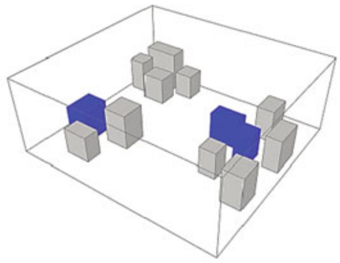
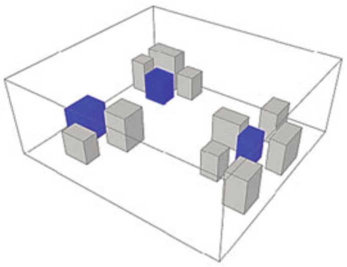
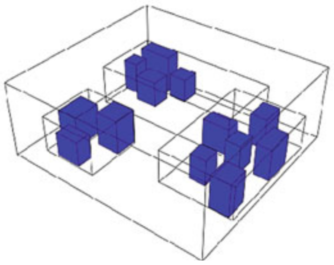
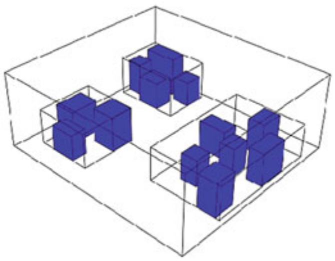


Fig. 3 Uniformly spread cluster centers C_i

Table 1 Produced structure of 3D R-tree using different approaches

	Current Approach	Proposed 3D Crisp Clustering
Clustering Results		
3D R-Tree	 Number of produced node = 3 Number of overlapping node = 2	 Number of produced node = 3 Number of overlapping node = 0

and splitting operation. The workflow of 3D R-Tree construction based on crisp clustering approaches is illustrated in Fig. 4. The test and results of this implementation are discussed in the following section.

4 Analysis and Results

To find the potential and efficiency of this algorithm for 3D R-Tree construction, a set of vector data needs to be tested and quantized. Datasets used in this research are a set of 3D building blocks with Level of Detail 2 (LoD 2) of CityGML definition. By following CityGML the standard, these 3D building blocks are then clustered using 3D *k*-means++ crisp clustering algorithm. The cluster classes are defined based on the maximum number of entries *M*. Clustered blocks are then grouped into parallelepipeds and each parallelepiped with the overflown node *N* is qualified for the next cycle. In our experiment, the maximum number of 40 building blocks ($n = 40$) is demonstrated using the proposed crisp clustering algorithm. As an initial

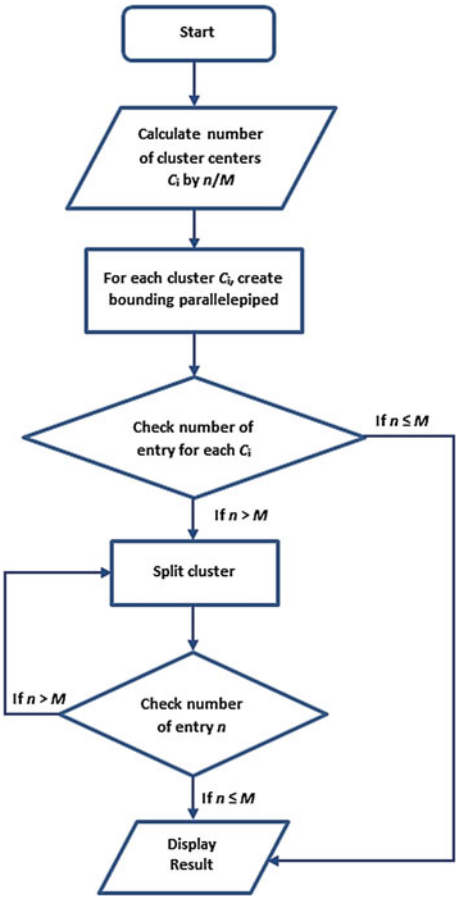

Start
Calculate number of cluster centers C_i by n/M
For each cluster C_i, create bounding parallelepiped
Check number of entry for each C_i
If $n \leq M$
If $n > M$
Split cluster
If $n > M$
Check number of entry n
If $n \leq M$
Display Result

Fig. 4 Workflow of 3D R-Tree construction based on crisp clustering

procedure, the maximum number of entries is initialized with $M = 6$. Later, n building blocks are clustered based on the value of M, that is to say, that each clustered group could only hold at most M objects. In this experiment, blocks will be grouped into six clusters.

As a result, building blocks are categorized into six groups of 3D R-Tree parallelepipeds named P, Q, R, S, T and U (see Fig. 5) based on their geographic location or position as exemplified in Fig. 5. However, among these groups there are two parallelepipeds (P and T) holding more than M entries. Thus, nodes P and T are entitled for the next cycle. In the second cycle, each node will be split again using the crisp clustering algorithm and divided into two sub-groups of P (P1 and P2) and T (T1 and T2). Overall results of this dataset are explained in Fig. 6.

Another experiment conducted in this research is to test the efficiency of the proposed crisp clustering method in reducing the number of overlapping nodes. For this experiment, a set of 3D vector data is indexed using the 3D R-Tree structure. Objects are indexed using three different approaches which are the original R-Tree,

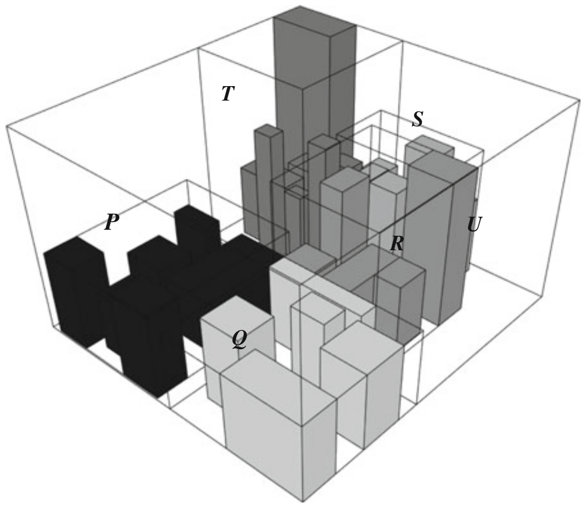


Fig. 5 3D R-Tree based on crisp clustering

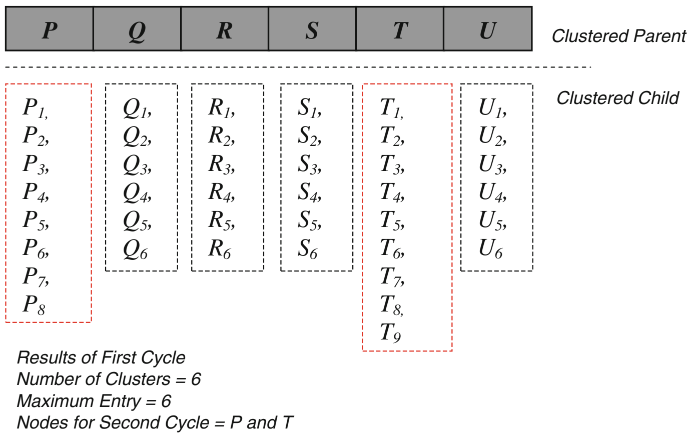


Fig. 6 Clustered objects in 3D space

the new linear splitting by Ang/Tan and our new 3D *k*-means++ crisp clustering. Then, each result is compared based on total overlapping percentage for all nodes. Figure 7 shows the construction of 3D R-Tree using different approaches. Solid parallelepiped in Fig. 6 represents the overlapping nodes while wireframe parallelepipeds are non-overlapping nodes. From the experiment, the percentage of total overlapping volume for original R-Tree is 35 %, Ang/Tan 29 % and crisp clustering is only 3 %.

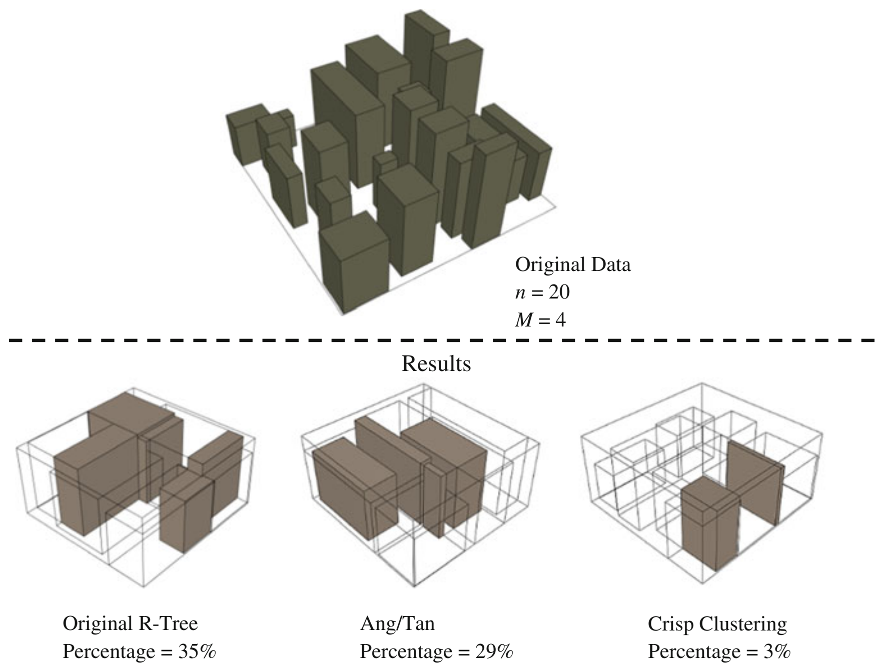


Fig. 7 Percentage of overlapping nodes using different approaches

Another important benchmark in constructing the 3D R-Tree is the coverage volume of parallelepipeds. Several groups of 3D building blocks have been used in this experiment. Each group contain different number of building blocks which is A = 200 blocks, B = 500 blocks, C = 1,000 blocks, D = 10,000 blocks and E = 30,000 blocks. Based on the experiment, crisp clustering offers lower volume coverage than other two techniques. The result of this experiment is described in the following Fig. 8.

In order to verify the efficiency of crisp clustering towards the size of 3D R-Tree structure, the tree height and number of parallelepiped produced is measured and analyzed. A set of 3D building blocks with $n = 30{,}000$ and $M = 25$ is indexed using 3D R-Tree with a different approaches. Height of the tree is measured and total number of node is calculated. The height and number of produced node should be low in order to minimize the traversal exploration from the root node to the leaf node. Table 2 presents a comparison of height and total number of nodes using different approaches.

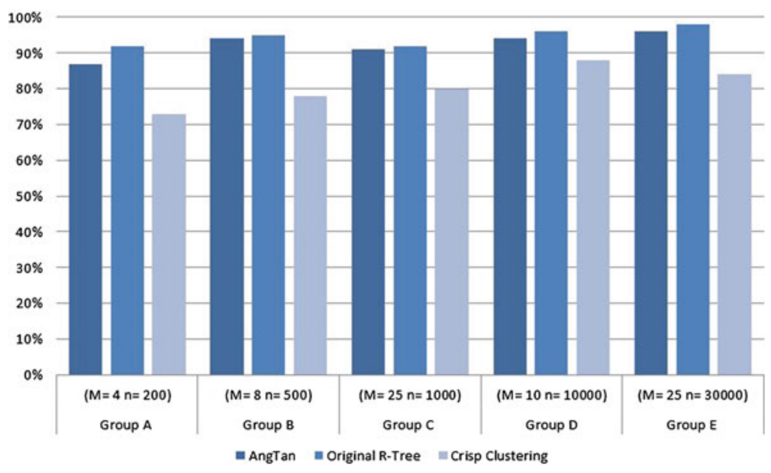


Fig. 8 Comparison of volume coverage percentage based on several approaches

Table 2 Produced number of nodes using different approaches

Method	Height	Number of node
Ang/tan	4	1,222
Linear	4	1,226
Exhaustive	4	1,230
Crisp clustering	4	1,200

5 Conclusions

For a massive 3D geospatial data application, retrieving information could be a crucial task and issue in data management. Practically, by implementing spatial indexing with the query operation, information will be organized sufficiently and the information will be retrieved efficiently. However, in some cases of spatial indexing structure such as 3D R-Tree, several factor need to be fine-tune according to several factors such as overlapping node, coverage area and splitting operation.

In this paper, we have proposed 3D crisp clustering algorithm for the construction of 3D R-Tree. Due to the comprehensive consideration of overlapping parallelepipeds, volume coverage and splitting operation, the crisp clustering algorithm proposed in this paper demonstrated the practically in minimizing the overlapping nodes and volume coverage. Besides that, the produced number of nodes based on the 3D crisp clustering method is smaller than other methods. Furthermore, based on the test of coverage, overlapping and produce number of nodes in this paper, our proposed crisp clustering algorithm method offers a new potential and promising approach for 3D R-Tree construction. For future research, we plan to extend the experiment and analysis by proofing the proposed algorithm for performance testing comparison towards the existing approaches.

Acknowledgments Major funding for this research was supported by the Ministry of Education Malaysia.

References

Al-Badarneh AF, Al-Alaj AS, Mahafzah BA (2013) Multi small index (MSI): a spatial indexing structure. J Info Sci 39(5):643–660

Ang CH, Tan TC (1997) New linear node splitting algorithm for R-trees. In: Scholl M, Voisard A (eds) Advances in spatial databases. Lecture notes in computer science, vol 1262. Springer, Berlin, pp 337–349. doi:10.1007/3-540-63238-7_38

Arens C, Stoter J, van Oosterom P (2005) Modelling 3D spatial objects in a geo-DBMS using a 3D primitive. Comput Geosci 31(2):165–177. doi:http://dx.doi.org/10.1016/j.cageo.2004.05.013

Arthur D Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for industrial and applied mathematics, pp 1027–1035

Balasubramanian L, Sugumaran M (2012) A state-of-art in R-tree variants for spatial indexing. Int J Comput Appl 42(20)

Fu Y, Teng J-C (2002) Subramanya S node splitting algorithms in tree-structured high-dimensional indexes for similarity search. In: Proceedings of the 2002 ACM symposium on applied computing. ACM, pp 766–770

Guttman A (1984) R-trees: a dynamic index structure for spatial searching. SIGMOD Rec 14 (2):47–57. doi:10.1145/971697.602266

Izham MY, Uznir U, Alias AR, Ayob K (2010) Georeference, rainfall-runoff modeling and 3D dynamic simulation: physical influence, integration and approaches. In: ACM, 1st international conference and exhibition on computing for geospatial research and application, Washington, DC. 1st international conference and exhibition on computing for geospatial research and application, COM.Geo 2010

Jazayeri I (2012) Trends in 3D land information collection and management. In: Rajabifard A, Williamson I, Kalantari M (eds) A national infrastructure for managing land information. University of Melbourne, pp 81–87

Kamel I, Faloutsos C (1994) Hilbert R-tree: an improved R-tree using fractals. Paper presented at the proceedings of the 20th international conference on very large data bases

Korotkov A (2012) A new double sorting-based node splitting algorithm for R-tree. Program Comput Softw 38(3):109–118

Liu Y, Fang J, Han C (2009) A new R-tree node splitting algorithm using MBR partition policy. In: 17th international conference on geoinformatics, 2009. IEEE, pp 1–6

Liu Y, Liu G, He Z (2010) Spatial index technology for multi-scale and large scale spatial data. In: 18th international conference on geoinformatics, 2010. IEEE, pp 1–4

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, California, USA, p 14

Manolopoulos Y, Nanopoulos A, Papadopoulos AN, Theodoridis Y (2006) Rtrees: theory and applications. Springer, Heidelberg

Paar P (2006) Landscape visualizations: applications and requirements of 3D visualization software for environmental planning. Comput Environ Urban Syst 30(6):815–839. doi:http://dx.doi.org/10.1016/j.compenvurbsys.2005.07.002

Sleit A (2008) On using B++-tree for efficient processing for the boundary neighborhood problem. Paper presented at the WSEAS transactions on systems

Sleit A, Al-Nsour E (2014) Corner-based splitting: an improved node splitting algorithm for R-tree. J Info Sci. doi:10.1177/0165551513516709

Sugihara K, Shen Z (2012) Automatic generation of virtual 3D city models for urban planning. In: Geospatial techniques in urban planning. Advances in geographic information science. Springer, Berlin, pp 265-283. doi:10.1007/978-3-642-13559-0_13

Uznir U, Anton F, Suhaibah A, Rahman A, Mioc D (2013a) Improving 3D spatial queries search: newfangled technique of space filling curves in 3D city modeling. In: 8th 3D geoinfo conference and ISPRS WG II/2 workshop

Uznir U, François A, Alias AR (2013b) Unified data model of urban air pollution dispersion and 3D spatial city model: groundwork assessment towards sustainable urban development for Malaysia. J Environ Prot 4(7):701–712. doi:10.4236/jep.2013.47081

Wand M, Berner A, Bokeloh M, Fleck A, Hoffmann M, Jenke P, Maier B, Staneker D (2007) Schilling a interactive editing of large point clouds. In: SPBG, pp 37–45

Wang Y, Sheng Y, Zhou L, Guo F, Zhao L (2010) An underground space object-oriented three-dimensional hybrid spatial indexing method. In: 18th international conference on geoinformatics, 2010. IEEE, pp 1–5

Xu Z, Coors V (2012) Combining system dynamics model, GIS and 3D visualization in sustainability assessment of urban residential development. Build Environ 47(0):272–287. doi: http://dx.doi.org/10.1016/j.buildenv.2011.07.012

Zhu Q, Gong J, Zhang Y (2007) An efficient 3D R-tree spatial index method for virtual geographic environments. ISPRS J Photogram Remote Sens 62(3):217–224. doi:http://dx.doi.org/10.1016/j.isprsjprs.2007.05.007

Zlatanova S (2000) 3D GIS for urban development. International Institute for Aerospace Survey and Earth Sciences (ITC)