

Highly Efficient Computer Oriented Octree Data Structure and Neighbours Search in 3D GIS

Noraidah Keling, Izham Mohamad Yusoff, Habibah Lateh
and Uznir Ujang

Abstract Three-dimensional (3D) visualization has given a new perspective in various fields such as urban planning, hydrology, infrastructure modelling and geology. This is due to its capability of handling real world object in more realistic manners, rather than the two-dimensional (2D) approach. However, implementation of 3D spatial analysis in the real world situations has proven to be difficult to comprehend due to the complexity of the algorithm, computational process and time consuming. The existing Geographical Information Systems (GIS) enable 2D and two-and-a-half-dimensional (2.5D) spatial datasets, but less capable of supporting 3D data structures. Recent development in Octree showed that more effort was given to improve the weakness of Octree in finding neighbouring nodes by using various address encoding scheme with specific rule like matrix, lookup table and arithmetic to eliminate the need of tree traversal. Therefore, the purpose of this paper is to propose a new method to speed up the neighbouring search by eliminating the needs of complex operation to extract spatial information from Octree by preserving 3D spatial information directly from the Octree data structure. This new method will be able to achieve $O(1)$ complexity and utilizing Bit Manipulation Instruction 2 (BMI2) to speed up address encoding, extraction and voxel search 1000x compared to generic implementation.

1 Introduction

Recent developments in the field of three-dimensional (3D) spatial data representation have led to a renewed interest in Geographic Information Systems (GIS). Application of 3D technologies in GIS developments is one of the issues that has

N. Keling (✉) · I. Mohamad Yusoff · H. Lateh
School of Distance Education, Universiti Sains Malaysia, George Town, Malaysia
e-mail: noraidahkeling@gmail.com

U. Ujang
Faculty of Geoinformation and Real Estate, Universiti Teknologi Malaysia,
Johor Bahru, Malaysia

been given special attention in this platform and has come to the limelight among various geo-related professionals such as land practitioners and environmentalists (Berry et al. 2008; Goodchild 2009). Perspective in third dimension is really helpful in some fields such as urban planning and management (Ujang et al. 2014), geological sub surface modelling (Li et al. 1996), mining and oil exploration (Pouliot et al. 2008; Jin et al. 2011), hydrology (Izham et al. 2011), infrastructure modelling, and geology (Zhu et al. 2012, 2013).

Apart from visualization task, 3D has given a new perception in this field due to its ability to handle complex real-world phenomena in a more realistic manner for a better recognition of geographical and geological space and to master the inherent laws in earth-science and engineering application (Tianding 2010). Currently, most of the direct and indirect approaches to effectively utilize GIS is by using GIS as a spatial database for storing, displaying and updating the input data for site-specific measurement and experiment for a better understanding of the interaction of the conditioning factor (Rolf 2004).

However, the major problem with this kind of application is the existing commercial GIS software only support 2D and 2.5D spatial datasets which is only useful to visualize adjacent surfaces with variations of height such as terrain, but it is not suitable to represent real 3D data structures. So far, the 3D functionality of GIS software which is derived from the 2.5D data extrapolation is only being used as visualization (Abdul-Rahman and Pilouk 2008). Currently the usage of 2.5D for the 3D data representation is only useful as an approximation and it does not satisfy the current need anymore (Lixin 2004; Rogers and Luna 2004).

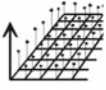
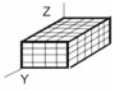





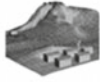

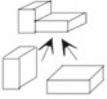






Therefore, the demand on a fully functional 3D GIS is increasing and became more and more important to the geological analysis and engineering application to understand geological phenomena and complex structure of any kind of information related with their 3D environment in a more realistic way and to carry out the complicated 3D spatial analysis (Wenzhong 2000).

There are various data representations that has been proposed for GIS and it can be categorized into three different models; surface model, volumetric model and hybrid model (Li et al. 1996; Lixin 2004). Table 1 summarizes some examples of the data representation and its classification (Wenzhong 2000a; Shen et al. 2003, 2006, 2013; Huixin and Huifeng 2006; Shen and Takara 2006; Abdul-Rahman and Pilouk 2008).

For 2D GIS, there are two types of data representations that are regularly used; Triangulated Irregular Network (TIN) and Grid. TINs is usually used by GIS software to store and display surface and boundaries due to the accuracy and relatively small database size. However, TIN is not suitable for some application, such as spatial analysis like slope stability, topographic wetness index and hydrology infiltration model. For those application, Grid data representation is usually used.

In 3D data representation, Tetrahedral Network (TEN) is actually equivalent to 3D-TIN while 3D Array is equivalent to 3D-Grid. However, 3D array is not widely

Table 1 Various data model representation

Surface Model	Volumetric Model		Hybrid Model
	Regular	Irregular	
 Grid	 3D array	 TEN	 TIN-Octree
 Shape	 Octree	 Pyramid	 TIN-CSG
 Facet	 CSG	 TP	 Octree-TEN
 B-Rep		 3D Voronoi	
 TIN		 Geocellular	

used in practical application due to the amount of memory needed to keep the data and its role is usually being replaced by the Octree. In this paper, we tried to focus on improving the Octree to be better and more practical in 3D spatial analysis by improving the neighbour search performance.

2 Methods

2.1 Related Research

Octree is a method where 3D space is being represented in a hierarchical tree structure as visualized in Fig. 1. Classical Octree is originated from Klinger (1971) as quadtree and subsequently expanded to cover 3D space as Octree. The earlier implementation of Octree used tree traversal as suggested by Samet (1989), Ballard and Brown (1982), Besançon and Faugeras (1988). Neighbours search can be done by backtracking upwards from selected node until the common ancestor is found and then travel down the tree until the neighbours are found. The advantage of this implementation is that the algorithm is simplified by exploiting recursive nature of octree, but the drawback is it require long tree traversals.

Recent research development in Octree see more effort to improve weakness of Octree in finding neighbour node which is required for spatial analysis by using various address encoding scheme with the specific rule to eliminate the need of tree traversal.

Vörös (2000) suggested that each leaf node is to be described by a unique locational code corresponding to a sequence of directional codes encoding the path from the root to that node as Fig. 2 and represented in matrix formed as shown in Fig. 3. The neighbour nodes are located by using matrix based approach.

Payeur (2006) suggested a generic set lookup table that determines the address offset from a given cell to its neighbour for each possible direction based on the observation of the addressing structure as shown in Fig. 13. 26 neighbours voxel is categorized into 3 different categories, which are face neighbour, edge neighbour and vertex neighbour. Each of these categories has its own lookup table as shown in Figs. 4, 5, and 6.

Kim and Lee (2009) suggested another method by relying on the combination of arithmetic algorithm as shown in Fig. 7 and also lookup table as in Fig. 8. The lookup table is derived based on the observation of his address scheme as shown in Fig. 14.

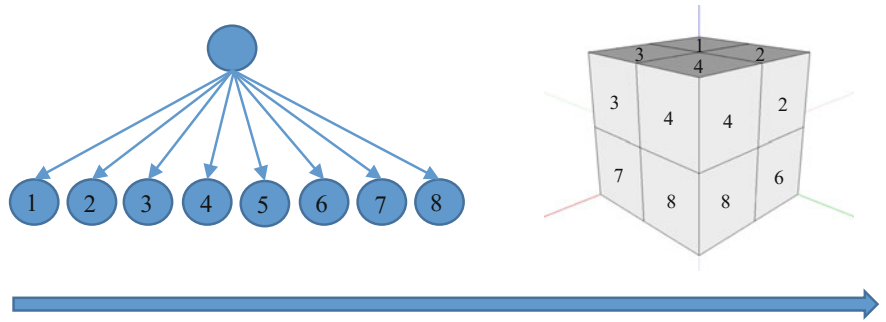


Fig. 1 Octree node to 3D space mapping

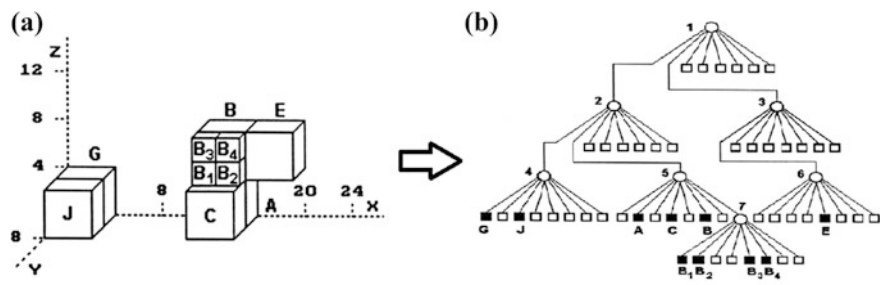
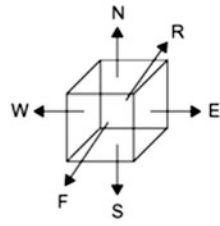


Fig. 2 Voxel/leaf locational code (Vörös 2000)

$$A = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

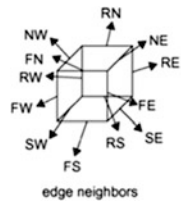
Fig. 3 Leaf matrix representation (Vörös 2000)



face neighbors

		Direction					
		N	S	E	W	F	R
Initial Digit Value	0	2	2 + S	1	1 + W	4	4 + R
	1	3	3 + S	0 + E	0	5	5 + R
	2	0 + N	0	3	3 + W	6	6 + R
	3	1 + N	1	2 + E	2	7	7 + R
	4	6	6 + S	5	5 + W	0 + F	0
	5	7	7 + S	4 + E	4	1 + F	1
	6	4 + N	4	7	7 + W	2 + F	2
	7	5 + N	5	6 + E	6	3 + F	3

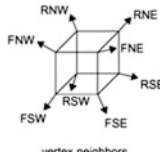
Fig. 4 Face neighbours definition and look-up table (Payeur 2006)



edge neighbors

		Direction											
		NW	NE	SW	SE	FN	RN	FS	RS	FE	FW	RE	RW
Initial Digit Value	0	3+W	3	3+SW	3+S	6	6+R	6+S	6+RS	5	5+W	5+R	5+RW
	1	2	2+E	2+S	2+SE	7	7+R	7+S	7+RS	4+E	4	4+RE	4+R
	2	1+NW	1+N	1+W	1	4+N	4+RN	4	4+R	7	7+W	7+R	7+RW
	3	0+N	0+NE	0	0+E	5+N	5+RN	5	5+R	6+E	6	6+RE	6+R
	4	7+W	7	7+SW	7+S	2+F	2	2+FS	2+S	1+F	1+FW	1	1+W
	5	6	6+E	6+S	6+SE	3+F	3	3+FS	3+S	0+FE	0+F	0+E	0
	6	5+NW	5+N	5+W	5	0+FN	0+N	0+F	0	3+F	3+FW	3	3+W
	7	4+N	4+NE	4	4+E	1+FN	1+N	1+F	1	2+FE	2+F	2+E	2

Fig. 5 Edge neighbours definition and look-up table (Payeur 2006)



		Direction							
		FNE	FNW	FSE	FSW	RNE	RNW	RSE	RSW
Initial Digit Value	0	7	7+W	7+S	7+SW	7+R	7+RW	7+RS	7+RSW
	1	6+E	6	6+SE	6+S	6+RE	6+R	6+RSE	6+RS
	2	5+N	5+NW	5	5+W	5+RN	5+RNW	5+R	5+RW
	3	4+NE	4+N	4+E	4	4+RNE	4+RN	4+RE	4+R
	4	3+F	3+FW	3+FS	3+FSW	3	3+W	3+S	3+SW
	5	2+FE	2+F	2+FSE	2+FS	2+E	2	2+SE	2+S
	6	1+FN	1+FNW	1+F	1+FW	1+N	1+NW	1	1+W
	7	0+FNE	0+FN	0+FE	0+F	0+NE	0+N	0+E	0

Fig. 6 Vertex neighbours definition and look-up table (Payeur 2006)

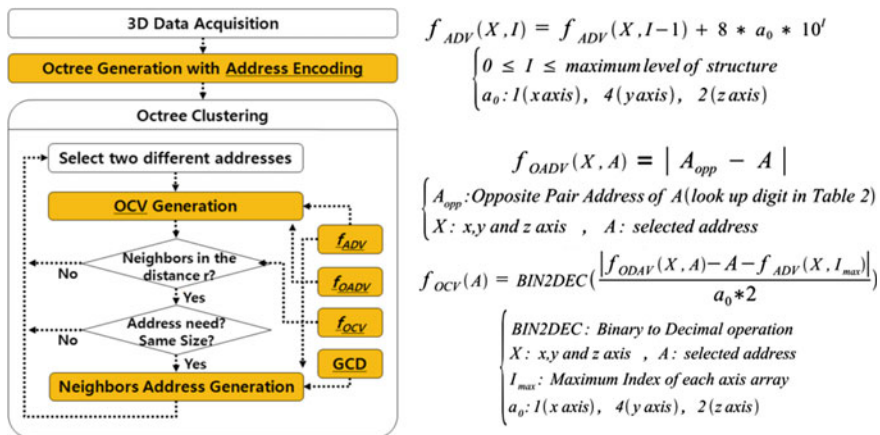


Fig. 7 Neighbour search flow and arithmetic involved (Kim and Lee 2009)

Fig. 8 Opposite pair address lookup table (Kim and Lee 2009)

Digit	X axis	Y axis	Z axis
1	2	5	3
2	1	6	4
3	4	7	1
4	3	8	2
5	6	1	7
6	5	2	8
7	8	3	5
8	7	4	6

2.2 Proposed Octree Structure

In this paper, the author is also proposing another address encoding scheme and data structure which bases on binary ordering which efficiently preserve 3D spatial information from Octree structure and carefully designed in such way to fully take

advantage of the advancement of computer CPU accelerator instruction to speed up neighbour search and eliminating the need of complex operation to derive voxel spatial information like suggested by Kim and Lee (2009), the matrix solution like Voros (2000) or lookup tables like Payeur (2006).

2.2.1 Binary Address Encoding

The basis of the proposed Octree encoding scheme is to assign each node of Octree a unique address in binary form with 3D spatial information embedded. Octree in 3D space is binary by nature. Voxel progression along with the axis can be represented by only 1 bit as shown in Fig. 9. For example, the voxel which is close to axis X, the address is set as 0 and for the voxel located further from the axis the address is set as 1. By incorporating this nature into address encoding, we can preserve all 8 voxels spatial data information by using only 3 bits.

These 3 bits can be arranged using $X_pY_pZ_p$ standard, where P is Octree level or depth. For example, for the voxel which located in coordinate $x = 0, y = 0, z = 0$, the voxel address is 0b000 (0d0) and voxel located at coordinate $x = 1, y = 1, z = 1$ then the voxel address will be 0b111 (0d7). The addressing for the voxel is straight forward and easy to identify voxel physical location just based on Octree address. The Fig. 10 shows the complete address for proposed Octree in binary form.

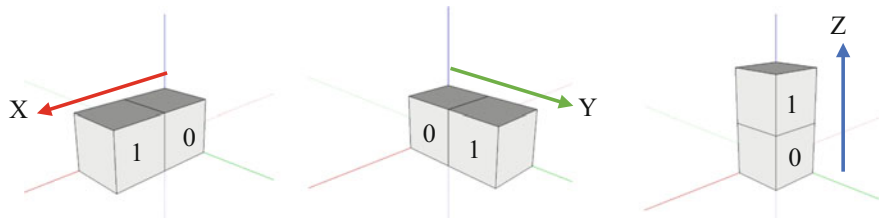
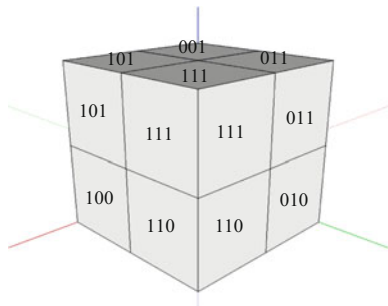


Fig. 9 Voxel numbering progression

Fig. 10 Proposed octree addressing scheme



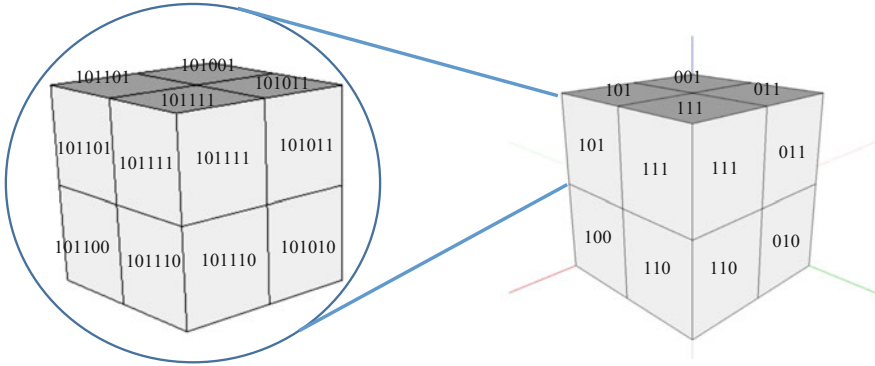


Fig. 11 Proposed octree subdivision addressing scheme (binary)

2.2.2 Subdivision Address Encoding

$$\text{Voxel Address} = (X_p Y_p Z_p) \cap (X_{p+1} Y_{p+1} Z_{p+1}) \cap \dots \cap (X_{p+n} Y_{p+n} Z_{p+n}) \quad (1)$$

The address encoding for subdivision Octree is similar with parent voxel except the parent address is directly concatenating into child voxel address as shown in (1). The main difference between proposed encoding with previous methods that was introduced by Voros (2000), Payeur (2006), Kim and Lee (2009) is this method strictly uses 3 bits binary encoding per level as shown in Figs. 11 and 12 while others are using decimal based as shown in Figs. 13 and 14. Decimal representation is only make sense for human readable for manual processing because parent address can be easily identified concatenated with child address, but to represent those number in decimal will require 25 % more space and without any advantage for computer processing.

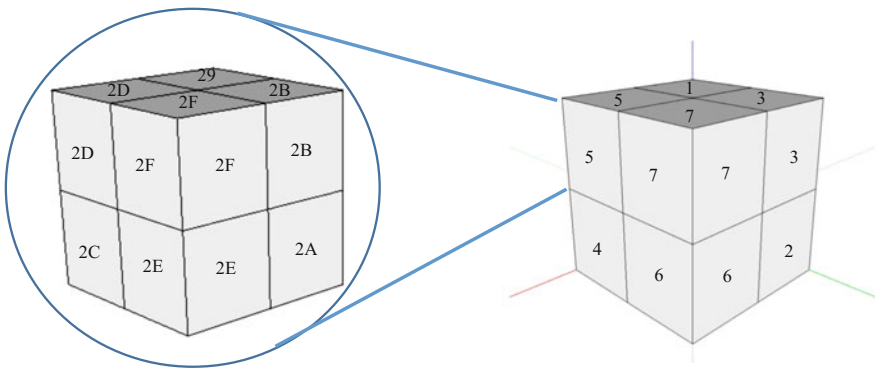


Fig. 12 Proposed octree subdivision addressing scheme (hexadecimal)

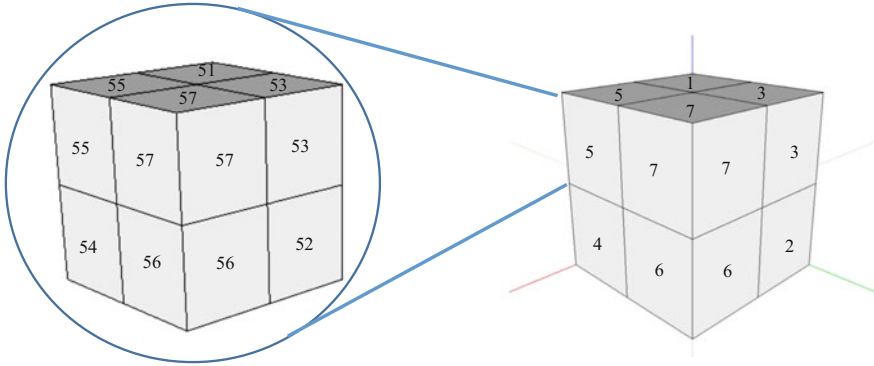


Fig. 13 Payeur and Voros octree subdivision addressing scheme

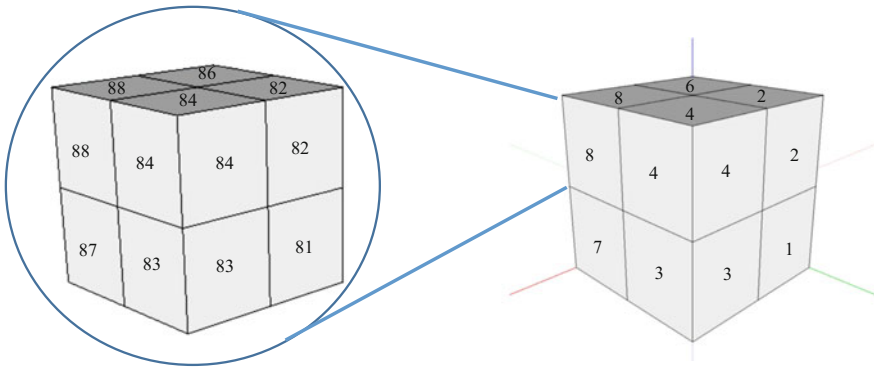


Fig. 14 Kim and Lee octree subdivision addressing scheme

2.3 Data Structure

In order to take advantage of modern computer (explained in Sect. 2.4), each voxel should be kept either in 32 bits or 64 bits. By using Eq. (2), 32 bits data structure are able to hold maximum of 10 levels of Octree and for 64 bits data structure will be able to hold up to 21 levels of Octree.

$$\text{Maximum Octree Maximum level } (P) = \left\lfloor \frac{\text{Data Structure Bits}}{3} \right\rfloor \quad (2)$$

By using Eq. (3), with the assumption of minimum resolution of surface area that we want to cover is 1 m², 32 bits data structure only able to cover up to 1,048,576 km² but 64 bits data structure is able to cover up to 4,398,046,511.104 km². With the total earth surface area is estimated only around 510,072,000 km², 64 bits data structure is practically more than enough to cover the

Fig. 15 Proposed data structure

MSB													LSB		
X ₁	Y ₁	Z ₁	X ₂	Y ₂	Z ₂	X ₃	Y ₃	Z ₃	...	X ₂₁	Y ₂₁	Z ₂₁	1		

Fig. 16 Proposed data structure with partially fill up bits

MSB																		LSB		
X ₁	Y ₁	Z ₁	X ₂	Y ₂	Z ₂	1	0	0	...	0	0	0	0	0	0	0	0	0	0	0

whole earth thus there is no need for more than 64 bits data structure. 1 m² resolution generally considered as very high resolution and enough for detail spatial analysis, for example in hydraulic and hydrological modelling (Vaze et al. 2010).

$$Covered\ Area = 2^{P*2} * Resolution \quad (3)$$

For this paper, we focus on 64 bits data structure but the same method also can be applied for 32 bits data structure. The Octree node address X_pY_pZ_p for each level should be start concatenated from MSB to LSB with 0b1 appended after last 3 bits address as terminator as shown in Figs. 15 and 16.

Numbers of Octree level can be determined by (4) and since each level requires 3 bits of information embedded into the data structure, the total of bits that needs to be used is calculated according to (5).

$$\text{Numbers of Octree level (P)} = \frac{\log_2 \left(\left(\frac{\text{Area Size to Cover}}{\text{Smallest unit@resolution}} \right) \right)}{2} \quad (4)$$

$$\text{Bits Required} = 3P + 1 \quad (5)$$

2.4 CPU Accelerated Instruction

Bit Manipulation Instruction 2 (BMI2) is a series of new instructions introduced by Intel as part of Advance Vector Extensions 2 (AVX 2) in 2013 (Intel 2011). The same instruction and also supported by AMD in Excavator 2015. Intel and AMD is the market leader in computing with 80 % of the total computer sold is based on Intel or AMD CPU. There are 3 instructions from BMI2 that can be used to accelerate Octree address generation and manipulation; TZCNT, PDEP and PEXT.

2.4.1 Count the Number of Trailing Zero Bits (TZCNT)

TZCNT is a single instruction to count trailing 0 in the registers. This instruction eliminates the need of bit extraction-compare loops as shown in Fig. 17. TZCNT

```

temp ← 0
DEST ← 0
DO WHILE ( (temp < OperandSize) and (SRC[ temp] = 0) )
    temp ← temp +1
    DEST ← DEST+ 1
OD
IF DEST = OperandSize
    CF ← 1
ELSE
    CF ← 0
FI
IF DEST = 0
    ZF ← 1
ELSE
    ZF ← 0
FI

```

Fig. 17 TZCNT pseudo code (Intel 2011)

can be used to identify Octree level from voxel address as demonstrate in Sect. 2.5.1.

2.4.2 Parallel Bits Extract (PEXT)

PEXT is a single instruction to insert bits into different position from sequences as shown in Fig. 18. PEXT is being used in the proposed method to reconstruct voxel address as demonstrate in Sects. 2.5.2 and 2.5.4.

```

TEMP ← SRC1;
MASK ← SRC2;
DEST ← 0 ;
m← 0, k← 0;
DO WHILE m< OperandSize
    IF MASK[ m] = 1 THEN
        DEST[ k] ← TEMP[ m];
        k ← k+ 1;
    FI
    m ← m+ 1;
OD

```

Fig. 18 PEXT pseudo code (Intel 2011)

```
TEMP ← SRC1;
MASK ← SRC2;
DEST ← 0 ;
m← 0, k← 0;
DO WHILE m< OperandSize
    IF MASK[ m] = 1 THEN
        DEST[ m] ← TEMP[ k];
        k ← k+ 1;
    FI
    m ← m+ 1;
OD
```

Fig. 19 PDEP pseudo code (Intel 2011)

2.4.3 Instructions, Parallel Bits Deposit (PDEP)

PDEP is a single instruction to gather bits from different position into sequences as shown in Fig. 19. PDEP is being used in the proposed method to extract X Y and Z 3D spatial coordinate from voxel address as demonstrate in Sects. 2.5.3 and 2.5.4.

2.5 Manipulation

2.5.1 Determine Octree Level from Voxel Address

Using proposed voxel address encoding and data structures, we can easily determine the Octree level from voxel address alone without requiring the whole Octree traversal. Here TZCNT can help to determine the location of termination bit as shown in Fig. 20.

After getting the termination bit location, Octree level can be determined by using (6)

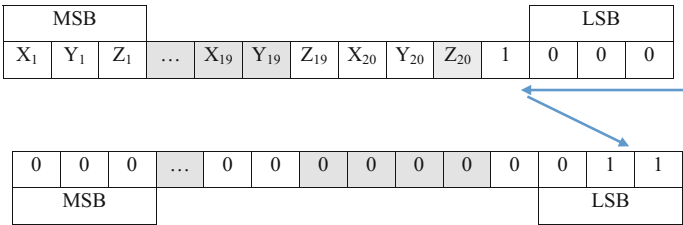


Fig. 20 Counting trailing zeros

Fig. 21 PDEP and PEXT mask generation

Xmask:
 $0x4924924924924924 \ll (TZCNT(\text{Voxel Address Register})+1)$

Ymask:
 $0x2492492492492492 \ll (TZCNT(\text{Voxel Address Register})+1)$

Zmask:
 $0x1249249249249249 \ll (TZCNT(\text{Voxel Address Register})+1)$

$$\text{Voxel Octree Level} = 21 - \frac{TZCNT((\text{Voxel Address Register}))}{3} \quad (6)$$

Octree level determination is important to determine the area/volume covered by respective voxel. Unlike voxel data representation, Octree voxel size is variable because Octree keeps a high resolution data only when needed in order to keep the data structure size optimized. The area covered by voxel is the inverse proportion of voxel level.

TZCNT value also will be used to generate Xmask, Ymask and Zmask for PDEP and PEXT instructions as shown in Fig. 21.

2.5.2 Voxel Address to 3D Spatial Coordinate

Our proposed method is to make the process to determine 3D spatial coordinate from voxel address simple and fast. With PEXT instruction, the extraction will be much faster. The coordinate can be easily extracted by using (7):

$$\text{Coordinate} = (PEXT(X\ mask), PEXT(Y\ mask), PEXT(Z\ mask)) \quad (7)$$

Figure 22 visualize how X axis coordinate can be extracted from voxel address. The same process is also true for Y and Z axis as shown in Figs. 23 and 24.

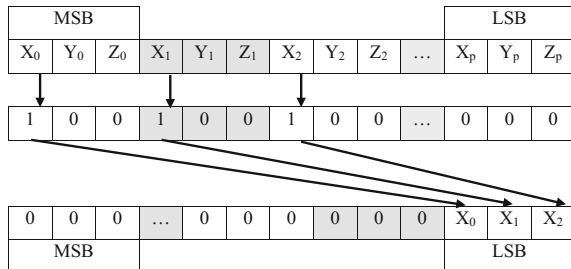
Fig. 22 Extracting voxel X coordinate

Fig. 23 Extracting voxel Y coordinate

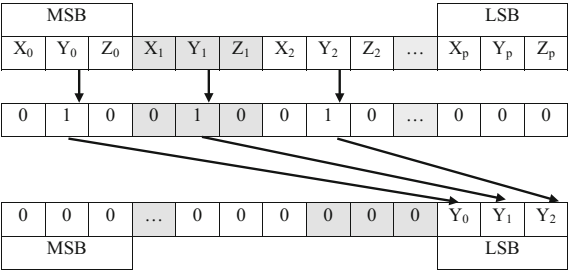
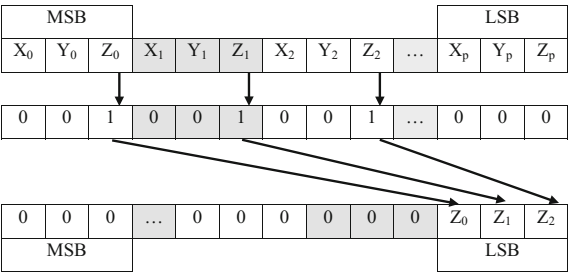


Fig. 24 Extracting voxel Z coordinate



2.5.3 3D Spatial Coordinate to Voxel Address

For application that needs voxel address generation for certain voxel coordinate like voxel neighbour discovery, PDEP can be used efficiently to reconstruct voxel address as shown in (8).

$$\text{Voxel address} = \text{PDEP}(X \text{ mask}) \cup \text{PDEP}(Y \text{ mask}) \cup \text{PDEP}(Z \text{ mask}) \quad (8)$$

Figure 25 visualize how X axis coordinate can be converted into voxel address. The same process is also true for Y and Z axis as shown in Figs. 26 and 27.

Fig. 25 Voxel address generation for X axis

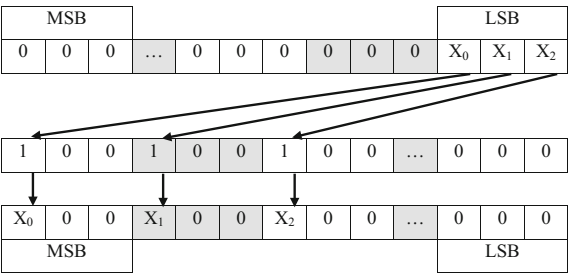
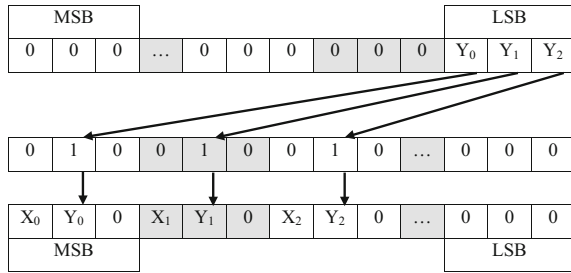
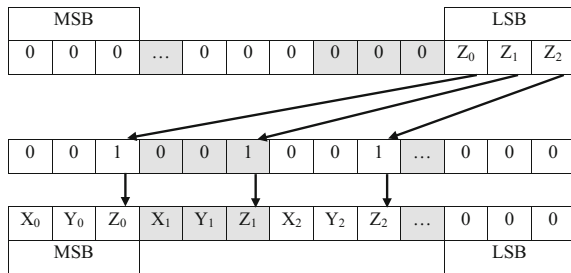


Fig. 26 Voxel address generation for Y axis**Fig. 27** Voxel address generation for Z axis

2.5.4 Finding Neighbour Voxel

Neighbour search is commonly used for spatial navigation and analysis. Finding the neighbouring search in classical Octree very resource extensive since it requires full node traverse to find each neighbour so in order to practically use Octree in those situation, address encoding like by proposed Voros (2000), Payeur (2006), Kim and Lee (2009) is favoured.

There are total of 26 neighbour voxels in an Octree and the offset of XYZ coordinate is shown in Fig. 28. Our proposed method is able to find each of them with 3 simple generic steps without involving complicated math or lookup table. For example, to get neighbour address at offset +X+Y + Z:

- (1) Extract voxel reference coordinate

$$X_{\text{ref}} = \text{PEXT}(X \text{ mask}, \text{Voxel}_{\text{ref}})$$

$$Y_{\text{ref}} = \text{PEXT}(Y \text{ mask}, \text{Voxel}_{\text{ref}})$$

$$Z_{\text{ref}} = \text{PEXT}(Z \text{ mask}, \text{Voxel}_{\text{ref}})$$

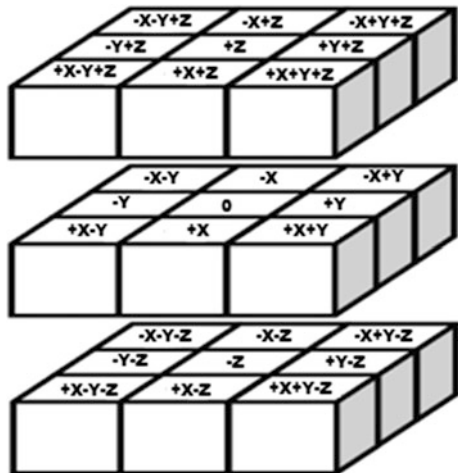
- (2) Add neighbour offset

$$X_{+X+Y+Z} = X_{\text{ref}} + 1$$

$$Y_{+X+Y+Z} = Y_{\text{ref}} + 1$$

$$Z_{+X+Y+Z} = Z_{\text{ref}} + 1$$

Fig. 28 Neighbours voxels



(3) Reconstruct Voxel_{+X+Y+Z} address

$$\begin{aligned} \text{Voxel}_{+X+Y+Z} = & \text{PDEP}(\text{X Mask}, X_{+X+Y+Z}) \cup \text{PDEP}(\text{X Mask}, Y_{+X+Y+Z}) \\ & \cup \text{PDEP}(\text{X Mask}, Z_{+X+Y+Z}) \end{aligned}$$

Our proposed method also has a fixed complexity regardless Octree level so it is suitable for simple and huge Octree structure.

3 Results and Discussions

Classical Octree is inefficient to find neighbour address and has a complexity $O(2^{3(p-1)})$ where p represents the maximum number of resolution levels in Octree. Payuer's (2004) method has a maximum complexity of $O(26(p-1))$ if there is no direction information given. Kim and Lee's (2009) method has the worst case complexity $O(6(p-1))$. Gargantini (1982), Schrack (Schrack 1992) both have $O(p)$ complexity. Our proposed method is independent of Octree level and has a fixed $O(1)$ complexity. Figure 29 shows comparison of the complexity neighbour search for those alternative implementation and our proposed method.

Another advantage of having data structure and encoding that is computer-friendly is the ability to speed up address encoding, extraction and neighbour search with BMI2 instruction. It is very fast and simpler to implement compared to generic implementation. Figure 30 shows comparison of performance between generic implementation and BMI2 accelerated instruction. By average, BMI2 implementation is 1000 times faster than generic implementation.

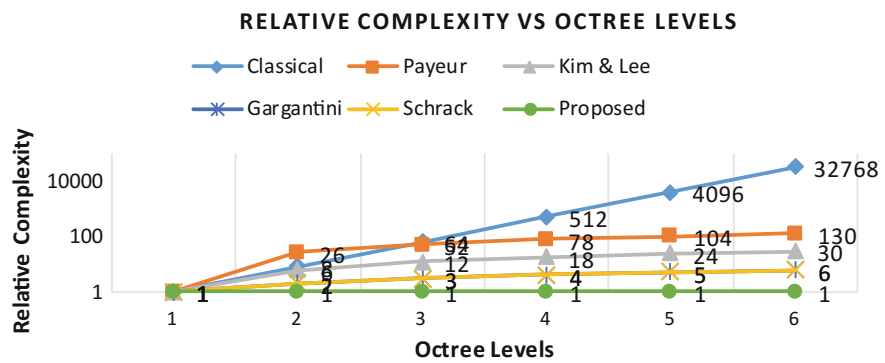


Fig. 29 Worst case complexity of neighbour search

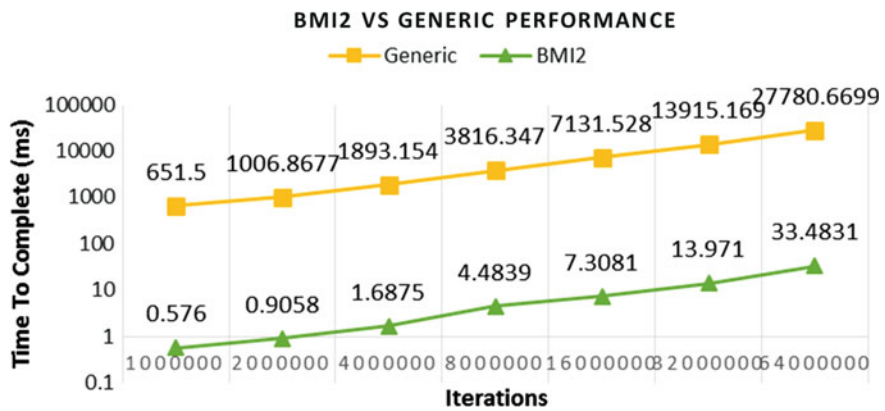


Fig. 30 Octree neighbour finding performance improvement with BMI2

4 Concluding Remarks

The key for faster and simple Octree address encoding scheme is the use of binary form. Octree in 3D space is binary compatible and by using the clever rule, 3D spatial information can be easily embedded inside the address itself and can be constructed, extracted and manipulated without using complex post processing. We have proposed a new 64 bits data structure which is designed with BMI2 compatible and introduced a simple and fast method for neighbouring search which is having constant complexity regardless Octree level. This method can be expanded in the future for neighbouring search aimed at different intra Octree level.

Acknowledgments The authors would like to thank the reviewers for their valuable comments and suggestions significantly improving this paper. The author also gratefully acknowledges financial support from the Ministry of Higher Education, Malaysia—Japan International

Cooperation Agency (MOHE-JICA) via grant scheme No. 203/PJJAUH/6711279 and UTM Research University Grant, Vote J.130000.2427.02G77 for their support and funding for this research work.

References

- Abdul-Rahman, A., & Pilouk, M. (2008). *Spatial data modelling for 3D GIS*. Springer.
- Ballard, D., & Brown, C. (1982). Computer vision.
- Berry, B. J. L., Griffith, D. A., & Tiefelsdorf, M. R. (2008). From spatial analysis to geospatial. *Science*, 40, 229–238.
- Besaçon, J., & Faugeras, O. (1988). Vision par ordinateur en deux et trois dimensions.
- Gargantini, I. (1982). Linear octrees for fast processing of three-dimensional objects.
- Goodchild, M. F. (2009). Geographic information systems and science: today and tomorrow. *Annals of GIS*, 15, 3–9.
- Huixin, W.U., & Huifeng, X.U.E. (2006). A new hybrid data structure for 3D GIS. In *First international conference on innovation in computing and information control* (Vol. 1, pp. 162–166). doi:[10.1109/ICICIC.2006.15](https://doi.org/10.1109/ICICIC.2006.15).
- Intel, I. (2011). *Advanced vector extensions programming reference*.
- Izham, M. Y., Muhamad Uznir, U., Alias, A. R., et al. (2011). Influence of georeference for saturated excess overland flow modelling using 3D volumetric soft geo-objects. *Computers and Geosciences*, 37, 598–609. doi:[10.1016/j.cageo.2010.05.013](https://doi.org/10.1016/j.cageo.2010.05.013).
- Jin, B., Fang, Y., & Song, W. (2011). 3D visualization model and key techniques for digital mine. *Transactions of the Nonferrous Metals Society of China*, 21, s748–s752. doi:[10.1016/S1003-6326\(12\)61674-4](https://doi.org/10.1016/S1003-6326(12)61674-4).
- Kim, J., & Lee, S. (2009). Fast neighbor cells finding method for multiple octree representation. *IEEE International Symposium on Comput Intelligence Robotics and Automation*, 2009, 540–545.
- Klinger, A. (1971). *Patterns and search statistics*.
- Lixin, W. (2004). Topological relations embodied in a generalized tri-prism (GTP) model for a 3D geoscience modeling system. *Computers and Geosciences*, 30, 405–418. doi:[10.1016/j.cageo.2003.06.005](https://doi.org/10.1016/j.cageo.2003.06.005).
- Payeur, P. (2004). An optimized computational technique for free space localization in 3-D virtual representations of complex environments. In *2004 IEEE Symposium on Virtual Environ Human-Computer Interfaces Measurement Systems 2004 (VCIMS)* (pp. 1–7). doi:[10.1109/VECIMS.2004.1397175](https://doi.org/10.1109/VECIMS.2004.1397175).
- Payeur, P. (2006). A computational technique for free space localization in 3-D multiresolution probabilistic environment models. *IEEE Transactions on Instrumentation and Measurement*, 55, 1734–1746. doi:[10.1109/TIM.2006.881028](https://doi.org/10.1109/TIM.2006.881028).
- Pouliot, J., Bédard, K., Kirkwood, D., & Lachance, B. (2008). Reasoning about geological space: Coupling 3D geomodels and topological queries as an aid to spatial data selection. *Computers and Geosciences*, 34, 529–541. doi:[10.1016/j.cageo.2007.06.002](https://doi.org/10.1016/j.cageo.2007.06.002).
- Li, R., Chen, Y., Dong, F., & Qian, L. (1996). 3D data structures and applications in geological subsurface modeling. In *International archives of photogrammetry and remote sensing* (Vol. XXXI, Part B4, pp. 508–513). Vienna.
- Rogers, J.D., & Luna, R. (2004). *Impact of geographical information systems on geotechnical engineering* (pp. 1–23).
- Rolf, A.D. (2004). *Principles of geographic information systems—An introductory textbook*.
- Samet, H. (1989). Neighbor finding in images represented by octrees. *Computer Vision, Graphics and Image Processing*, 45, 400. doi:[10.1016/0734-189X\(89\)90099-6](https://doi.org/10.1016/0734-189X(89)90099-6).
- Schrack, G. (1992). Finding neighbors of equal size in linear quadtrees and octrees in constant time.

- Shen, D., & Takara, K. (2006). A modelling and 3-D simulation system for water erosion on hillslopes—M3DSSWEH. *Journal of Hydraulic Research*, 44, 674–681. doi:[10.1080/00221686.2006.9521716](https://doi.org/10.1080/00221686.2006.9521716).
- Shen, D., Wong, D. W., Camelli, F., & Liu, Y. (2013). An ArcScene plug-in for volumetric data conversion, modeling and spatial analysis. *Computers and Geosciences*, 61, 104–115. doi:[10.1016/j.cageo.2013.08.004](https://doi.org/10.1016/j.cageo.2013.08.004).
- Shen, D. Y., Ma, A. N., Lin, H., et al. (2003). A new approach for simulating water erosion on hillslopes. *International Journal of Remote Sensing*, 24, 2819–2835.
- Shen, D. Y., Takara, K., Tachikawa, Y., & Liu, Y. L. (2006). 3D simulation of soft geo-objects. *International Journal of Geographical Information Science*, 20, 261–271. doi:[10.1080/13658810500287149](https://doi.org/10.1080/13658810500287149).
- Tianding, H. (2010). 3D GIS interactive editing method: Research and application in glaciology. Science and Engineering (ICISE), 2010 2nd, (pp. 1–4).
- Ujang, U., Rahman, A. A., Anton, F. (2014). *An Approach of Instigating 3D City Model s in Urban Air Pollution Modeling for Sustainable Urban Development in Malaysia An Approach of Instigating 3D City Model s in Urban Air Pollution Modeling f or Sustainability Urban Development in Malaysia* (pp. 1–22).
- Vaze, J., Teng, J., & Spencer, G. (2010). Impact of DEM accuracy and resolution on topographic indices. *Environmental Modelling and Software*, 25, 1086–1098. doi:[10.1016/j.envsoft.2010.03.014](https://doi.org/10.1016/j.envsoft.2010.03.014).
- Vörös, J. (2000). Strategy for repetitive neighbor finding in octree representations. *Image and Vision Computing*, 18, 1085–1091. doi:[10.1016/S0262-8856\(00\)00049-4](https://doi.org/10.1016/S0262-8856(00)00049-4).
- Wenzhong, S. (2000a). Development of a hybrid model for three-dimensional GIS. *Geo-Spatial Information Science*, 3, 6–12.
- Wenzhong, S. H. I. (2000b). Development of a hybrid model for three-dimensional. *GIS Geo-Spatial Information Science*, 3, 6–12.
- Zhu, L., Zhang, C., Li, M., et al. (2012). Building 3D solid models of sedimentary stratigraphic systems from borehole data: An automatic method and case studies. *Engineering Geology*, 127, 1–13. doi:[10.1016/j.enggeo.2011.12.001](https://doi.org/10.1016/j.enggeo.2011.12.001).
- Zhu, L. F., Li, M. J., Li, C. L., et al. (2013). Coupled modeling between geological structure fields and property parameter fields in 3D engineering geological space. *Engineering Geology*, 167, 105–116. doi:[10.1016/j.enggeo.2013.10.016](https://doi.org/10.1016/j.enggeo.2013.10.016).