SPATIO-TEMPORAL HIERARCHICAL DATA STRUCTURE BASED ON SPARSE VOXEL OCTREES FOR REPRESENTATION OF THE GEOMETRY OF TIME-VARIABLE THREE-DIMENSIONAL SCENES

Heidar KHORSHIDIYEH, Branislav MADOŠ, Dávid VAĽKO
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, Tel. +421 55 602 3023
E-mails: heidar.khorshidiyeh@tuke.sk, branislav.mados@tuke.sk, david.valko@tuke.sk

ABSTRACT

The paper deals with the problematics of representation of the geometry of three-dimensional scenes in computer graphics using domain-specific hierarchical data structures based on octant trees or directed acyclic graphs as the means of space saving binary representation of this information. Several hierarchical data structures were introduced in this field in past decades. In this paper we are moving this problematic to the field of the representation of three-dimensional scenes which are rather dynamic in time, so their geometry is changing on the regular basis. We are investigating the influence of this time-variability on the size of the related hierarchical data structure on binary level. We are comparing the approach in which each time step is considered as the stand-alone static three-dimensional scene and therefore is represented by its own hierarchical data structure with the approach when this time-variable scene is considered as the four dimensional space and spatio-temporal sparse voxel octree hierarchical data structure is used for the representation of the geometry of this scene. We tested both approaches on three different scenes which were obtained by voxelization into two different resolutions of polygonal surface models represented originally as the mesh of triangles in Wavefront Object file format. Results of the tests show that spatio-temporal sparse voxel octree is better solution, when we obtained from 1.73 to 2.08 more compact binary representation, in comparison to the solution when time frames were encoded separately as the set of sparse voxel octrees.

Keywords: *lossless data compression, voxelized spatio-temporal three-dimensional scenes, geometry of the scene, hierarchical data structures, sparse voxel octrees*

1. INTRODUCTION

The use of domain-specific hierarchical data structures (HDSs) such as quadtrees for 2D data or octrees for 3D data has been explored for decades. We can mention works from the 80s of the last century - the data structure 2Dimensional Template-Based Encoding (2DTE) [1] or works [2, 3, 4]. These HDSs represent homogeneous sub-volumes in an economical way. If they are homogeneously filled with passive voxels, they are considered passive, if they are homogeneously filled with active voxels, they are considered active. If they are not homogeneously filled, they are intended for decomposition.

Modern hierarchical data structures are referred to as GPU friendly. High Resolution Sparse Voxel Directed Acyclic Graphs (HR SVDAGs) were introduced in 2013 [5] (from which GPU friendly HDS Sparse Voxel Octrees - SVOs were later retroactively defined). Other hierarchical data structures were later derived (more in the Related works section). Those HDSs are intended for the representation of 3D voxelized scenes that contain a very small proportion of active voxels. So they are sparse and that transformed into the names of these data structures. In the case of these modern HDSs, passive octants are homogeneously filled with passive voxels. Active octants are those octants that contain at least one active voxel, and these octants are subject of recursive decomposition.

Much of the research in this area has been devoted to the design of HDSs for the representation of the geometry of static three-dimensional scenes. Proposed HDSs have been very successful in compacting this information at the binary level.

Ad-hoc modification of the geometry of static 3D scenes, however, means computationally demanding decompression of the data structure, making of the change and recompression [6]. A special problem is the representation of the geometry of 3D scenes, which are time-variable, that is, which geometry changes systematically in individual time frames.

The paper deals with this issue and its contribution is in the examination of the assumption that the use of spatio-temporal Sparse Voxel Octrees (SVOs) allows more compact storage of the geometry of a time-variable scene compared to its storage in the form of SVO sets, when SVO is built for each time frame separately.

The structure of the paper is as follows:

Section 2 of the paper is devoted to related works in the field of representing the geometry of 3D scenes through domain-specific hierarchical data structures based on Sparse Voxel Octrees (SVOs) and Sparse Voxel Directed Acyclic Graphs (SVDAGs) and data structures derived from them.

Section 3 of the paper deals with the description of the representation of the time-varying scene geometry by means of classical SVOs and through spatio-temporal Sparse Voxel Octrees.

Section 4 of the paper summarizes the results of tests carried out on three scenes voxelized into two resolutions, the geometry of which was stored in the form of SVO sets and in the form of spatio-temporal Sparse Voxel Octrees.

Section 5 of the paper presents the conclusions that were made based on the tests, the results of which were summarized in the previous section of the paper.

2. RELATED WORKS

A section of the paper deals with domain-specific hierarchical data structures used to represent the geometry of voxelized three-dimensional scenes, devoting only resources that are particularly very closely related to the topic of this paper.

The basic data structure intended for representing the geometry of a voxelized 3D scene is Sparse Voxel Octrees (SVOs). For its construction, a two-step out-of-core algorithm was created in 2013 [7, 8]. In the first step of this algorithm, voxelization takes place, the input of which is a mesh of triangles. In the second step, the output of the first step, which is the list of active voxels, is used and the SVO is constructed. In 2015, an algorithm was presented [9] enabling voxelization into SVO without the need for an intermediate product. SVO HDS contains pointers to child nodes. If pointers are absent, the data structure is called Pointerless Sparse Voxel Octrees (PSVOs).

In 2022, HDS called Clustered Sparse Voxel Octrees (CSVO) was introduced [10]. It brings a more compact binary representation of the HDS at the cost of a more complex construction of nodes, of which there are more compared to other HDSs. CSVO consists not only of internal nodes (INODEs) and leaf nodes (LNODEs) as other HDSs do, but also of so-called preleaf nodes (PLNODEs). Nodes support multiple lengths of binary representation of pointers to child nodes.

While SVO represents a tree-based HDS, in 2013 a directed acyclic graph-based HDS, called High Resolution Sparse Voxel Directed Acyclic Graphs (HR SVDAGs) was introduced [5]. While a tree does not allow a child node to have multiple parent nodes, a directed acyclic graph does. The main difference compared to SVO is the possibility to search for matching subtrees within one data structure and to represent them economically in such a way that these trees are excluded from the structure, leaving only one of them and referring to it multiple times. This is called Common Subtree Merge (CSM).

In 2016, HDS called Symmetry-aware Sparse Voxel Directed Acyclic Graphs (SSVDAGs) was reported in [11,12]. In this data structure reflective symmetry is used, i.e. mirroring, which can be independently realized in each of the axes of space, i.e. in the x, y and z axes. Multiple lengths of pointers to child nodes are used at the same time, when short pointers are 16b and long pointers are 32b. The most frequently addressed nodes are located in the address space covered by short pointers. Therefore, nodes in each layer of this HDS are sorted according to their addressing frequency. This is called Frequency Based Compaction (FBC). Grids of voxels with size 4^3 are used as leaf nodes.

In 2020, an HDS called Pointerless Sparse Voxel Directed Acyclic Graphs (PSVDAGs) was introduced [13]. It enables CSM in the absence of pointers to child nodes, which is possible thanks to the fact that labels have been introduced to mark subtrees that will be referenced multiple times and callers have been introduced to call these subtrees from their multiple occurrence locations. HDS has a wide range of pointer lengths to child nodes and uses FBC. Quick transformation algorithm from PSVDAG to SVDAG was introduced in [14].

3. PROPOSED SOLUTION

In this paper we are comparing two approaches to the representation of the geometry of a time-varying three-dimensional voxelized scene.

The first approach is to view each frame of a time-varying scene as a separate static three-dimensional scene. In that case, its geometry can be represented through data structures that are currently established. These include Sparse Voxel Octrees (SVOs), Sparse Voxel Directed Acyclic Graphs (SVDAGs), Symmetry-aware Sparse Voxel Directed Acyclic Graphs (SSVDAGs) or Clustered Sparse Voxel Octrees (CSVOs). SVO was used in this paper. In the case of a time-variable scene with n time frames, an ordered set of SVOs with cardinality n will be created. The scene must have dimensions N^3 and $N = 2^m$; $N, m \in \mathbb{N}$. The number of time frames n can be arbitrary in this case.

In the case of a time-varying two-dimensional image, each time frame can be considered as the new static two-dimensional image, similar to the previous approach (quadrant trees could be used to store their geometry in that case). However, it is possible to integrate these two-dimensional time frames into the form of three-dimensional space and represent its geometry with the help of classic SVO, as shown in Fig. 1a. Similarly, one can look at the time frames of a 3D scene and integrate them into a form of 4D space, as shown in Fig. 1b, and represent the geometry of this space through a single hierarchical data structure that is analogous to SVO, and which is named as spatio-temporal SVO in this paper.

A limiting factor of this approach may be that the generated scene must meet the conditions when it has dimensions N^4 and $N = 2^m$; $N, m \in \mathbb{N}$. The number of time frames must therefore be the same as the geometric dimension N.

The consequences of this limiting assumption can be mitigated so that if the number of time frames is greater than N, they are divided into sets of cardinality N (the last of them will thus have cardinality $\leq N$). If the number of time frames is less than N, the missing time frames will be filled in, and their voxels will be set as homogeneously passive. Each completed frame will contribute N^3 voxels to the total number of voxels of all time frames. Each set of frames will be then compressed to spatio-temporal SVO.

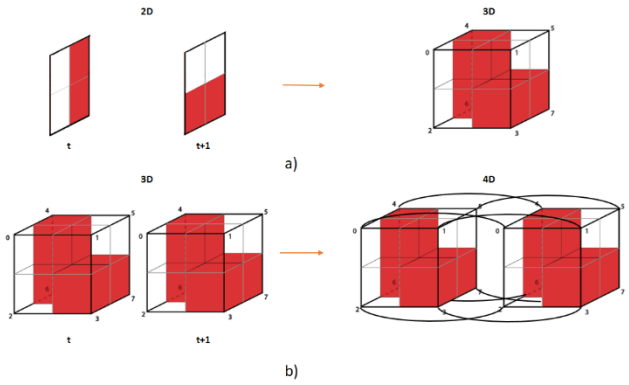


Fig. 1 Representation of a) time-variable 2D scene as the 3D static scene and b) time-variable 3D scene as the 4D static scene

Another option to mitigate this consequence is to adjust the geometric dimensions of the scene, when $N = 2^m$ is adjusted to $N_1 = 2^{m_1}$, where $m_1 > m$. The number of coded time frames can thus become $\leq N_1$. If the value of m is increased by 1, the number of voxels of each time frame is increased by $7 \times N^3$.

Both, increasing the number of time frames and increasing the geometric resolution of the scene, contribute a large number of new voxels to the grid of scene voxels. However, although large subspaces of the scene are created, they are homogeneously filled with passive (empty) voxels. The binary representation of such homogeneous passive subspaces of scenes using hierarchical data structures such as SVO is extremely economical, so the addition of these passive voxels will not significantly affect the overall size of the binary representation of the data structures.

We are using Backus-Naur Form for formal description of the a) SVO and b) spatio-temporal SVO. The difference is in the length of the child node mask (CHNM) and the number of pointers in array of pointers (PTS).

```
SVO ::= (n)<NODE>
NODE ::= <INODE>|<LNODE>
INODE::=<CHNM>(p)<BIT><PTS>
LNODE::= <CHNM>(q)<BIT>
CHNM ::= (8)<HT>
PTS ::= (1)*(8)<PT>
PT ::= (r)<BIT>
HT ::= <BIT>
BIT ::= "0" | "1"
```

a)

```
STSVO ::= (n)<NODE>
NODE ::= <INODE>|<LNODE>
INODE::=<CHNM>(p)<BIT><PTS>
LNODE ::= <CHNM>(q)<BIT>
CHNM  ::= (16)<HT>
PTS   ::= (1)*(16)<PT>
PT    ::= (r)<BIT>
HT ::= <BIT>
BIT   ::= "0" | "1"
```

b)

where the following applies:
<SYM> - a mandatory non-terminal symbol SYM
"sym" - terminal symbol sym
(n)<SYM> - the SYM symbol, concatenated n-times
(n)*(m)<SYM> - the SYM symbol, concatenated n to m times
| - alternative

4. RESULTS

In the tests we used three models, Angel Lucy, Skull and Porsche, originally stored as polygonal surface models composed of a mesh of triangles. Angel Lucy was composed from 488,880 triangles; Skull from 80,016 triangles; and Porsche from 22,011 triangles. Subsequently, they were each stored in their own scene and these scenes were voxelized into two resolutions 128^3 and 256^3 voxels. The number of active voxels of individual static three-dimensional scenes is shown in Table I. Visualizations of these scenes can be seen in Fig. 2.

In all tests, the number of time frames generated was identical to the geometric resolution of the scene. In the case of scenes with a dimension of 128^3, it was 128 time frames, in the case of a resolution of 256^3, it was 256 time frames.

The temporal variability of the scenes was achieved by moving the models in the linear trajectory in one or more of the x, y and z axes of the scene, both in terms of trajectory and speed. In each time frame, the object in the scene moved by one voxel in the given axis, so it can be said that position of the model in the scene was the function of the number of time frame.

In tests, movements of objects in scenes were realized in the manner that in the test no. 1 movement was in the x axis, in the test no. 2 movement was in the y axis, in the test no. 3 movement was in the z axis, in the test no. 4 movement was in the x and y axis, in the test no. 5 movement was in the x and z axis and in the test no. 6 movement was in the x, y and z axis.

The compaction of the binary representation of the information regarded the geometry, that was achieved in tests, ranged from 1.73 to 2.08 times when spatio-temporal SVO was used instead of the set of SVOs, as it can be seen in Table II.

Table 1 Static three-dimensional scenes Angel Lucy, Skull and Porsche active voxels count in thousands and share of active voxels on overall count of voxels of the scene

		Angel Lucy		Skull		Porsche	
Resolution		128^3	256^3	128^3	256^3	128^3	256^3
Voxels	**[10^3]**	2 097	16 777	2 097	16 777	2 097	16 777
Active voxels	**[10^3]**	22.48	91.52	74.10	298.85	54.20	233.40
	%	1.70	0.55	3.53	1.78	2.58	1.39



a) b) c)

Fig. 2 Visualizations of 3D scenes a) Angel Lucy, b) Skull and c) Porsche

Table 2 Test results for different scenes, resolutions and kinds of move of the object within scene

		Angel Lucy		Skull		Porsche	
Resolution		128^4	256^4	128^4	256^4	128^4	256^4
Test 1 *move in axis x*	**3D SVOs [KB]**	5 769	49 133	15 912	130 371	11 562	107 563
	STSVO [KB]	2 946	25 234	8 223	67 570	5 714	53 556
	Size ratio	1.96	1.95	1.94	1.93	2.02	2.01
	Active voxels [10^3]	2 416	19 635	6 327	50 883	5 445	46 690
	Active voxels [%]	0.90	0.46	2.36	1.18	2.03	1.09
Test 2 *move in axis y*	**3D SVOs [KB]**	3 704	31 506	13 339	110 350	13 443	123 043
	STSVO [KB]	1 814	15 412	6 590	54 447	6 810	64 528
	Size ratio	2.04	2.04	2.03	2.03	1.97	1.91
	Active voxels [10^3]	1 556	12 582	5 378	43 496	6 158	52 773
	Active voxels [%]	0.58	0.29	2.00	1.01	2.29	1.23
Test 3 *move in axis z*	**3D SVOs [KB]**	4 861	41 618	12 831	106 052	7 209	66 316
	STSVO KB]	2 446	21 152	6 368	52 845	3 461	31 957
	Size ratio	1.99	1.97	2.01	2.01	2.08	2.08
	Active voxels [10^3]	2 047	16 613	5 171	41 778	3 338	28 603
	Active voxels [%]	0.76	0.39	1.93	0.97	1.24	0.67
Test 4 *move in axis x, y*	**3D SVOs [KB]**	3 566	30 176	11 061	91 285	11 398	105 373
	STSVO [KB]	1 901	16 220	6 019	49 745	6 118	59 364
	Size ratio	1.88	1.86	1.84	1.84	1.86	1.78
	Active voxels [10^3]	1 491	12 068	4 449	35 892	5 309	45 523
	Active voxels [%]	0.56	0.28	1.66	0.84	1.98	1.06
Test 5 *move in axis x, z*	**3D SVOs [KB]**	4 854	41 290	11 137	91 907	6 702	62 300
	STSVO [KB]	2 610	22 585	6 123	50 801	3 458	32 795
	Size ratio	1.86	1.83	1.82	1.81	1.94	1.90
	Active voxels [10^3]	2 033	16 499	4 482	36 158	3 158	27 025
	Active voxels [%]	0.76	0.38	1.67	0.84	1.18	0.63
Test 6 *move in axis x, y, z*	**3D SVOs [KB]**	3 387	28 760	8 960	74 406	6 762	62 475
	STSVO [KB]	1 869	16 106	5 071	42 299	3 702	36 018
	Size ratio	1.81	1.79	1.77	1.76	1.83	1.73
	Active voxels [10^3]	1 421	11 506	3 637	29 422	3 156	27 010
	Active voxels [%]	0.53	0.27	1.35	0.69	1.18	0.63

5. CONCLUSIONS

The paper dealt with the issue of coding the information regarding the geometry of voxelized three-dimensional scenes using domain-specific hierarchical data structures based on octant trees and directed acyclic graphs including SVOs, CSVOs, SVDAGs, SSVDAGs and others. The paper moves from static scenes and concentrates on the coding of the time-variable scenes geometry. In consideration were two different approaches. First approach treated each time frame as a static 3D scene and used classical SVO HDS to store the geometry. Thus, an ordered set of SVOs was created. The second approach considered the time-variable scene as a 4D scene and used spatio-temporal Sparse Voxel Octree HDS to store its geometry in one HDS.

Performed tests showed that the use of spatio-temporal Sparse Voxel Octree enables a more compact representation of the geometry of time-variable voxelized scenes compared to the set of SVOs. The compaction of the binary representation of this information, that was achieved in tests, using spatio-temporal SVO instead of the set of SVOs ranged from 1.73 to 2.08 times.

The following research in this topic will be focused on the development of a more sophisticated hierarchical data structures that are able to use time coherency between frames, while the influence of time variability factors on the achieved compression ratios also will be considered.

ACKNOWLEDGMENTS

This work was supported by KEGA Agency of the Ministry of Education, Science, Research, and Sport of the

Slovak Republic under Grant No. 015TUKE-4/2024 Modern Methods and Education Forms in the Cybersecurity Education.

REFERENCES

[1] CHANG, H. K. – LIU, S.-H. – TSO, C.-K.: Two-Dimensional Template-Based Encoding for Linear Quadtree Representation. Photogrammetric Engineering and Remote Sensing, Vol. 63, 1997, No. 11, pp. 1275–1282.

[2] MEAGHER, D. J. R.: Octree Encoding: A New Technique for the Representation, Manipulation, and Display of Arbitrary 3-D Objects by Computer. Technical Report No. IPL-TR-80-111. Rensselaer Polytechnic Institute, Troy, NY, 1980.

[3] MEAGHER, D. J. R.: Geometric Modeling Using Octree Encoding. Computer Graphics and Image Processing, Vol. 19, 1982, No. 2, pp. 129–147, doi: 10.1016/0146-664X(82)90104-6.

[4] MEAGHER, D. J. R.: The Octree Encoding Method for Efficient Solid Modeling. Technical Report IPL-TR-032, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, 1982.

[5] KÄMPE, V. – SINTORN, E. –ASSARSON, U.: High Resolution Sparse Voxel DAGs, ACM ToG, 2013, 32, pp. 1–13, ISSN 0730-0301, https://doi.org/10.1145/2461912. 2462024

[6] CAREIL, V. – BILLETER, M. – EISEMANN, E.: Interactively Modifying Compressed Sparse Voxel Representations. In {\em Computer Graphics Forum}, {\bf 2020}, {\em 39}, pp. 111–119. ISSN 0167-7055. https://doi.org/10.1111/cgf.13916

[7] BAERT, J. – LAGAE, A. – DUTRÉ, Ph.: Out-of-Core Construction of Sparse Voxel Octrees. In Proceedings of the 5th High-Performance Graphics Conference (HPG ’13), Anaheim, CA, USA, 19–21 July, 2013, pp. 27–32. https://doi.org/10.1145/ 2492045.2492048

[8] BAERT, J. – LAGAE, A. – DUTRÉ, Ph.: Out-of-Core Construction of Sparse Voxel Octrees, Computer Graphics Forum, 2014, 33, pp. 220–227, ISSN 0167-7055, https://doi.org/10.1111/cgf.12345

[9] PATZOLD, M. – KOLB, A.: Grid-free out-of-core voxelization to sparse voxel octrees on GPU. In Proceedings of the 7th Conference on High-Performance Graphics (HPG '15), LA, CA, USA, 7--9 August 2015, pp. 95–103. ISBN 9781450337076, https://doi.org/10.1145/2790060. 2790067

[10] MADOŠ, B. – CHOVANCOVÁ, E. – CHOVANEC, M. – ÁDÁM, N.: CSVO: Clustered Sparse Voxel Octrees–A Hierarchical Data Structure for Geometry Representation of Voxelized 3D Scenes. Symmetry 2022, 14, 2114. https://doi.org/10.3390/sym14102114

[11] VILLANUEVA, A.J. – MARTON, F. – GOBETTI, E.: Symmetry-aware Sparse Voxel DAGs. In Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D ’16), Redmond, WA, USA, 27–28 February, 2016; pp. 7–14. ISBN 978-1-4503-4043-4. https://doi.org/10.1145/2856400.2856420

[12] VILLANUEVA, A. J. – MARTON, F.: GOBETTI, E.: Symmetry-aware Sparse Voxel DAGs (SSVDAGs) for compression-domain tracing of high-resolution geometric scene. J. Comput. Graph. Tech., (JCGT), 2017}, 6, pp. 30, http://jcgt. org/ published/0006/02/01

[13] VOKOROKOS, L. – MADOŠ, B. – BILANOVÁ, Z.: PSVDAG: Compact Voxelized Representation of 3D Scenes Using Pointerless Sparse Voxel Directed Acyclic Graphs, Computing and Informatics, 2020, 39, pp. 587-616, ISSN 1335-9150 (print); 2585-8807 (online), https://doi.org/10.31577/cai_2020_3_587

[14] MADOŠ, B. – ÁDÁM, N.: Transforming Hierarchical Data Structures---A PSVDAG---SVDAG Conversion Algorithm, Acta Polytechnica Hungarica, 2021, 18, pp.47-66, ISSN 1785-8860, https://doi.org/10.12700/APH.18.8.2021.8.3

Received June 24, 2024, accepted July 2, 2024

BIOGRAPHIES

Heidar Khorshidiyeh (Ing., PhD. Student) was born in Al Ain, UAE. He holds Iranian nationality. In 2019, he graduated with a Bachelor's degree (Bc.) from the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. In 2021, he earned his Master's degree (Ing.) from the same department and faculty. Since 2021, he is PhD student at the Technical University of Košice. His scientific research is centered on computer graphics, specifically on the representation of 3D scenes using hierarchical data structures

Branislav Madoš (doc. Ing., PhD.) was born in Trebišov, Slovakia. In 2006 he graduated (Ing.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. He defended his PhD. in the field of Computers and computer systems in 2009; his thesis title was "Specialized architecture of data flow computer". From 2021 he is an associate professor at the DCI, FEI, TUKE. His scientific research is focused on the representation of voxelized 3D scenes using hierarchical data structures.

Dávid Vaľko (Ing. Bc. PhD. student) was born in Košice, Slovakia. He completed his master’s degree in Computer science in 2021 at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Since 2021, he has been a PhD. student at the TUKE. His scientific research focuses on the use of computer science in medicine, particularly the classification and processing of medical signals such as ECG. He also received a bachelor's degree in Urgent medical care in 2023.