ON DATA HIDING USING DOMAIN SPECIFIC HIERARCHICAL DATA STRUCTURE SVDAG FOR GEOMETRY REPRESENTATION OF VOXELIZED THREE-DIMENSIONAL SCENES

Branislav MADOŠ, Zuzana BILANOVÁ
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: {branislav.mados | zuzana.bilanova}@tuke.sk

ABSTRACT

The paper deals with the problematics of data hiding into domain specific hierarchical data structures which are dedicated to the geometry of the voxelized three-dimensional scenes representation. Sparse Voxel Directed Acyclic Graph (SVDAG) as the suitable representant of this category of data structures was chosen. Possibility of data hiding into this structure, based on the knowledge of its construction on binary level was investigated and several data hiding methods were described and leveraged as the contribution of this paper.

Keywords: *data hiding, steganography, sparse voxel octrees, SVO, sparse voxel directed acyclic graph, SVDAG, symmetry-aware sparse voxel directed acyclic graph, SSVDAG*

1. INTRODUCTION

Computer security, cybersecurity or information technology security is becoming more important nowadays because of our rising reliance on information technology. There are fields of IT security including intrusion detection [1][2], honeypot technology [3][4], security using computer vision [5] etc. Important part of the field includes cryptography and not so often mentioned steganography.

Steganography is considered to be not only a science, but also a craft of concealing ongoing communication by hiding messages into unsuspicious cover documents called stegomedia, such as texts, digital images, audio and video sequences and other domain specific data structures. The word steganography comprises the Greek words *steganos* (στεγᾰνός) that means covered or concealed and the word *graphein* (γραφή) that means writing. Steganography as the word has been used in 1499 for the first time by Johannes Trithemius in his work *Steganographia*, but the first use of steganography as the method itself can be dated back to the 440 BC, when the use of steganographic procedures have been mentioned by Herodotus in his work *Histories*. Interesting overview of the problematics of steganography can be found in [6][7].

Steganography is evolving not only through centuries but lieterally through ages and different techniques have been proposed using rich repertoire of various stegomedia. It is possible to use one of basic classifications of steganography, when pre-computer and computer era of steganography can be established. Computers are not only simplifying and accelerating use of steganographic techniques, but brings also possibility of computer specific algorithms and stegomedia that does not have opportunity to exist without computers. Steganography is constantly seeking new possibilities how to hide information in those innovative kinds of computer related stegomedia and tries to find computer specific algorithms of data hiding. To name only a few examples, with the rise of computers possibility of the use of secondary storage systems, their changeable media and file systems were examined for the possibility to hide information based on the hardware and software features [8][9][10]. Another way how to hide information which is connected to the use of computers is the possibility to hide information into pictures stored in JPEG file format [11][12]. Specific algorithms were created in relation to rise of smartphones and their operating systems including iOS and Android.

That is why we decided to concentrate our attention to the computers related domain specific data structures which are dedicated to the voxelized three-dimensional scenes geometry representation, that are including Sparse Voxel Octrees (SVO), Sparse Voxel Directed Acyclic Graphs (SVDAG) and Symmetry-Aware Sparse Voxel Directed Acyclic Graphs (SSVDAG). The aim of the paper is to investigate possibilities of the data hiding into Sparse Voxel Directed Acyclic Graph (SVDAG) data structure and leverage their usefulness and capacity of the SVDAG format for data hiding.

Contribution of the paper is in Discussion of the several methods of data hiding based on the leveraging of the features of binary representation of Sparse Voxel Directed Acyclic Graph (SVDAG) data structure nodes.

The structure of the paper is as follows.

Section 2 of the paper summarizes related work in the field of hierarchical data structures used for geometry representation of voxelized multi-dimensional data, including 2D and 3D grids of pixels resp. voxels.

Section 3 of the paper introduces construction of the Sparse Voxel Directed Acyclic Graph (SSVDAG) and describes nodes construction of this data structure on the binary level.

Section 4 of the paper discusses possibilities of data hiding into SVDAG data structure based on the knowledge about binary representation of the data structure described in the section 2 of the paper and leverages advantages and disadvantages of mentioned data hiding methods from the capacity and usefulness point of view.

Section 5 of the paper summarizes conclusions based on previous sections of the paper.

2. RELATED WORKS

Multi-dimensional data linearization. For the purpose of linearization of multi-dimensional data Morton order is often used, because of many advantages. Morton order is known also as the Z-order or Morton Code [13] because it uses Z-Curve, it means curve in the shape of the character Z, as the space-filling curve, as it is depicted on the Figure 1 for two-dimensional space.

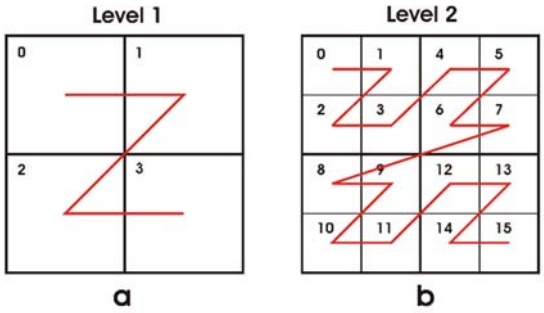

Fig. 1 Morton order, using z-curve as the tool for linearization of multidimensional grids of data. Example shows two-dimensional data linearized a) to the first level and b) to the second level.

Voxelization. is the process in which in most cases regular three-dimensional grid of voxels is created from different kinds of data sources. In [14] there is out-of-core algorithm described which consists of two steps. First step is represented by the voxelization process in which set of voxels is created from the triangle mesh as the intermediate product. In the second step of the algorithm Sparse Voxel Octree (SVO) data structure is created from the set of voxels.

Two-dimensional hierarchical representation. Possibility of two-dimensional picture representation using quadtrees along with the use of Common Subtree Merging was introduced in [15]. Two-Dimensional Template-based encoding (2DTE) along with the use of common subtrees merging was introduced in [16] as the solution for cartographical information representation. Extension of this approach to 3D data was introduced in [17].

Three-dimensional hierarchical representation. of voxelized data is evolving over last decade from Sparse Voxel Octrees (SVO) through the Efficient Sparse Voxel Octrees (referenced in literature as the ESVO) that were introduced in 2010 in [18]. In 2013 Sparse Voxel Directed Acyclic Graphs (SVDAG) were introduced in [19]. It transforms octrees into directed acyclic graph, because it allows common subtrees merge. Another advantage is in 32 bit pointers to the child nodes. There are pointers with the 0 and 32-bit length. Evolution of this data structure into Symmetry-aware Sparse Voxel Directed Acyclic Graph was introduced in [20]. Pointers to the child nodes are 0, 16 and 32-bit long and there is possibility to common subtree merge not only in the case of absolute match of two or more subtrees, but also if subtrees are identical when symmetry transformation is applied in one or more axes. Leaf nodes are compacted into 4^3 grids of voxels, because it creates chance for compaction of the data representation in comparison of the use of pointers.

3. SVDAG DATA STRUCTURE

Sparse Voxel Directed Acyclic Graph (SVDAG) is data structure that unlike previous data structure called Sparse Voxel Octree (SVO) brings new features:

- Pointers to the child nodes, that are 32-bit long
- Possibility of common subtrees merge
- 32 bit aligned components of data structure node

Pointers bring possibility of quick and easy traversing of the data structure when it is loaded into operating memory of the computer or memory of the graphics card. Pointers are 32-bit long and represent address of particular node in whole addressing space of the data structure.

SVDAG is directed acyclic graph, because it allows common subtree merge, when only one copy of the subtree is fully represented and this subtree is then multiple times referenced using pointers that are referencing the same address in addressing space of the data structure (are referencing the same node, that is root of the common subtree).

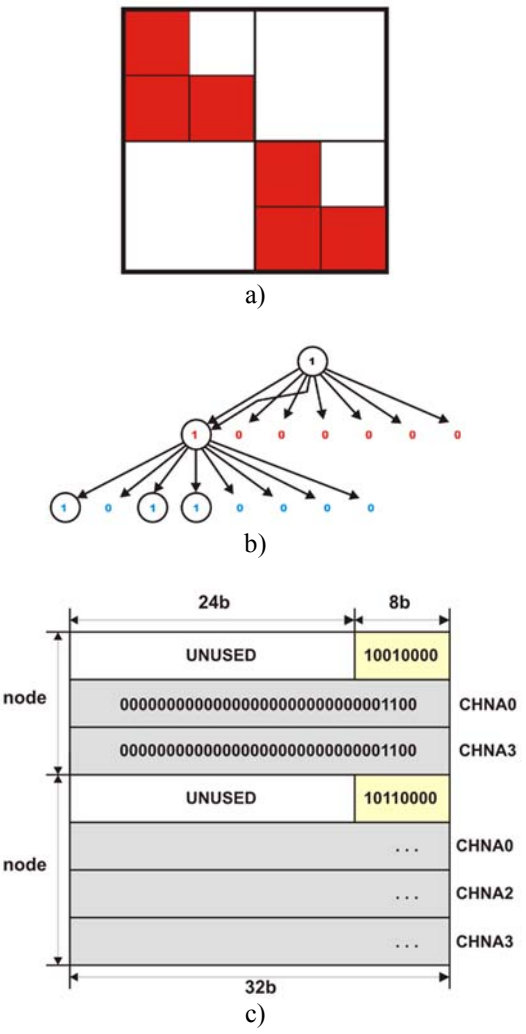

Fig. 2 Example of Sparse Voxel Directed Acyclic Graph construction for voxelized data (2D for simplification) when a) representation of active pixels, which are red b) representation in the form of directed acyclic graph and c) example of binary representation of nodes with CHild Node Addresses (CHNA). Pixels are in Morton order.

Advantage of common subtree merge is the compaction of the data structure, that can be considered as the lossless data compression. Very important in that case is the decompression overhead in time of the data structure traversing, because data structure is continually and repeatedly decompressed, when it is used for data visualization or other possible kind of processing, especially in interactive or real time mode. Advantage is that there is no decompression overhead of common subtree merge in comparison to SVO data structure which is enriched with pointers and not using common subtree merge. Example of SVDAG can be seen on Figure 2.

In the same way as the Sparse Voxel Octrees (SVO) data structure, SVDAG decomposes space into 8 suboctants and therefore each node of the SVDAG comprises 8-bit child node mask which includes one bit per child node. When this bit is set to 0, there is no child node, because suboctant comprises no active voxels and there is no need of further decomposition. There is no pointer to the child node. When this bit is set to 1, suboctant comprises at least one active voxel and further decomposition of the suboctant is needed, and there is child node present in the data structure and also pointer which references this child node. There is 24 bit long unused space along with the child node mask, because there was need for 32-bit alignment of the mask in the data structure. This unused space can be considered as the disadvantage of the data structure along with the length of child pointers that can be considered as unnecessary long when only smaller volumes are stored.

Number of child nodes is from the range of $< 1; 8 >$ and each child node is represented by its own child node pointer. Number of child nodes is therefore variable in the same range $< 1; 8 >$. Mask with unused space is 32-bit long and each pointer is 32-bit long. Each node comprises mask and at least one child node pointer with maximum of 8 child node pointers. Length of the node is therefore from the range of $< 64; 288 >$ bits ($< 8; 36 >$ Bytes). Leaf nodes do not need any child node pointers and it is possible to represent voxels directly in the child node mask of the node, so there is possibility to use 32-bits for this node.

4. DATA HIDING METHODS

The natural requirement for data hiding into SVDAG data structure is not to change represented geometry of the three dimensional scene. All discussed methods are in accordance with this requirement.

Unused space. First possibility how to hide information into SVDAG data structure is obvious possibility to fill information into unused space related to the child node mask. There is 24 bits (3 Bytes) of unused space in each node of the data structure. In comparison to the overall length of the binary representation of the node, which can be from the range of $< 64; 288 >$ bits per node, it is possible to use relatively big part of the space for data hiding. When this unused space represents from 8.33% when all 8 child pointers are used to the 37,5% when only one child pointer is used. In the case of leaf nodes, unused space represents even bigger portion – 75% of the binary representation of the leaf node (Figure 3). Relatively big capacity of this unused space can be considered as the advantage of this possibility of hiding information, however it is extremely easy to find out that this space if filled with information, especially in case of extensive use of the disposable capacity of data structure.

Extra child node pointers. Number of child pointers for particular node can be found by analysis of child node mask of the node and this number can be from the range of <1; 8> for non-leaf nodes. It is not possible to introduce extra child node pointer prior to the last real child node pointer however there is possibility to concatenate extra child node pointer after last of regular one. This extra child node pointer does not have its particular bit in child node mask. Not to exceed the natural number of child node pointers of regular node of the data structure, there is possibility to concatenate from 0 to 7 extra child node pointers, which brings possibility to store from 0 to 224 bits of information (0 to 28 Bytes) which is from 0% to 77,78% of the normal child node when full capacity of extra child pointers is used (Figure 4).

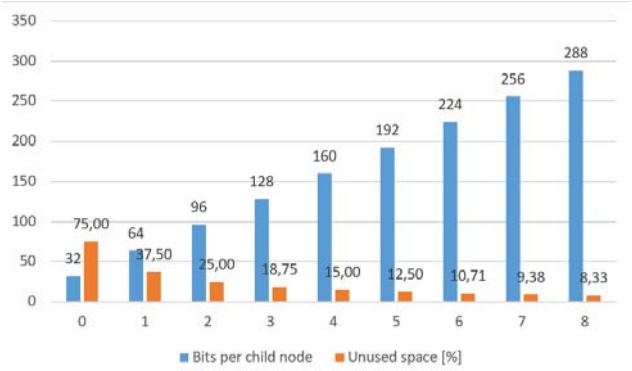

Fig. 3 Relative size of unused space in the node in comparison to the overall size of binary representation of the node with various number of child node pointers, when leaf nodes of the structure comprise no child node pointers and non-leaf nodes comprise number of child node pointers from the range of <1; 8> .

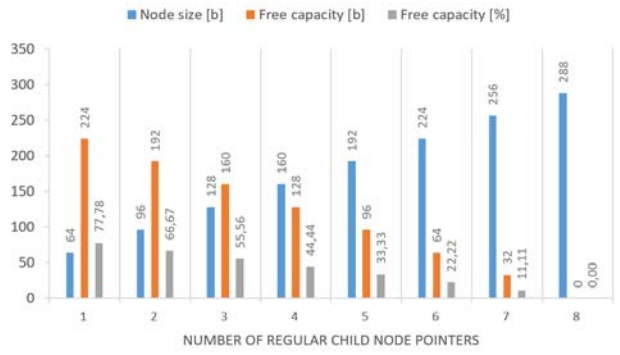

Fig. 4 Size of space for data hiding when extra child node pointers are introduced into the non-leaf node of the data structure and full capacity of extra child node pointers is used. Number of extra child nodes can vary from the range of <0; 7>.

It is possible to do analysis of the number of child pointers indicated in all child node masks of the data structure by counting of bit 1 occurrences and then count number of child node pointers in all nodes and identify discrepancies between assumed and real number of child node pointers.

Extra nodes or subDAGs. There is possibility to introduce extra nodes or even whole subDAGs which are not connected to the main DAG and therefore the extra nodes or subDAGs are not reachable. Extra nodes or subDAGs are not altering visual representation of data stored in data structure because they are not traversed in time of data structure use. Those extra nodes or subDAGs can resemble real nodes or subDAGs of the data structure and can implement all possibilities how to hide data into SVDAG. Extra nodes can be used not only for data hiding, but also for altering of statistical information about overall data structure, because there can be discrepancy, when for example there are more child pointers indicated in child node mask then are represented in the node. It can compensate extra child node pointers as it was depicted in previous method. There is strikingly big space for extra subDAGs amounting hundreds of MB because of the large addressing space that is available in SVDAG.

It is possible to do analysis of the assumed number of nodes in the data structure and their size and compare it to the real number of nodes and their size to find discrepancies.

Extra data. part of the file can be used for extra data, when structure of the data does not imitate structure of nodes or subDAGs. It is not altering visual representation of data stored in the data structure because extra data are not traversed in time of data structure use.

Extra free space. It is assumed that nodes of data structure are concatenated to each other to fill addressing space as economically as it is possible. However, there is possibility to leave free space between nodes where no information is stored. Size of the free space (32-bit aligned) and location of it can encode information even when it is regulary filled with strings of 0 bits and therefore can be used for data hiding and different encoding schemas can be proposed.

Extra data and extra free space can be identified, because as in the previous method of data hiding, it is possible to do analysis of the assumed number of nodes in the data structure and their size and compare it to the real size of the data structure to find discrepancies.

Common subtree merge. There is possibility to build different DAGs that are representing the same visual information of the 3D scene in case of possibility of common subtree merge, because it is possible to do common subtree merge or ignore this possibility independently for each pair of common subtrees. It allows different DAGs with the same geometry of the scene information. Different encoding methods can be suggested based on this possibility. Analysis can be done if all possible common subtree merges were performed, it means, if the SVDAG data structure is as compact as possible and in case there is discrepancy, it can indicate use of data hiding method.

Relative positions of nodes. There is possibility to do permutations of child node addresses in the addressing space of the data structure and their relative positions along with some features of nodes can be used for different encoding schemas for hiding information, for example:

- Affiliation of child nodes to the same/different parent node. (For example there is possibility to suggest encoding, when address of the first child node must be lower than second child node to encode bit 0 and higher than second child node to encode bit 1) for child nodes of the same parent node.
- Affiliation of nodes to the same/different level of nodes in data structure. For example, when address of parent node is lover than child node it encodes bit 0 and opposite, when child node address is lower than parent node address it encodes bit 1.
- Affiliation of child nodes to the same/different subDAGs.

Absolute position of nodes. There is possibility to use absolute value of the node address in addressing space along with features of node to encode information, when for example particular bit of the node address (for example third least significant bit) for node that has even number of child nodes means encoding of bit 0 and odd number of child nodes means encoding of bit 1.

Disadvantage of relative and absolute position of nodes data hiding methods is in relatively small capacity, when encoding methods can hide one bit of information per node with the size from the range of 32 bits to 288 bits. In that case capacity of information encoding vary from 0.347% to the 3.125% of the node size in bits.

In the case of relative and absolute positions of nodes analysis can be made in which irregularities in addressing of nodes can be analysed and identification of those irregularities can be evaluated as the sign of the use of data hiding.

5. CONCLUSIONS

The paper deals with the problematics of data hiding using steganography in hierarchical data structures that are used in the field of geometry representation of voxelized three dimensional scenes. From the family of hierarchical data structures, that includes Sparse Voxel Octrees (SVO), Sparse Voxel Directed Acyclic Graph (SVDAG) and Symmetry-Aware Sparse Voxel Directed Acyclic Graph (SSVDAG) our attention was focused on SVDAG data structure. In the first part of the paper SVDAG data structure construction was described and then several methods of data hiding were discussed along with the discussion about usability and capacity of the stegomedia in SVDAG data format in accordance to the particular data hiding methods.

In the future research we will focus our attention to the most sophisticated hierarchical data structure in this field – Symmetry-aware Sparse Voxel Directed Acyclic Graph (SSVDAG), which construction is more complex and it allows even more complex and advantageous steganography approaches in comparison to the SVDAG data structure.

ACKNOWLEDGMENTS

This research was supported by the Slovak Research and Development Agency, project number APVV-18-0214 and by KEGA Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under Grant No. 003TUKE-4/2017 Implementation of Modern Methods and Education Forms in the Area of Security of Information and Communication Technologies towards Requirements of Labor Market.

REFERENCES

[1] VOKOROKOS, L. – BALÁŽ, A. – CHOVANEC, M.: Distributed detection system of security intrusions based on partially ordered events and patterns. In: Towards Intelligent Engineering and Information Technology. Volume 243, Studies in Computational Intelligence. - Heidelberg: Springer Berlin, 2009 P. 389-403. - ISBN 9783642037368.

[2] VOKOROKOS, L. – BALÁŽ, A. – CHOVANEC, M.: Intrusion detection system using self-organizing map - 2006. In: Acta Electrotechnica et Informatica. Roč. 6, č. 1 (2006), s. 81-86. - ISSN 1335-824.

[3] CHOVANCOVÁ, E. – ÁDÁM, N. – BALÁŽ, A. – PIETRIKOVÁ, E. – FECIĽAK, P. – ŠIMOŇÁK, S. – CHOVANEC, M.: Securing distributed computer systems using an advanced sophisticated hybrid honeypot technology - 2017. In: Computing and Informatics. Roč. 36, č. 1 (2017), s. 113-139. - ISSN 1335-9150.

[4] CHOVANCOVÁ, E. – HURTUK, J.: Highly Interactive Hybrid Honeypot - 2019. In: Applied Computational Intelligence and Informatics. - Timişoara (Rumunsko): Universitatea Politehnica Timisoara s. 33-37 [online]. - ISBN 978-1-7281-0685-4.

[5] VOKOROKOS, L. – CHOVANCOVÁ, E. – RADUŠOVSKÝ, J. – CHOVANEC, M.: A Multicore Architecture Focused on Accelerating Computer Vision Computations - 2013. In: Acta Polytechnica Hungarica. Vol. 10, no. 5 (2013), p. 29-43. - ISSN 1785-8860.

[6] PETITCOLAS, F. A. P. – ANDERSON, R. J. – KUHN, M. G.: Information Hiding – A Survey. Proceedings of the IEEE, Vol. 87, 1999, No. 7, pp. 1062–1078, doi:10.1109/5.771065.

[7] ZIELIŃSKA, E. – MAZURCZYK, W. – SZCZYPIORSKI, K.: Trends in Steganography. Communications of the ACM, Vol. 57, 2014, No. 3, pp. 86–95, doi:10.1145/2566590.2566610.

[8] VOKOROKOS, L. – MADOŠ, B. – ÁDÁM, N. – BALÁŽ, A. – PORUBÄN, J. – CHOVANCOVÁ, E.: "Multi-Carrier Steganographic Algorithm Using File Fragmentation of FAT FS" - 2019. In: Computing and Informatics: Computers and Artificial Intelligence. - Bratislava (Slovensko), Ústav informatiky Roč. 38, č. 2 (2019), s. 343-366 [print]. - ISSN 1335-9150.

[9] AYCOCK, J. – DE CASTRO, D. M. N.: Permutation Steganography in FAT Filesys-tems. In: Shi, Y. (Ed.): Transactions on Data Hiding and Multimedia Security X.Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8948, 2015, pp. 92–105, doi: 10.1007/978-3-662-46739-86.

[10] LIU, S. F. – PEI, S. – Huang, X.Y. – TIAN, L.: File Hiding Based on FAT File System. Proceedings of the 2009 IEEE International Symposium on IT in Medicine and Education, Jinan, China, Vol. 1, 2009, pp. 1198–1201, doi:10.1109/ITIME.2009.5236280.

[11] JÓKAY, M. – KOŠDY, M.: Steganographic File System Based on JPEG Files. Tatra Mountains Mathematical Publications, Vol. 57, 2013, No. 1, pp. 65–83, doi: 10.2478/tmmp-2013-0036. ISSN: 1210-3195.

[12] JÓKAY, M. – KOŠDY, M. – ČAVOJ, M.: Steganographic File System Embedded in static Images. Central European Conference on Cryptology 2013, Telč, Czech Republic, 2013, pp. 76.

[13] MORTON, G. M. 1966.: A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing, Research Report. International Business Machines Corporation (IBM), Ottawa, Canada, 20.

[14] BAERT, J. – LAGAE, A. – DUTRÉ, P.: "Out-of-core construction of sparse voxel octrees". In Proceedings of the 5th High-Performance Graphics Conference. HPG '13. ACM, New York, NY, USA, 27–32.

[15] WEBBER, R.E. – DILLENCOURT, M.B.: "Compressing quadtrees via common subtree merging". Pattern Recognition Letters 9 (1989), April 1989, pp. 193–200.

[16] KER-CHANG CHANG, H. – SHING-HUA, L. – CHENG-KUAN, G.: "Two dimensional template-based encoding for linear quadtree representation"

[17] PARKER, E. – UDESHI, T.: "Exploiting self-similarity in geometry for voxel based solid modeling". In Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications. SM '03. ACM, New York, NY, USA, 157–166.

[18] LAINE, S. – KARRAS, T.: "Efficient sparse voxel octrees". In Proceedings of ACM SIGGRAPH 2010 Symposium on Interactive 3D Graphics and Games. ACM Press, New York, NY, USA, 1–9.

[19] KAMPE, V. – SINTORIN, E. – ASSARSON, U.: "High resolution sparse voxel DAGs". ACM Trans. Graph. 32, 4, Article 101 (July 2013), 13 pages. DOI: https://doi.org/10.1145/2461912.246 2024.

[20] VILLANUEVA, A. J. – MARTON, F. – GOBETTI, E.: "SSVDAGs: symmetry-aware sparse voxel DAGs". In Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '16). ACM, New York, NY, USA, 7-14. DOI: https://doi.org/10.1145/2856400.2856420.

Received December 5, 2019, accepted December 17, 2019

BIOGRAPHIES

Branislav Madoš (Ing., PhD.) was born on 20th of May 1976 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". Since 2010 he is working as an Assistant Professor at the Department of Computers and Informatics. His scientific research is focused on the parallel computer architectures and architectures of computers with data driven computational model and computer security using cryptographic and steganographic methods.

Zuzana Bilanová (Ing.) was born on 13th of July 1992 in Košice, Slovakia. In 2015 she graduated (Ing.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Since 2015 she is PhD. student at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Her main scientific orientations are focused on creating new approaches in logical analysis of natural language, non-classical logical systems in +89 computer science, and resource-oriented logic programming. Her secondary research areas are educational technologies for the effective implementation of engineering education concentrating on project and team based teaching.