Reading Volume Datasets from Storage – Using Segmentation Metadata, for an Enhanced User Experience

Branislav Madoš, Norbert Ádám

Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9/A, 042 00 Košice
e-mail: branislav.mados@tuke.sk, norbert.adam@tuke.sk

Abstract: This paper deals with the issues of volume dataset representation as an important part of data storage and processing in many fields including science, research and development, medicine or industry. Due to the significant amount of data included in volume datasets, operations performed on them are often, time- and space-consuming. One of those operations – loading data from secondary storage into the operating memory of computer or memory of graphics card – can be time-consuming and lead to a bad user experience and significantly delay the subsequent processes. Therefore, the main contribution hereof is the design and introduction of an algorithm to generate volume dataset segmentation metadata. It allows (with a small data overhead, as a trade-off) to prepare metadata about splitting the particular volume dataset into segments with different priority levels. Subsequently, it is possible to reorganize the volume dataset according to the priority of the data segments, in descending order. The algorithm proposed herein allows to start the visualization of the volume dataset in its final quality (resembling visualization of the complete volume dataset, although only a part of the data was loaded from the secondary storage), within a fraction of the total load time of the volume dataset. The remaining data are continually read in the background during data visualization, without affecting volume data visualization quality. The first section herein, contains an introduction to the proposed algorithms. Results of tests, performed with different parameter setups on non-invasive medical imaging volume datasets, obtained by computed tomography and magnetic resonance imaging, are included in the second part of the paper. Conclusions, drawn from test results, are summarized in the last part of the paper.

Keywords: volume dataset; three-dimensional image; 3D image, image segmentation algorithm; metadata; user experience; computed tomography; CT; magnetic resonance imaging; MRI

1 Introduction

Volume datasets are often represented as regular three-dimensional grids of scalar values or vectors of scalar values. Volume dataset representation, pre-processing, visualization and other forms of processing are important in the field of science, research and development [1] [2], medicine [3], industry [4] [5], etc. The amount of the space that volume datasets need for their representation in the operating memory of computer or memory of graphics card and even the secondary storage has always been a challenge. Although the capacities of computers, in terms of system memory, secondary storage size and data throughput are constantly and significantly growing, as do the storage space requirements of volume datasets – in terms of their geometrical resolutions and the size of their binary representations (the number of bits) of voxels.

Computed tomography, invented by Sir Godfrey Hounsfield in 1967, was first used to scan a patient in 1971 [6]. Back then, computed tomography produced volume datasets of 64^3 voxels (262144 voxels), taking 2.5 hours to compute. Nowadays, 512^3 voxel computed tomography scans are common: when using 12b/vox, without any compression, they need 192 MB of secondary storage space (256MB when using 16b/vox). Raising the resolution of common volume datasets to $2K^3$ $(2048 \times 2048 \times 2048)$ voxels and using 16b/vox, their size on the secondary storage and in system memory will rise to 16 GB. While in 2007, 70 million CT scans were performed in the USA alone, this number raised to 80 million in 2015 [7].

Not only the size of volume datasets itself can be significant challenges (in terms of both time and space), but also operations performed on them during their pre-processing, visualization and other forms of processing. Modern approaches to volume dataset visualization, in combination with virtual reality, augmented reality [8] [9] or computer vision [10] are even more demanding. Spatiotemporal volume datasets are even more demanding to process. One of those time-consuming operations – loading the volume dataset from secondary storage into system or graphics memory – can result even in bad user experience and can significantly delay the subsequent processes.

That is why the authors decided to work on an algorithm that can enhance the user experience, concerning the loading of the volume dataset, from secondary storage, into operating memory of computer. In [11], we designed an algorithm decomposing the volume dataset into two segments, creating a three-dimensional image – segmentation metadata – and rearranging the volume dataset to allow reading of the important segment of voxels from secondary storage preceding the unimportant segment. In comparison to the original volume dataset, the produced segmentation metadata represent a significant amount of data and that is why we applied lossless compression to the volume dataset metadata (see: Related works section of the paper).

In this work, we build on this previous research, proposing an algorithm to create volume dataset segmentation metadata, to rearrange the volume dataset and to enhance reading the volume dataset from storage.

The contribution lies in the following:

- An algorithm splitting the volume dataset into data blocks, assigning a level of priority (importance) to each data block of the volume dataset, creating volume dataset segmentation metadata to represent the information stored in the dataset and reorganizing the volume dataset (linearizing the segments, ordered by the level of priority in descending order). The segmentation metadata allow reconstruction of the original location of the voxels for each segment of the volume dataset.
- An algorithm that allows to start the visualization of the volume dataset in its final quality (resembling visualization of the complete dataset, although only a corresponding fraction of the dataset was loaded from secondary storage), within a fraction of the total load time of the dataset. The remaining data are continually read in the background, during data processing, without affecting data visualization quality.

The structure of the remainder of this paper is as follows:

Section 2 presents the related works concerning multi-dimensional data linearization, volume dataset segmentation metadata and lossless compression of those segmentation metadata using domain-specific hierarchical data structures based on octant trees and directed acyclic graphs and other lossless compression algorithms including Run-Length Encoding and ΔRLE.

Section 3 introduces the proposed algorithms: one performing volume dataset segmentation and creating the volume dataset segmentation metadata and the other, improving the user experience concerning reading the volume dataset from storage. The inputs of these algorithms, their particular steps and outputs are described in detail.

Section 4 represents the test results of the algorithms described in the previous section using various medical imaging volume datasets using CT and MRI and various parameter setups.

The *Conclusions* section summarizes the conclusions based on the tests, described in Section 4.

2 Related Works

This section mentions only very close related works, related to the linearization of the multi-dimensional data, to the enhancement of the user experience concerning reading volume datasets from secondary storage and compression of volume datasets using hierarchical data structures.

Linearization of multi-dimensional data. Space-Filling Curves (SFC), introduced by Peano and Hilbert at the end of the 19th Century [12] [13], are used for linearization not only of two- or three- but in general of multi-dimensional data. The Morton order is an SFC popular in computer graphics for its better addressing abilities [14] and Hilbert Space Filling-Curve (HSBC) is used in computer science for better locality preserving [15].

Volume dataset segmentation metadata. In [11], we designed an algorithm assigning 1b of metadata to each voxel of the volume dataset – this allows including that voxel into the background (the unimportant voxel segment) or into the region of interest (the important voxel segment) of the volume dataset. It allows rearranging the volume dataset in the manner that all voxels from the important segment are linearized in the first part of the dataset, to be read before the unimportant voxels, stored in the second part of the dataset. In each segment of the rearranged dataset, the order of voxels is the same as it is in linearized form of the original dataset. The size of metadata is 1b per voxel – if 16b are used as the size of the binary representation of voxels, metadata represent 1/16 of the volume dataset size. That is too much, which is why lossless compression of the volume dataset metadata is applied. The 3D metadata image of the volume dataset allows reconstruction of the original location of voxels, for each segment of the volume dataset.

Lossless compression of volume segmentation metadata. In connection with the above-mentioned algorithm, lossless compression of the image segmentation metadata was proposed and different Run-Length Encoding (RLE) schemes were used [16]. Then, ΔRLE – a new compression algorithm based on the combination of Delta encoding and Run-Length Encoding – was tested on the image segmentation metadata of the volume datasets.

Suitable solutions for compressing volume dataset segmentation metadata – the volume dataset itself – are octree-based Hierarchical Data Structures (HDS), also in their pointerless versions [17] [18] (these are suitable for dense volume datasets and can encode multi-bit-value voxels), and sparse, octree-derived, hierarchical data structures - directed acyclic graphs (DAGs) – e.g. Sparse Voxel Directed Acyclic Graphs (SVDAGs) [19], Symmetry-aware Sparse Voxel Directed Acyclic Graphs (SSVDAGs) [20] and Pointerless Sparse Voxel Directed Acyclic Graphs (PSVDAGs) [21]. The latter are suitable for compressing metadata if the volume dataset has only two segments e.g. one bit per voxel can be used for the geometry representation.

3 Proposed Algorithms

This section introduces the main contribution of the paper: an algorithm to create the volume dataset segmentation metadata (described in subsection 3.1) and an

algorithm to enhance the user's experience concerning reading the volume dataset from the secondary storage (described in subsection 3.2).

3.1 An Algorithm to Generate Volume Dataset Segmentation Metadata

This section describes the input, the steps of algorithm and the outputs of the proposed algorithm.

3.1.1 Input

The inputs of the proposed algorithm are as follows:

1) The volume dataset $R[X,Y,Z]$, organized as a regular three-dimensional grid of voxels, with $X,Y,Z \in \mathbb{N}$ grid dimensions. Each voxel $v[x,y,z] \in R: x \in < 0; X-1 >;\ y \in < 0; Y-1 >;\ z \in < 0; Z-1 >; x,y,z \in \mathbb{N}_0$ is represented by a scalar value $val \in < 0; v_{max} >$, where v_{max} is the maximal value.
2) The size of the volume dataset data block that is represented by its dimensions $B_x \in < 1; X >$, $B_y \in < 1; Y >$, $B_z \in < 1; Z >$; $B_x, B_y, B_z \in \mathbb{N}$.
3) Number of priority levels $\rho_{max} \in \mathbb{N}$.

3.1.2 Steps

The following five steps of the algorithm are performed consecutively:

Step 1 of the algorithm divides the volume dataset $R[X,Y,Z]$ into a regular three-dimensional grid $R'[X',Y',Z']$ of voxel data blocks (from the volume dataset); $X' = \left\lceil \frac{X}{B_x} \right\rceil, Y' = \left\lceil \frac{Y}{B_y} \right\rceil, Z' = \left\lceil \frac{Z}{B_z} \right\rceil; X',Y',Z' \in \mathbb{N}$ are the dimensions of the grid. Each data block $[x',y',z'] \in R': x' \in < 0; X'-1 >;\ y' \in < 0; Y'-1 >;\ z' \in < 0; Z'-1 >; x',y',z' \in \mathbb{N}_0$.

Each volume dataset data block has a size B_{size} determined by its dimensions B_x, B_y and B_z, expressed in number of voxels. This can be calculated using the following formula:

$$B_{size} = B_x * B_y * B_z \tag{1}$$

The data block count N (i.e. the number of data blocks, to which the volume dataset is divided), can be calculated as follows:

$$N = \left\lceil \frac{X}{B_x} \right\rceil * \left\lceil \frac{Y}{B_y} \right\rceil * \left\lceil \frac{Z}{B_z} \right\rceil \tag{2}$$

Step 2 To each data block $\delta[x',y',z'] \in R'[X',Y',Z']$ of the volume dataset, its priority level is assigned from the set P with the cardinality $|\rho_{max}|$, when this set is defined as

$$\mathrm{P} = \{0,1,2,\ldots,\rho_{max}-3,\rho_{max}-2,\rho_{max}-1\} \tag{3}$$

$R''[X'',Y'',Z'']$, where $X'' = X', Y'' = Y', Z'' = Z'; X'', Y'', Z'' \in \mathbb{N}$, is the regular three-dimensional grid of scalar values, each of these represents the priority level of the particular data block in the volume dataset. Priority level 0 is assigned to the data blocks having the highest priority, while priority level $\rho_{max}-1$ is assigned to the data blocks having the lowest priority in the volume dataset.

The above-mentioned assignment is performed using function f (this function can be designed according to the specific needs of the particular volume dataset segmentation):

$$R''[X'',Y'',Z''] = f\colon (R'[X',Y',Z']) \rightarrow \mathrm{P} \tag{4}$$

The priority level of each volume dataset data block is encoded in binary. If fixed-length encoding in the form of unsigned integer values is used, the encoding requires $\lceil \log_2 \rho_{max} \rceil$ bits per data block. The overall number of volume dataset data blocks is quantified using (2) as N, so the overall size of $R''[X'',Y'',Z'']$, the R''_{size} value, in bits, is:

$$R''_{size} = N \times \lceil \log_2 \rho_{max} \rceil \, [b] \tag{5}$$

All data blocks constituting R' and having the same particular priority level $p \in P; p \in < 0; \rho_{max}-1 >; p \in \mathbb{N}_0$ assigned in R'', constitute segment S_p of that particular priority in the volume dataset. The set of segments Σ with the cardinality $|\rho_{max}|$ may be described as follows:

$$\Sigma = \{S_0, S_1, S_2, \ldots, S_{\rho_{max}-3}, S_{\rho_{max}-2}, S_{\rho_{max}-1}\} \tag{6}$$

Each data block from R' belongs to one of the above segments and therefore:

$$R'[X',Y',Z'] = \bigcup_{r=0}^{\rho_{max}-1} S_r \tag{7}$$

Step 3 Volume dataset R' is linearized, when all of its data blocks are ordered in a one-dimensional stream according to the selected linearization.

Step 4 Volume dataset R' is rearranged, when its data segments are ordered according to the priority in the linearized one-dimensional stream of data blocks, in descending order. In each volume dataset segment, the data blocks are ordered according their order in the linearized volume dataset obtained in ***Step 3*** of the algorithm.

Step 5 Voxels of volume dataset R' are linearized separately in each volume dataset data block according to selected linearization.

3.1.3 Outputs

The output of the algorithm is represented by the volume dataset segmentation metadata stored in R'' and by the linearized rearranged volume dataset R' obtained in ***Step 5***.

3.1.4 Metadata Generation and Volume Dataset Rearrangement – an Example

Figure 1 shows the steps of the proposed algorithm – for the sake of simplicity, using two-dimensions. Figure 1a contains a grid R' of 8×8 pixels.

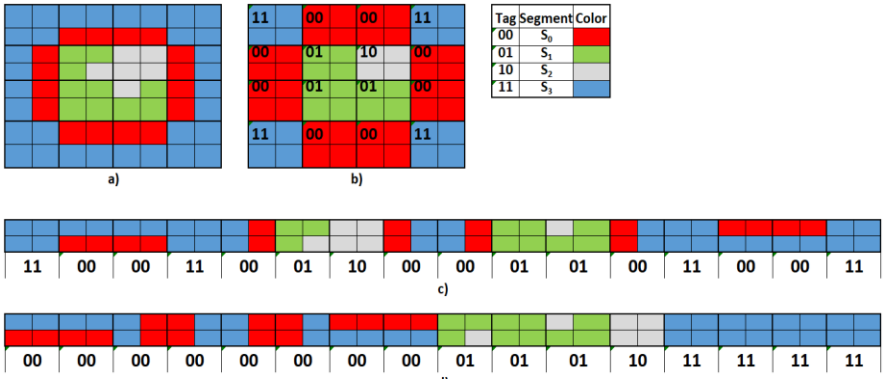


Figure 1
A two-dimensional (for the sake of the simplicity) example of the steps of the algorithm generating volume metadata and rearranging dataset

In ***Step 1,*** as shown in Figure 1b, the grid is divided into 16 data blocks forming the 4×4 grid R''. Each data block has 2×2 pixels.

In ***Step 2***, each data block is assigned a priority from the set $P = \{p_0, p_1, p_2, p_3\}$. Priority values are encoded in binary as the vectors

$$p_0 = \text{"00"}, p_1 = \text{"01"}, p_2 = \text{"10"}, p_3 = \text{"11"}$$

and also in color – see the color coding table in Figure 1.

Data blocks constitute the segments $\Sigma = \{S_0, S_1, S_2, S_3\}$ of the volume dataset.

In ***Step 3,*** the data blocks are linearized as shown in Figure 1c. The linearized sequence of the binary priority codes shown in Figure 1c constitutes the linearized form of the segmentation metadata in the algorithm output.

In ***Step 4,*** the data blocks are rearranged (Figure 1d), when all data blocks from the same segment $S_r : r \in < 0; 3 >$ are located, within the segment, in the order of appearance in the linearized form (Figure 1c). The segments are ordered according to their priority levels, in descending order.

Step 5, in which each data block is linearized according the selected linearization, is missing from Figure 1. The rearranged form of the data blocks is the output of the algorithm.

It is possible to reconstruct the original position of each voxel in the volume dataset for each data block loaded from the secondary storage, using rearranged volume dataset and metadata (volume segmentation metadata R''), even if the volume dataset is loaded only partially, i.e. if only some segments of the dataset are loaded from the secondary storage.

In ***Step 1***, volume segmentation metadata is transformed from the linearized form into the three-dimensional grid R'', respecting selected linearization.

In ***Step 2***, volume dataset data blocks forming particular priority segment are placed into the locations of particular priority codes respecting order of volume dataset data blocks in the priority segment.

The number of data blocks in each segment can be obtained as the result of analysis of segmentation metadata R'' in time of original volume dataset reconstruction from rearranged volume dataset and therefore can be omitted from the metadata representation.

3.2 An Algorithm for Improved Reading of Volume Datasets from Storage

The algorithm is based on the above-mentioned algorithm, when all its steps have been performed, using a specific function f for the assignment of the priority level to volume dataset data blocks, on a pre-processed volume dataset.

3.2.1 Input

The proposed algorithm has the following inputs:

1) The volume dataset $R[X, Y, Z]$, organized as a regular three-dimensional grid of voxels, with $X, Y, Z \in \mathbb{N}$ grid dimensions. Each voxel $v[x, y, z] \in R: x \in <0; X-1>;\ y \in <0; Y-1>;\ z \in <0; Z-1>; x, y, z \in \mathbb{N}_0$ is represented by a scalar value $val \in <0; v_{max}>$ where v_{max} is the maximal value.
2) The size of the volume dataset data block that is represented by its dimensions $B_x \in <1; X>$, $B_y \in <1; Y>$, $B_z \in <1; Z>$; $B_x, B_y, B_z \in \mathbb{N}$
3) Number of priority levels $\rho_{max} = 4$
4) Threshold $\tau \in <0; v_{max}>$

3.2.2 Steps

Steps P1 and P2 are the pre-processing steps of the algorithm; after these, all five steps of the volume dataset segmentation and metadata generation algorithm are performed, applying function f designed specifically for this algorithm.

Step P1. This step decomposes the volume dataset $R[X,Y,Z]$ (a slice of example volume dataset is shown in Figure 2a) into two segments (unimportant, i.e. background and important, i.e. foreground voxels) and creates a three-dimensional grid $R_s[X,Y,Z]$. As a proof of concept, the authors used a segmentation threshold value τ (that is why τ is one of the inputs of the algorithm):

$$R_s[x,y,z] = \begin{cases} 0 \; if \; R[x,y,z] < \tau \\ 1 \; if \; R[x,y,z] \geq \tau \end{cases} \tag{8}$$

Two segments T_0 and T_1 are formed, when:

$$R[x,y,z] \in T_0 : R_s[x,y,z] = 0$$
$$R[x,y,z] \in T_1 : R_s[x,y,z] = 1 \tag{9}$$

Each volume dataset voxel from R belongs to one of those segments and therefore:

$$R[X,Y,Z] = T_0 \cup T_1 \tag{10}$$

After the ***Step P1*** of the algorithm is performed:

- Segment T_0 contains each voxel of R, having a value lower than the threshold τ (the corresponding value is set to 0 in R_s). In Figure 2b, these are displayed in grey.
- Segment T_1 contains each voxel of R, having a value equal or greater than the threshold τ (the corresponding value is set to 1 in R_s). In Figure 2b, these are displayed in green.

Step P2. The volume dataset is segmented further, into set Φ of four segments U_0, U_1, U_2 and U_3:

$$\Phi = \{U_0, U_1, U_2, U_3\} \tag{11}$$

Each volume dataset voxel from R belongs to one of those segments, therefore:

$$R[X,Y,Z] = \bigcup_{r=0}^{3} U_r \tag{12}$$

After ***Step P2***:

- Segment U_0 contains all voxels of the surface of the region of interest (foreground). In Figure 2c, these are displayed in red.

- Segment U_1 contains all important voxels within the region of interest that are not constituting its surface. In Figure 2c, these are displayed in green.
- Segment U_2 contains all unimportant voxels within the region of interest. In Figure 2c, these are displayed in grey.
- Segment U_3 contains all unimportant voxels beyond the region of interest (background voxels). In Figure 2c, these are displayed in blue.

In the proof of concept, the authors used flood fill to implement this step.

Step P2a. Voxels constituting the borders of the volume dataset grid are examined, each voxel $v[x, y, 0] \in R$, $v[x, y, Z-1] \in R$, $v[x, 0, z] \in R$, $v[x, Y-1, z] \in R$, $v[0, y, z] \in R$, $v[X-1, y, z] \in R$ is added to segment U_3, if this voxel is a part of segment T_0. In that case, this voxel is used also as the start point of the further flood fill (***Step 2b***). If this voxel belongs to segment T_1, it is added to segment U_0.

Step P2b. Flood fill is performed starting from each voxel that was determined as the starting point in ***Step 2a*** and each voxel reached and belonging to segment T_0, not yet added to segment U_3, is added to segment U_3 and used for the further flood fill (***Step 2b***). Each reached voxel from segment T_1, not a member of segment U_0, is added to this segment (it is not used in further flood fill). Flood fill ends when all reached voxels are members of the U_0 or U_3 segments and there are no more starting points for further flood fill available.

Step P2c. All voxels from the dataset R that are members of segment T_0 and were not added to segment U_3 are now added to segment U_2. All voxels from the dataset R that are members of segment T_1 and were not added to segment U_0 are now added to segment U_1.

Step 1 is identical to ***Step 1*** of above-mentioned algorithm (Subsection 3.1.4).

Step 2 is identical to ***Step 2*** of above-mentioned algorithm, with four segments: S_0, S_1, S_2 and S_3.

Function f performs the following mapping:

- Segment S_0 represents the set of surface data blocks of the region of interest. Each data block in this set contains at least one voxel belonging to the U_0 surface voxel set. The assigned priority level is 0. In Figure 2d, these data blocks are displayed in red.
- Segment S_1 represents the set of important data blocks of the region of interest. Each data block in this set contains at least one voxel belonging to the U_1 important voxel set and can contain voxels from segment U_2 but does not contain any voxels from the U_0 surface voxel set nor from the U_3 unimportant voxel set. The assigned priority level is 1. In Figure 2d, these data blocks are displayed in green.

- Segment S_2 represents the set of unimportant data blocks within the region of interest. Each data block is homogenously filled with voxels from U_2, the set of unimportant voxels of the region of interest. The assigned priority level is 2. In Figure 2d, these data blocks are displayed in grey.
- Segment S_3 represents the set of unimportant data blocks beyond the region of interest (the background). Each data block is homogenously filled with voxels from U_3, the set of unimportant voxels beyond the region of interest. The assigned priority level is 3. In Figure 2d, these data blocks are displayed in blue.

Each data block is assigned a 2b priority tag: data blocks from the S_0 segment are tagged with "00", from the segment S_1 with "01", from the S_2 with "10" and from segment S_3 with "11". The cardinality – the number of data blocks in the particular set – of each segment S_0, S_1, S_2 and S_3 is evaluated.

Step 3, ***Step 4*** and ***Step 5*** of the algorithm are identical to the corresponding steps of the algorithm described in subsection 3.1.

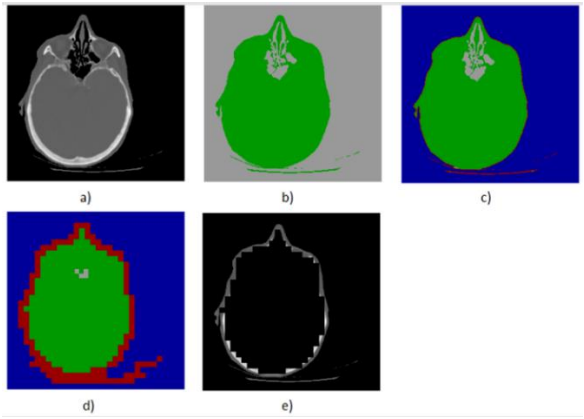


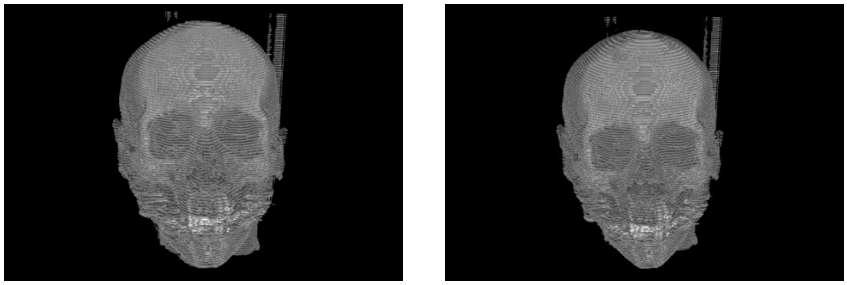
Figure 2

Visualization of algorithm steps: a) slice of the original data; b) slice of binarized data; c) slice of data with four segments of voxels; d) slice of dataset split into volume data blocks of volume segments; e) visualization of data blocks constituting segment S_0

3.2.3 Output

The output of the algorithm is represented by the volume dataset segmentation metadata stored in R'' and by the linearized rearranged volume dataset R' obtained in ***Step 5*** of the algorithm. The enhancement of reading volume datasets from storage lies in the possibility to load the volume dataset segmentation metadata along with the segment S_0 of the dataset, which allows to start the visualization of this segment in 3D immediately – with full possibility of interactions and transformations – while loading the remaining, less important segments on the

background, according to the set priority level. It allows to see surface of all parts of the region of interest in the final quality after only a fraction of the time needed to load the whole volume dataset (see Figure 3).



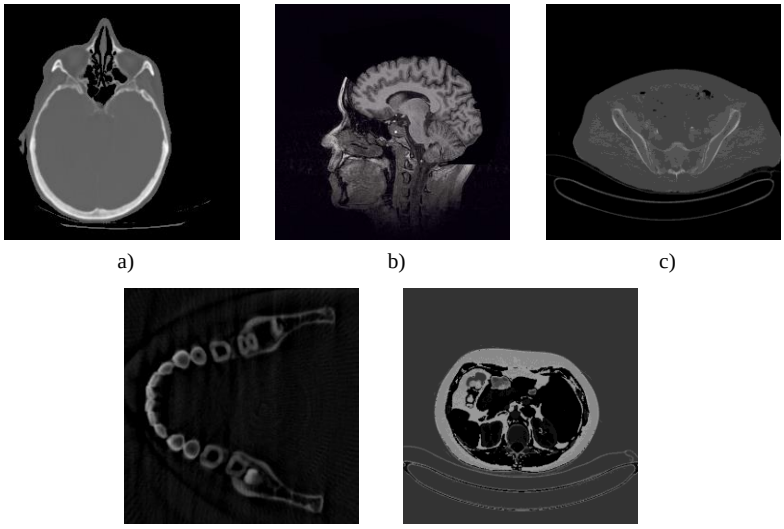
a) b)

Figure 3

Visualization of volume dataset a) only S_0 segment b) all segments

4 Test Results

Tests were performed on volume datasets obtained using medical imaging techniques, including Computed Tomography and Magnetic Resonance Imaging– see the volume dataset parameters in Table 1 and slice visualizations in Figure 4.



a) b) c)

d) e)

Figure 4

Slices of volume datasets obtained using different medical imaging techniques including Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) that were used for testing purposes: a) Head, b) Brain, c) Abdomen, d) Human Skull, e) Pancreas

Voxels of these volume datasets were represented as scalar values. Some volume datasets were encoded as 8-bit unsigned integers per voxel (range: <0;255>), others were encoded as 16-bit unsigned integers per voxel (range: <0;4095>) (using 12 bits for the value and keeping 4 bits reserved).

For test purposes, the authors used cubic data blocks of $n^3; n \in \mathbb{N}$ voxels. Four different data block sizes were selected: $2^3 = 8$, $4^3 = 64$, $8^3 = 512$ and $16^3 = 4096$ voxels, respectively.

Tests were performed on a computer with a four-core Intel® Core i5-8265U @ 1.6 GHz CPU, 8 GB system memory, an NVIDIA GeForce GTX 1050 3 GB graphics card and a 256 GB SSD as the secondary storage.

Table 1

Summary of parameters of the tested volume datasets obtained by the Computed Tomography (CT) and Magnetic Resonance Imaging (MRI)

	Dataset	Dimensions	Voxels [M]	b/vox	Size [MB]	Threshold	Active voxels [mil] / [pct]
A	**Head**	256 × 256 × 112	7.34	16	14.0	200	2.437 (33.21%)
B	**Brain**	256 × 256 × 96	6.29	16	12.0	1150	1.670 (26.54%)
C	**Abdomen**	512 × 512 × 160	41.94	16	80.0	200	21.646 (51.61%)
D	**Human skull**	256 × 256 × 256	16.78	8	16.0	25	2.199 (13.11%)
E	**Pancreas**	240 × 512 × 512	62.91	16	120.0	1150	57.114 (90.78%)

In all dataset tests, when only data blocks of the S_0 segment – the surface data blocks of the region of interest – of the volume datasets were loaded from secondary storage and the size of data block was 2^3 voxels, only a fraction of data blocks – ranging from 2.96% (in case of dataset e) to 15.47% (in case of dataset d) – had to be read. Considering also metadata, the share of data needed to load rose, ranging from 4.52% (in case of dataset e) to 18.60% (in case of dataset d) (see Table 2 and Figure 5).

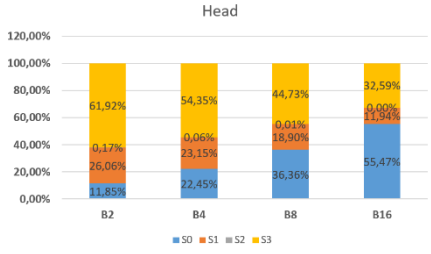
In general, an increase in the data block size generates – compared to the total voxel count – an increased share of voxels from the S_0 data block set and a decreased share of voxels from the S_1, S_2 and S_3 data block sets. For example, in dataset a, the share of S_0 raised from 11.85% (using 2^3 voxels pre data block) to 55.47% (using 16^3 voxels per data block). The total overhead (image segmentation metadata) is dependent on the data block size. In the tests, this ranged from 0.003% (using 16^3 data block size) to 1.563% (using 2^3 data block size) with datasets encoded using 16 bits per voxel and from 0.006% to 3.125% in case of datasets encoded using 8 bits per voxel.

The share of the S_0 segment of the volume dataset and the share of metadata are inversely proportionate. That is why there is a trade-off between those two parameters. Increasing the metadata share to 1.563% and/or 3.125% allows a significant reduction of the share of the S_0 dataset segment, leading to an overall improvement of the user experience.

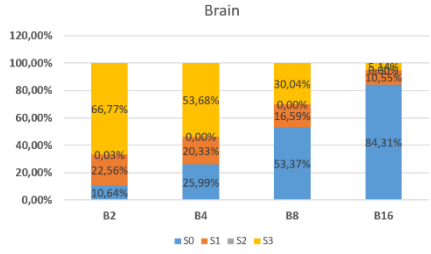
Table 2

Test results where image segmentation metadata were created for five volume datasets and four data block sizes, ranging from 2^3 to 16^3 voxels. S_0 is the segment of surface blocks, S_1 is the segment of important data blocks within the region of interest, S_2 is the segment of unimportant data blocks within the region of interest and S_3 is the segment of unimportant blocks of background data

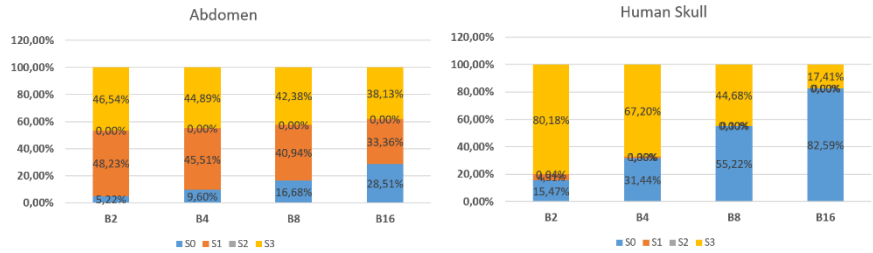
Block size	S_0		S_1		S_2		S_3		Blocks total	Metadata	
	Blocks	%	Blocks	%	Blocks	%	Blocks	%		[KB]	[%]
a – Head											
2^3	108769	11.85	239105	26.06	1536	0.17	568094	61.92	917504	224.00	1.563
4^3	25745	22.45	26546	23.15	69	0.06	62328	54.35	114688	28.00	0.195
8^3	5213	36.36	2709	18.90	2	0.01	6412	44.73	14336	3.50	0.024
16^3	994	55.47	214	11.94	0	0.00	584	32.59	1792	0.44	0.003
b – Brain											
2^3	83652	10.64	177408	22.56	261	0.03	525111	66.77	786432	192.00	1.563
4^3	25546	25.99	19989	20.33	0	0.00	52769	53.68	98304	24.00	0.195
8^3	6558	53.37	2039	16.59	0	0.00	3 691	30.04	12288	3.00	0.024
16^3	1295	84.31	162	10.55	0	0.00	79	5.14	1536	0.38	0.003
c – Abdomen											
2^3	273824	5.22	2528784	48.23	54	0.00	2440218	46.54	5242880	1280.00	1.563
4^3	62901	9.60	298245	45.51	0	0.00	294214	44.89	655360	160.00	0.195
8^3	13664	16.68	33541	40.94	0	0.00	34715	42.38	81920	20.00	0.024
16^3	2919	28.51	3416	33.36	0	0.00	3905	38.13	10240	2.50	0.003
d - Human skull											
2^3	324435	15.47	90337	4.31	835	0.04	1681545	80.18	2097152	512.00	3.125
4^3	82412	31.44	3574	1.36	10	0.00	176148	67.20	262144	64.00	0.391
8^3	18093	55.22	34	0.10	0	0.00	14641	44.68	32768	8.00	0.049
16^3	3383	82.59	0	0.00	0	0.00	713	17.41	4096	1.00	0.006
e – Pancreas											
2^3	232906	2.96	6970321	88.63	15	0.00	661078	8.41	7864320	1920.00	1.563
4^3	60037	6.11	843948	85.85	0	0.00	79055	8.04	983040	240.00	0.195
8^3	14979	12.19	98721	80.34	0	0.00	9180	7.47	122880	30.00	0.024
16^3	3616	23.54	10784	70.21	0	0.00	960	6.25	15360	3.75	0.003



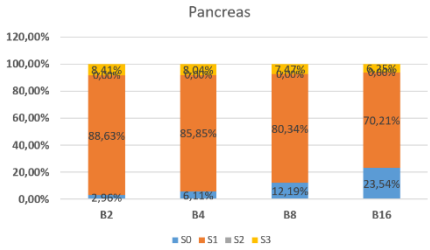
a)



b)



c) d)



e)

Figure 5

Shares of the respective segments of the corresponding volume datasets and block sizes, where the number of voxels in the block is: $B2 = 2^3$ voxels, $B4 = 4^3$ voxels, $B8 = 8^3$ voxels and $B16 = 16^3$ voxels

Tests show that, in most optimistic case, the user can start working with the volume dataset after only 4.52% of the total time required to load the whole volume dataset from secondary storage, when image segmentation metadata and all S_0 data blocks are loaded.

Table 3

Share of data blocks, when loading only segment S_0, both S_0 and S_1 and all of S_0, S_1 and S_2, respectively, for the corresponding volume datasets and data block sizes. Loading all segments ($S_0 \cup S_1 \cup S_2 \cup S_3$) represents 100% of the dataset

Block size	S_0 [%]	$S_0 \cup S_1$ [%]	$S_0 \cup S_1 \cup S_2$ [%]
a – Head			
2^3	11.85%	37.92%	38.08%
4^3	22.45%	45.59%	45.65%
8^3	36.36%	55.26%	55.27%
16^3	55.47%	67.41%	67.41%
b-Brain			
2^3	10.64%	33.20%	33.23%
4^3	25.99%	46.32%	46.32%
8^3	53.37%	69.96%	69.96%
16^3	84.31%	94.86%	94.86%

Block size	S_0 [%]	$S_0 \cup S_1$ [%]	$S_0 \cup S_1 \cup S_2$ [%]
c – Abdomen			
2^3	5.22%	53.46%	53.46%
4^3	9.60%	55.11%	55.11%
8^3	16.68%	57.62%	57.62%
16^3	28.51%	61.87%	61.87%
d - Human skull			
2^3	15.47%	19.78%	19.82%
4^3	31.44%	32.80%	32.80%
8^3	55.22%	55.32%	55.32%
16^3	82.59%	82.59%	82.59%
e – Pancreas			
2^3	2.96%	91.59%	91.59%
4^3	6.11%	91.96%	91.96%
8^3	12.19%	92.53%	92.53%
16^3	23.54%	93.75%	93.75%

We may assume that using the same volume dataset at a higher resolution and using the same data block size, the share of the S_0 segment – in terms of both data blocks and voxels, compared to all data blocks and voxels – will be smaller. That will result in a shorter load time of the S_0 segment, compared to the load time of the total volume dataset.

Another possible use case allows loading of the S_0, $S_1$and S_2 segments of the volume dataset, excluding the S_3 segment (the unimportant data outside the region of interest). This is useful when the system or the graphics card is short on memory or to shorten the total load time of the volume dataset. In the performed tests, the load time of the S_0, $S_1$and S_2 segments of the volume datasets using 2^3 data block size ranged from 19.82% (in case of dataset d) to 91.59% (in case of dataset e). Considering also metadata, the values range from 22.95% (in case of dataset d) to 93.15% (in case of dataset e).

Conclusions

This paper examined the issues of volume data representation. Due to the large amount of data included in volume datasets, many operations, including loading into operating memory of computer or graphics card memory, can be time consuming and lead to a negative user experience. For this reason, the algorithm generating segmentation metadata and subsequently rearranging the volume dataset, was developed. It improves the user experience, concerning loading the data from secondary storage, into the operating memory of the computer or the graphics card memory.

In performed tests, it allowed for work to begin on the volume dataset, in a fraction (4.52% – 18.60%) of the total time required to load the whole volume

dataset into operating memory, giving the impression of loading the complete volume dataset. However, this comes with a trade-off – a 1.563% and 3.125% data overhead, respectively. The algorithm allows for the stepwise loading of the volume dataset – each step represents the loading of a less important portion (a segment having lower priority) of the volume dataset. This process can be interrupted after each step. Therefore, loading unimportant data can be omitted, decreasing the total load time. The time required, for loading the S_0, $S_1$and S_2 segments, using 2^3 data block size, ranged from 19.82% to 91.59% in the performed tests. These values rose from 22.95% to 93.15%, when also considering metadata.

In future research, we will focus on the structure and encoding of the volume dataset segmentation metadata. The constant yet relatively high ratio of metadata size and data block size, can be significantly decreased, using lossless compression. Hierarchical data structures may also be used, albeit, their potential, has yet to be investigated. Smaller metadata size can contribute to further minimization of the load time of the first segments of the volume datasets.

Acknowledgement

This research was supported by the Slovak Research and Development Agency, project number APVV-18-0214. The volume datasets are courtesy of the following: CT Cadaver Head and MR Brain – The University of North Carolina Volume Rendering Test Data Set; Abdomen – Michael Meißner, Viatronix Inc., Human skull – Siemens Medical Solutions; Pancreas – DeepOrgan: Multi-level Deep Convolutional Networks for Automated Pancreas Segmentation.

References

[1] L. Főző, R. Andoga, L. Madarász, Mathematical model of a small Turbojet Engine MPM-20. In: Studies in Computational Intelligence Vol. 313: International Symposium of Hungarian Researchers on Computational Intelligence and Informatics. - Heidelberg: Springer, 2010, pp. 313-322 - ISBN 978-3-642-15220-7 - ISSN 1860-949X

[2] L. Nyulaszi, R. Andoga, P. Butka, et al., Fault Detection and Isolation of an Aircraft Turbojet Engine Using a Multi-Sensor Network and Multiple Model Approach, In. Acta Polytechnica Hungarica Vol. 1, No. 2, pp. 189-209, 2018, DOI: 10.12700/APH.15.1.2018.2.10

[3] P. Varga, M. Schnitzer, M. Trebuňová, R. Hudák and J. Živčák, Overview of the Current Methods for Reduction of Artifacts in CT and MR Imaging for Implants made by Additive Manufacturing, In: Acta Tecnología: International Scientific Journal about Technologies. - Šemša (Slovakia), Vol. 6, No. 2 (2020) pp. 55-58 - ISSN 2453-675X

[4] R. Andoga, L. Főző, R. Kovács, K. Beneda, T. Moravec, M. Schreiner, Robust Control of Small Turbojet Engines. Machines 2019, 7, 3, https://doi.org/10.3390/machines7010003

[5] S. Grys, L. Vokorokos and L. Borowik, Size determination of subsurface defect by active thermography – Simulation research, In: Infrared Physics & Technology. Vol. 62 (2014), pp. 147-153 - ISSN 1350-4495

[6] C. Richmond (2004) Obituary – Sir Godfrey Hounsfield, BMJ. 329 (7467): 687, doi:10.1136/bmj.329.7467.687

[7] A. Berrington de González, M. Mahesh, KP. Kim, M. Bhargavan, R. Lewis, F. Mettler and C. Land, (December 2009) Projected cancer risks from computed tomographic scans performed in the United States in 2007, Arch. Intern. Med. 169 (22): 2071-7

[8] B. Sobota and M. Guzan, Virtualization of Chua's Circuit State Space. In: Recent Advances in Chaotic Systems and Synchronization: From Theory to Real World Applications. - London, Great Britain : Elsevier Science pp. 127-164 [print] - ISBN 978-0-12-815838-8

[9] Zs. Racz, B. Sobota and M. Guzan, Parallelizing Boundary Surface Computation of Chua's Circuit - 2017. In: RADIOELEKTRONIKA 2017 - Danvers : IEEE, 2017, pp. 1-4 - ISBN 978-1-5090-4592-1

[10] L. Vokorokos, E. Chovancová, J. Radušovský and M. Chovanec, A Multicore Architecture Focused on Accelerating Computer Vision Computations - 2013. In: Acta Polytechnica Hungarica. Vol. 10, No. 5 (2013) pp. 29-43 - ISSN 1785-8860

[11] B. Madoš, A. Baláž, N. Ádám, J. Hurtuk and Z. Bilanová, Algorithm Design for User Experience Enhancement of Volume Dataset Reading from Storage Using 3D Binary Image as the Metadata - 2019. In: SAMI 2019 : IEEE 17th World Symposium on Applied Machine Intelligence and Informatics. - Danvers (USA) : Institute of Electrical and Electronics Engineers pp. 269-274 [print, online] - ISBN 978-1-7281-0249-8

[12] A. Laszloffy, J. Long and A. K. Patra, Simple data management, scheduling and solution strategies for managing the irregularities in parallel adaptive finite element simulations. Parallel Computing, 26, ISSN 1765-1788

[13] H. Sagan, Space-Filling Curves, Springer Verlag, 1994, eBook ISSN 978-1-4612-0871-6, ISBN 978-0-387-94265-0, DOI 10.1007/978-1-4612-0871-6

[14] G. M. Morton, A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing, Research Report. International Business Machines Corporation (IBM), Ottawa, Canada, 20, pp. 20, March 1st, 1966. Available: https://dominoweb.draco.res.ibm.com/reports/Morton1966.pdf

[15] D. Hilbert, Via the continuous mapping of a line onto a patch of area. Mathematical annals (orig. Über die stetige Abbildung einer Linie auf ein Flächenstück. Mathematische Annalen) 38 (1891), pp. 459-460

[16] B. Madoš and N. Ádám, Evaluation of Encoding Schemas for Optimization of Bit-Level Run-Length Encoding Within Lossless Compression of Binary Images - 2019. In: Intelligent Engineering Systems. - Budapest (Hungary): IEEE Industrial Electronics Society pp. 75-80 - ISBN 978-1-7281-1212-1

[17] B. Madoš, E. Chovancová and M. Hasin, Evaluation of Pointerless Sparse Voxel Octrees Encoding Schemes Using Huffman Encoding for Dense Volume Datasets Storage, In: ICETA 2020: 18th IEEE International conference on emerging elearning technologies and applications- Denver (USA) : Institute of Electrical and Electronics Engineers pp. 424-430, ISBN 978-0-7381-2366-0

[18] B. Madoš, N. Ádám and M. Štancel, Representation of Dense Volume Datasets Using Pointerless Sparse Voxel Octrees with Variable and Fixed-Length Encoding, IEEE 19th World Symposium on Applied Machine Intelligence and Informatics, SAMI 2021, Herľany, Slovakia, Jan., 21-23, 2021, p. 6

[19] V. Kämpe, E. Sintorn, and U. Assarsson, High Resolution Sparse Voxel DAGs. ACM Transactions on Graphics. 32, 4, Article 101 (July 2013) p. 8, ISSN 0730-0301, DOI: https://doi.org/10.1145/2461912.246 2024

[20] A. J. Villanueva, F. Marton, and E. Gobbetti, SSVDAGs: Symmetry-aware Sparse Voxel DAGs. In Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '16) February 27-28 2016, Redmond, WA, USA, pp. 7-14, ACM, New York, NY, USA. ISBN: 978-1-4503-4043-4/16/03, DOI: https://doi.org/10.1145/2856400.2856420

[21] L. Vokorokos, B. Madoš and Z. Bilanová, PSVDAG: Compact Voxelized Representation of 3D Scenes Using Pointerless Sparse Voxel Directed Acyclic Graphs", In: Computing and Informatics: Computers and Artificial Intelligence. - Bratislava (Slovakia), Vol. 39, No. 3 (2020), pp. 587-616 [print] - ISSN 1335-9150