Relazione tecnica permanenza all'estero programma Short Term Mobility 2016

Titolo:	Time-variant Direct Volume Rendering
Fruitore:	Fabio Ganovelli Mat. 9395
Ente Ospitante:	Trinity College of Dublin
periodo:	2-23 Giugno 2017

Abstract

My activity at the Trinity College of Dublin concerned, as explained in the STM proposal, direct volume rendering, a field of Computer Graphics where the hosting institution has been developing a solid reputation in the last decade. Together with dr. Dingliana, we investigated the problem of data compression for large volumes of data and designed and implemented a novel approach that, from the tests performed so far, outperforms the current state of the art.

1 Problem statement

In the field of Volume Rendering, the word *volume* most commonly refers to a three-dimensional grid of scalar values. In the case shown in Figure 1, the scalar data represents the absorption of an X-ray beam in a Computerized Axial Tomography (CAT). The figure shows a screenshot of a typical volume rendering application [3] where the volume of data is shown both as a sections along the three axis, hence as a collection of 2D images, and as a 3D rendering, named Direct Volume Rendering.

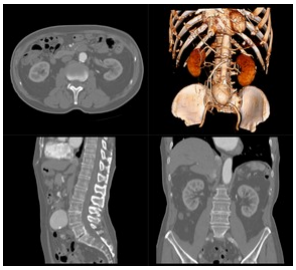

Figure 1: An example of slice-based (top let, bottom left and bottom right) and 3D direct volume rendering (top right)

In the example above the amount of absorption of X-rays allows us to distinguish between skin layers, fat tissues, muscles and so on. In other words it is possible to discriminate between these parts only by looking to the scalar values in the 3D grid (and other derivate quantities). The process of rendering the whole volume in 3D requires to decide what part of the volume is to be seen and how (bones and kidneys in the example). The most essential tool to this task is the use of the so called *transfer functions*. In its most common definition, a transfer function is a mapping between scalar values and color and opacity value, that is $tf : \mathbb{R} \rightarrow RGB\alpha$ (see Figure 2). For example, defining a transfer function that maps scalar values corresponding to bone to a high opacity value and all other scalar values to a 0 opacity (transparent). The result will be a rendering that only shows the bones. In the example above the transfer function is designed so to show bones and kidneys and to distinguish them mapping their corresponding ranges of scalar values to different color (more reddish for the kidneys).

Direct Volume rendering (DVR) is nowadays the standard approach for interactively exploring rectilinear scalar volumes and it is used in several contexts ranging from medical imaging to fluid simulation analysis, where the grid are often tetra-dimensional since the scalar value is time dependent (e.g. the local velocity of a fluid).

Even though the past several years witnessed great advancements in commodity graphics hardware, long data transfer times and GPU memory size limitations are often the main limiting factors, especially for massive, time-varying, or multi-volume visualization in both local and networked settings. As it happened in so many fields of computer science, the amount of available data increases much faster than the hardware performances. Therefore data compression is of great importance to save storage space and bandwidth at all stages of the processing and rendering pipelines.

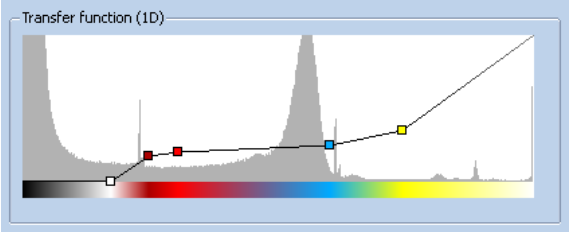

Figure 2: An example of one dimensional transfer function. The piecewise curve and the color bar show the RGBα color value assigned to each scalar value. The histogram shows the distribution of scalar values for a specific dataset

1.1 Volume Data Compression

The literature on volume data compression is vast and even an introduction is well beyond the scope of the report (the interested reader may refer to [2]. In general terms, the tool-set is similar to the one for 2D images, ranging from the simple run length encoding to frequency domain techniques (Discrete Fourier Transform, wavelet) combined with hierarchical/multiresolution subdivision schemes. This makes sense, since a volume can be seen just like a stack of 2D images. My research activity has been focused precisely on what makes DVR of volume data and viewing of 2D images different: the use of transfer functions.

2 Method: Using Transfer Functions in Volume Data Compression

Transfer functions are an interpretation tool that convert the information in a form that can be more easily interpreted by humans. They are always present and they are necessary to make sense of volume data. Furthermore, for how much counterintuitive can this be, they are very difficult to define effectively and require an expert user. Finally, they usually need to be adjusted for each new volume data. In practical terms this means that the transfer function to show the skeleton on one volume dataset may not be fit to serve the same purpose on a different dataset.

The key idea behind my activity was to leverage on the fact that on each practical use volume data must be coupled with one or more transfer functions, and hence that the latter could be harness to better compress the data.

We proposed a weighting scheme for taking into account a set of transfer functions in volume compression algorithms. With our scheme, the knowledge of those transfer functions is leveraged to obtain a better compression and peak signal to noise ratio. Our method relies on the definition of *neighborhood* of a transfer function, intended as the set of transfer functions that can be obtained by the input one by making small fine-tuning modifications. We process the input dataset by mapping its voxel values to the neighborhood of input transfer functions and analyze the result to classify the importance of each region of the dataset. The importance of a region is then used to weight it's contribution to the compression error function. We tested our algorithm with well known volume compression schemes such K-SVD and the early results shows a solid improvement both on compression time, ratio and peak signal to noise ratio (PSNR).

2.1 Compression Scheme

We used a block-based sparse coding compression scheme. In this kind of scheme, the data is subdivided in regular blocks of $m = n^3$ voxels. These blocks are then regarded to as m dimensional vectors and the compression error is defined as:

$$Err(D, X) = \sum_i \|y_i - Dx_i\|_0^2, \quad \|x_i\|_0 \leq s \tag{1}$$

where $Y = \{y_i\}$ is the $m \times p$ original data, D is a $m \times k$ matrix called *dictionary*, $X = \{x_i\}$ is the $k \times p$ matrix of coefficient, and s is the maximum number of non zero entries in each column

vector x_i. In short, each y_i is approximated by combining at most s vector of the dictionary: Dx_i (hence the name 'sparse coding').

The error is found as the sum of approximation errors in for each each block and the dictionary D and the coefficients X are learned through the minimization of $Err(D, X)$.

We focused on the idea of *weighting* the contribution of each block so to concentrate the accuracy in those parts that are involved in some of the transfer functions that will be used in the specific type of dataset. Referring the the first practical example: if we know that the transfer function showing bones will be used on the dataset, all the block of data that do not correspond to bones will have a smaller weight.

2.2 Transfer functions-based block weighting

Given a transfer function $\mathbf{t} : I\!R \rightarrow RGB\alpha$ we define a function $\omega_t : I\!R^m \rightarrow I\!R$ that determines the weight of the vector/block i in the error function Err. Defining ω_t was (and is) part of our research. One simple definition relies the average and variance of the opacity value:

$$\omega_t : \mathbf{E}(t(v)_\alpha)\ \mu(t(v)_\alpha) \tag{2}$$

In this way, blocks with low average opacity and small variance are considered less important. A slightly more involved formulation is:

$$\omega_t : \mathbf{E}(t(v)_\alpha)\ \mu(t(v)_\alpha)\ \mathbf{rd}(t(v)) \tag{3}$$

where

$$\mathbf{rd} = radius\ of\ the\ smallest\ enclosing\ sphere\ of \{\mathbf{t}(v)_{|RGB}\}$$

that penalizes also color-uniform blocks.

We have been experimenting with a few of these definitions and tested on a benchmark (see section 2.3.

However, the case of a single fixed transfer function is too limiting, for two reasons:

- in any practical use, several distinct transfer functions are used in the same type of volume data
- most of the times, every transfer function needs to be adjusted for every new dataset

Therefore we extended the idea of block weighting in two directions. First, we defined the *neighborhood* of a transfer function as:

$$\mathcal{N}(tf(v)) = [tf(v) - \delta(v), tf(v) + \delta(v)]$$

that is, each scalar value is mapped to an interval, a range of values, that accounts for adjustment of the transfer function tf. Then, the weigth ω is redefined accordingly. Second, given a set of transfer function $T = \{tf_i\}$, the weight is defined as the maximum weight over all the weights calculated using each transfer function separately:

$$\omega_i = max\{\omega(tf_0), \ldots, \omega(tf_h)\}$$

2.3 Benchmark and Results

Unfortunately there seemed not to be already organized benchmark data of such type. Given the effort that must be spent on building good sets of transfer functions, expertes and practitioners who design them are not willing to give them away lightly. Therefore part of the activity was on creating the benchmark to try our solution, consisting of several of volume data of various type and size with associated sets of transfer functions. We found a number of volume data and transfer presets from existing visualization software example datasets, such *Slicer* [3] and *imagevis3d* [4] and from the Visisble Human Dataset [1]. These data (both volume and transfer functions) come in a pletora of different format, so a conversion software had to be implemented in order to use them with our application.

As for the compression code, we involved the Visual Computing Group of Consorzio Ricerche Sardegna 4, who are relevant players in the field of volume visualization, who granted a version

of the source code of one of their previous works [5] that we could extend with our algorithms. Furthermore, following this first project, CRS4 is also willing to collaborate both with CNR and Trinity College in this topic.

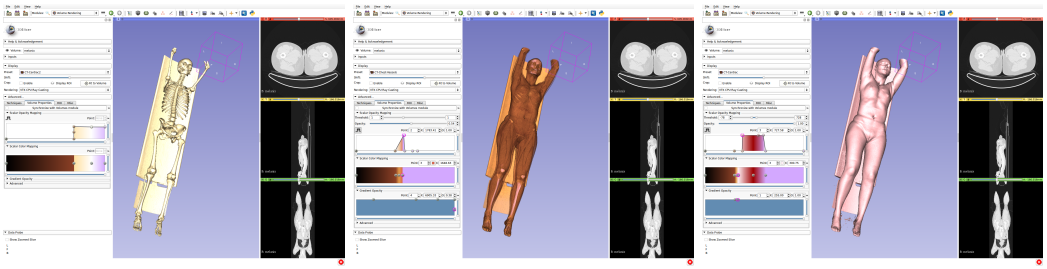

Figure 3: Example dataset *Melanix*, 512 × 512 × 1203 rendered with 3 different tranfer functions to evidentiate bones, muscles and skin.

dataset	rmse*	rmse	PSNR*	PSNR	time*	time
melanix 512 × 512 × 1203	0.00136625	0.00435422	57.2894	47.2218	21.1	170
tooth 256 × 256 × 161	0.00740672	0.00715468	42.6075	42.9082	0.4	2.5
head	0.00913644	0.0206525	40.7845	33.7005	15.5	6.5

3 Conclusions

Although we are still continuing the development phase, striving to bring the algorithm to live to its potential, the tests conducted so far have been successful and, as they are, already provide a solid result deserving publication. Our transfer function sensitive approach outperforms current state of the art approaches. It has to be said that the the previous work do not assume the knowledge of the transfer functions that will be applied at visualization time, therefore this kind of outcome was somehow grated beforehand. Nonetheless, our approach is general and can be applied to many existing volume compression algorithms, provided that they work with a subdivision of the volume in blocks, which essentially amount to most of the existing algorithms.

References

[1] The visible human project. https://lhncbc.nlm.nih.gov/project/visible-human-project.

[2] M. Balsa Rodríguez, E. Gobbetti, J.A. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S.K. Suter. State-of-the-art in compressed gpu-based direct volume rendering. *Computer Graphics Forum*, 33(6):77–100, 2014.

[3] Andriy Fedorov, Reinhard Beichel, Jayashree Kalpathy-Cramer, Julien Finet, Jean-Christophe Fillion-Robin, Sonia Pujol, Christian Bauer, Dominique Jennings, Fiona Fennessy, Milan Sonka, John Buatti, Stephen Aylward, James V. Miller, Steve Pieper, and Ron Kikinis. 3d slicer as an image computing platform for the quantitative imaging network. *Magnetic Resonance Imaging*, 30(9):1323 – 1341, 2012. Quantitative Imaging in Cancer.

[4] Thomas Fogal and Jens Kruger. Tuvok, an Architecture for Large Scale Volume Rendering. In *Proceedings of the 15th International Workshop on Vision, Modeling, and Visualization*, November 2010.

[5] Enrico Gobbetti, José Antonio Iglesias Guitián, and Fabio Marton. Covra: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. *Computer Graphics Forum*, 31(3pt4):1315–1324, 2012.