Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

A Compression Scheme for Volumetric Data Based on the Local Cosine Transform

Anselmo Cardoso de Paiva, Luiz Velho, Marcelo Gattass

Technical Report TR-02-07 Relatório Técnico

February - 2002 - Fevereiro

A Compression Scheme for Volumetric Data Based on the Local Cosine Transform

ANSELMO CARDOSO DE PAIVA[1 3], LUIS VELHO[2], MARCELO GATTASS[3]

[1]UFMA–Universidade Federal do Maranhão - CCET - Dep. de Informática
Campus do Bacanga, SN, São Luis, MA, Brasil
paiva@deinf.ufma.br
[2]IMPA–Instituto de Matemática Pura e Aplicada
Estrada Dona Castorina, 110, 22460 Rio de Janeiro, RJ, Brasil
lvelho@visgraf.impa.br
[3]TeCGraf - Grupo de Tecnologia em Computação Gráfica
Rua Marquês de S. Vicente, 225, 22453-980, Gavea, Rio de Janeiro, RJ, Brasil
{paiva,gattass}@tecgraf.puc-rio.br

Abstract. We propose a new lossy compression scheme for volumetric data based on the local cosine transform. This method is appropriated for further volume visualization because it provides good compression, minimizes reconstruction errors, and allows local decompression of the volume. We analyse the compression results and estimate scheme parameter variations, investigating the adaptivity properties and different space decomposition arrangements.

1 Introduction

Volumetric datasets are being used in increasing number, size, and complexity in many engineering and scientific applications. These applications are generally related to large and complex datasets that need to be submitted to volume visualization techniques. Volume visualization techniques have two classical computing problems: long execution time and large memory requirements. To handle the execution time requirement, several accelerated rendering techniques have been proposed, e.g. [1], [2], and [3]. The memory requirement has received less attention, even though for some applications, such as seismic analysis, it is very important. For example, for a resolution of 512^3 voxels, 134 Mbytes of memory are required if only one byte is allocated per voxel. Even with the memory cost decreasing, the size of the application's datasets never stops increasing.

Compression is very important to the visualization of large volume data on personal computers or low-end workstations with limited memory. What is needed is a method that allows the user to load a compressed version of the volume into a small amount of memory and enables the user to access and visualize it as if the whole uncompressed volume were present. Such a compression scheme should provide the local decompression of the data.

1.1 Previous Work

During the first years in the development of volume visualization techniques, the research efforts were concentrated on the search for real-time methods. Only with the emergence of WWW and distributed applications, the compression of volumetric data start to receive more research attention.

Ning and Hesselink [4] proposed a method that provided compression and visualization of volumetric data based on vector quantization. The volume is divided in blocks that are compressed using vector quantization, and can be directly visualized, without the need of decompression. The associated visualization process was too expensive for visualizing a compressed volume.

Muraki [5] introduced the use of wavelets to efficiently approximate volumetric data. Ihm and Park [6] proposed a scheme for the compression of large volumes, based on the wavelets transform. The proposed scheme yields high compression rates, and allows random access to the compressed voxels. This characteristic makes the scheme suited for the development of volume visualization techniques adapted to the compressed data structure. Following this work, [7] introduced the use of run-length encoding of the wavelet coefficients achieving better compression rates.

In [8] a method was proposed based on the wavelet transform and on the three-dimensional sub-band transform code algorithm developed to code video sequences, which provides fast random access to individual voxels within the volume and high compression rates. This algorithm has the problem of a high decoding cost.

The local cosine transform was discovered by several independent works ([9], [10], [11], [12] and [13]). They discovered the way to construct smooth orthogonal bases subordinate to arbitrary partitions of the real line. Malvar showed in [9] that discrete orthogonal bases with smooth windows modulated by a Cosine IV basis would be created, and Coifman and Meyer [10] rediscovered this result for continuous time functions. This construction is also known by different names: Lapped Cosine Transform (LCT), Lapped Orthogonal Transforms (LOT), Modulated Lapped Orthogonal Transforms, Time-Domain Aliasing Cancellation Filter Banks, Malvar Wavelets, and others.

In [9] the local cosine transforms were used to transform coding, given the reduction of the blocking effects and transform coding gain similar to DCT. In that work, local cosine transform was applied to the compression of unidimensional speech signals.

1.2 Contribution

As we can see in the literature, several techniques handling the compression of volumetric data have been recently emerging. In this paper we propose a new method based on the local cosine transform of volumetric data, generating good compression rates, minimizing the reconstruction errors and being suitable for local volume decompression. We also investigate various space (time) subdivision schemes and their reflections on compression results. This method extend some of the existing methods applied to images, adding the advantage to incorporate the correlation in the slices direction.

In the next section, we discuss local cosine bases and their properties. In section 3 we present our volumetric compression scheme and discuss some results of applying such scheme to real datasets. Finally, we present the obtained results in section 4 and provide concluding remarks in section 5.

2 The Local Cosine Transform

In the study of signal representation with the least possible correlation, we see that the correlation present in the signal is mostly local. Thus, the Fourier bases are not well suitable because their basis functions are non local. More localization can be obtained by dividing the time axis and separately computing the Fourier basis for each interval. This approach has the disadvantage of introducing the so called border effects. The Fourier basis border effects generate large coefficients, which are introduced because the signals analyzed are not, in general, periodic. This fact is interpreted by the Fourier analysis as a discontinuity or an abrupt variation in the function. In order to overcome this problem, we can use trigonometric bases (e.g. cosine) multiplied by a smooth window function. This way we gain more local correlation in the signal representation, minimizing the insertion of border effects and, additionally, obtaining the freedom to choose the time axis partition that best represents the signal. In the following sections we will discuss how to generate the local cosine basis and the method to search the bases that better represent a signal.

2.1 Local Cosine Basis

Let $\{I_p\}$ be a partition of the time axis in overlapping intervals, such that:

$$I_p = [a_p - \eta_p, a_{p+1} + \eta_{p+1}] \tag{1}$$

with $\lim_{p\to-\infty} a_p = -\infty$ and $\lim_{p\to+\infty} a_p = +\infty$, such that I_{p-1} and I_{p+1} do not intersect each other.

We can define a window function g_p whose support is I_p, and which has a raising profile on the left sobreposition subinterval(O_p) and a decaying profile on the right sobreposition subinterval(O_{p+1}), given by

$$g_p(t) = \begin{cases} 0 & \text{if } t \notin I_p, \\ \beta\left(\frac{t-a_p}{\eta_p}\right) & \text{if } t \in O_p, \\ 1 & \text{if } t \in C_p, \\ \beta\left(\frac{a_{p+1}-t}{\eta_{p+1}}\right) & \text{if } t \in O_{p+1}. \end{cases} \tag{2}$$

where $\beta(t)$ is a monotone profile function such that

$$\beta(t) = \begin{cases} 0, & \text{if } t < -1 \\ 1, & \text{if } t > -1 \end{cases} \tag{3}$$

$$\forall t \in [-1,1] \quad \beta^2(t) + \beta^2(-t) = 1 \tag{4}$$

Now we can define a local basis of $L^2(\mathbb{R})$ as a basis derived from a Cosine-IV basis of $L^2[0,1]$ by multiplying a translation and dilation of each cosine basis function by the smooth window g_p. The family of local cosine functions

$$\left\{ g_{p,k}(t) = g_p(t)\sqrt{\frac{2}{l_p}} \cos\left(\pi\left(k+\frac{1}{2}\right)\frac{t-a_p}{l_p}\right)\right\}_{\substack{k\in\mathbb{N} \\ p\in\mathbb{Z}}} \tag{5}$$

is an orthonormal basis of $L^2(\mathbb{R})$. The local cosine bases can be characterized, like wavelet packets, by position p, scale s, and wavenumber k, supposing $lp = f(s)$.

These bases are composed of functions $g_{p,k}(t)$ with compact support $[a_p - \eta_p, a_{p+1} + \eta_{p+1}]$, as shown in Figure 1.

Local cosine bases are discretized by replacing the orthogonal basis of $L^2[0,1]$ with a discrete Cosine-IV basis, and uniformly sampling window g_p. So, if we have a sampled window $g_p[n]$, then the family

$$\left\{ g_{p,k}[n] = g_p[n]\sqrt{\frac{2}{l_p}} \cos\left[\pi\left(k+\frac{1}{2}\right)\frac{n-a_p}{l_p}\right]\right\}_{\substack{0\le k<l_p \\ p\in\mathbb{Z}}} \tag{6}$$

is an orthonormal basis of $l^2(\mathbb{Z})$.

To decompose a function f in the basis, one must compute the inner products of the function with each discrete

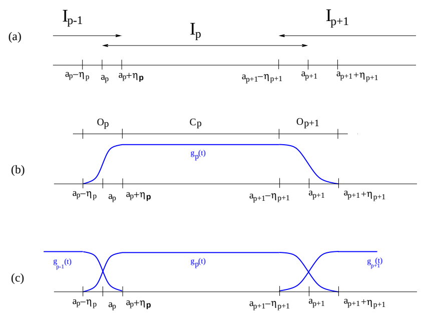

Figure 1: (a) The overlapping intervals spanning the time axis. (b) Window $g(t)$ and subintervals O_p, C_p and O_{p+1}. (c) The lapped windows $g_{p-1}(t)$, $g_p(t)$ and $g_{p+1}(t)$.

basis vector. A fast algorithm introduced in [14] replaces the calculations of $\langle f, g_{p,k}\rangle$ by a computation of inner products in the original bases, which can be computed with the fast DCT-IV algorithm, with a folding procedure.

Suppose we wish to fold a function f across a_p, onto the intervals $[a_p - \eta_p, a_p)$ and $(a_p, a_p + \eta_p]$, using the window function g_p, the folded function $f_p^f(x)$is given by:

$$f_p^f(x) = \begin{cases} g_p(x)f(x) + g_p(2a_p - x)f(2a_p - x) & \text{if } x \in O_p \\ f(x) & \text{if } n \in C_p \\ g_p(x)f(x) - g_p(2a_{p+1} - x)f(2a_{p+1} - x) & \text{if } x \in O_p \end{cases} \tag{7}$$

The folding operation can be graphically represented by flipping the modulate function (Figure 2(b)) portion within the interval $[a_p - \eta_p, a_p]$ ($[a_{p+1}, a_{p+1} + \eta_{p+1}]$) around the axis $t = a_p$ ($t = a_{p+1}$)(Figure 2c) and summing the flipped portion to the function portion supported in the subinterval $[a_p, a_p + \eta_p]$ ($[a_{p+1}, a_{p+1} + \eta_{p+1}]$) (Figure 2d), generating the folded function as shown in Figure 2e.

We can extend the local cosine bases to represent multidimensional signals. The easiest way is through the use of tensor products of unidimensional basis elements.

A local cosine basis in the 3D space can be constructed by partitioning the space $\mathbb{R}^3$ into volume intervals $\{[a_p, b_p] \times [c_q, d_q] \times [f_r, h_r]\}$ with arbitrary length $l_p = b_p - a_p$, width $w_q = d_q - c_q$, and depth $s_r = h_r - f_r$.

2.2 Local Cosine Tree

In the local cosine basis construction we can segment the time axis into intervals $[a_p, a_{p+1}]$ of arbitrary length. However, Coifman and Meyer [10] showed that restricting this partition to dyadic size we create a tree structure representing all possible time axis partitions. The generated structure, called Local Cosine Tree, is very similar to the

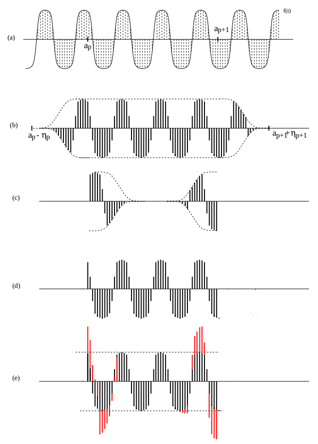

Figure 2: Graphical representation of the fold operation.(a) Function $f(t)$ to be folded. (b) The modulated version of $f(t)$. (c) The flipped portion of the modulated $f(t)$ that was outside the interval. (d) The portion of $f(t)$ inside the interval. (e) The flipped portion added to the inside portion of the $f(t)$.

wavelet packet tree, introduced in [15], and may be used to search the local cosine basis that is more adapted to the signal characteristics.

If we consider a time interval $[0, T]$ as the signal extent, we can divide into in $p = 2^j$ intervals, $I_p = [a_{p,j}, a_{p+1,j}]$ where $a_{p,j} = p2^{-j}T$ for $0 \le p \le 2^j$, which has length $l_p = 2^{-j}T$. The intervals in which $1 \le p \le 2^j - 2$ have support $[a_{p,j} - \eta, a_{p+1,j} + \eta]$, defined by window $g_{p,j}$ given by:

$$g_p(t) = \begin{cases} \beta\left(\frac{t - a_{p,j}}{\eta}\right) & \text{if } t \in [a_{p,j} - \eta, a_{p,j} + \eta], \\ 1 & \text{if } t \in [a_{p,j} + \eta, a_{p+1,j} - \eta], \\ \beta\left(\frac{a_{p+1,j} - t}{\eta}\right) & \text{if } t \in [a_{p+1,j} - \eta, a_{p+1,j} + \eta], \\ 0 & otherwise. \end{cases} \tag{8}$$

If we compute all possible dyadic intervals, we generate a tree , in which each level j represents a partition of the signal extent $[0, T]$ into 2^j subintervals. Then a local cosine tree node at level j and position p in this level is generated by the local cosine family

$$B_j^p = \left\{ g_{p,j}(t)\sqrt{\frac{2}{2^{-j}T}} \cos\left[\pi\left(k + \frac{1}{2}\right)\frac{t - a_{p,j}}{2^{-j}T}\right]\right\}_{k\in\mathbb{Z}} \tag{9}$$

In order to guarantee that the window overlaps only with its two adjacent intervals, length $l_p = a_{p+1,j} - a_{p,j}$ must be greater than $2^{-j}T$. As in the dyadic partition the length of an interval in the tree level j is given by $l_p = 2^{-j}T$, we have that the value of η is limited by $\eta \leq 2^{-j-1}T$ or that the maximum tree level is limited by $\log_2\left(\frac{T}{2\eta}\right)$.

To represent a signal we simply need to choose from the nodes of the local cosine tree a set of intervals that covers the complete signal extent. This is due to the fact that any interval $I_{j,k}$ in the tree can be obtained by the sum of its two child intervals $I_{j+1,2k}$ and $I_{j+1,2k+1}$. Therefore, the representation of a signal can be obtained by cutting the local cosine tree and selecting only intervals associated to the tree leaves, as illustrated in the Figure 3. The number of different dyadic local cosine bases is equal to the number of different subtrees of depth J at most.

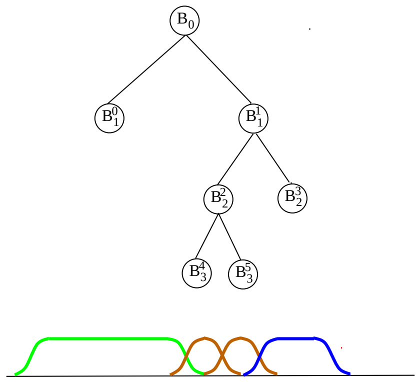

Figure 3: An admissible binary tree of local cosine bases.

The local cosine binary tree is extended in two dimensions into a quad-tree which divides the rectangular window into four smaller windows. In three dimensions, the generated structure is an octree which divides the volume window into eight sub-volume windows.

2.3 Best Basis Selection

A local cosine basis divides the time axis into intervals of varying sizes. To adapt the time segmentation to the variations of the signal time-frequency structures we can compare different arrangements from the local cosine tree to choose the one that provides the best signal representation. By minimizing a concave cost function we can select from the collection of local cosine bases in the tree, the subset that is the best representation for a signal. The cost of approximating f in a basis B^λ is defined by the Schür concave sum

$$C(f, B^\lambda) = \sum_{m=1}^{N} \Phi\left(\frac{|\langle f, g_m^\lambda\rangle|^2}{\|f\|^2}\right) \tag{10}$$

where $\langle f, g_m^d\rangle$ defines the representation of f in the basis B^λ, and Φ is the single concave function.

As examples of cost functions we can mention entropy and concentration in l^p. Entropy, given by $\Phi(x) = -x\log_e x$, is concave for $x \geq 0$. The corresponding cost is called entropy of the energy distribution, and is given by:

$$C(f, B) = -\sum_{m=1}^{N} \frac{|\langle f, g_m\rangle|^2}{\|f\|^2} \log_e\left(\frac{|\langle f, g_m\rangle|^2}{\|f\|^2}\right) \tag{11}$$

The l^p cost, $\Phi(x) = x^{\frac{p}{2}}$ for $p < 2$, is concave for $x \geq 0$. The resulting cost is:

$$C(f, B) = \sum_{m=1}^{N} \left(\frac{|\langle f, g_m\rangle|^2}{\|f\|^2}\right)^{\frac{p}{2}} \tag{12}$$

If the collection of bases is a tree with finite depth L, then we can find the best basis by computing the information cost of each node of the tree (transform block) and comparing children to parents starting from the bottom.

The best basis algorithm in 3D computes a cost function for each volume block. In this work we used as cost functions threshold, concentration in l^p and entropy. The threshold cost function simply counts how many coefficients have absolute value above a threshold value ϵ. The concentration in l^p is obtained by summing the p^th powers of block coefficients, for $0 < p < 2$. Finally, the entropy function used is the norm $-l^2 \log l^2$. The main idea of the fast algorithm is that the full local cosine octree is pruned recursively at each node by comparing its entropy to the summation of its corresponding children nodes,

```
if $Cost(ParentNode) \leq [Cost(child1) + Cost(child2)$
  $+Cost(child3) + Cost(child4) + Cost(child5)$
  $+Cost(child6) + Cost(child7) + Cost(child8)]$
  then
      cut off the child branches.
fi
```

In the beginning, a full binary-based decomposition tree with a preset maximum decomposition level is produced. Then, the pruning procedure starts from the leaf nodes towards the root. At the end of this procedure, an optimal pruned tree is obtained for the given signal, the so called adaptive binary local cosine basis. The search low complexity is obtained by the fact that each node is visited only twice. This occurs due to the cost functions' additive property. For a J-levels decomposition of a signal with $N = s^J$ samples we need just $O(N)$ comparisons.

3 Volumetric Data Compression

The transform based compression methods are commonly used for volumetric data with good results. In these methods, we have three main stages. The first and most important one, is the transform of the data representation to a domain, which is more appropriated to the processing of the following stages. The second and third stages are, respectively, quantization and codification. Essentially the first stage defines what will be coded in the subsequent stages.

The basic idea is to transform the data such that the information is concentrated in just a few coefficients , then we can discard the coefficients near zero, use a few bits to code the less important coefficients, and concentrate the representation in most significant coefficients. The transform's desired properties are: existence of fixed function basis, existence of efficient computation algorithms, separability, reduction of correlation between coefficients, and good energy compaction properties.

DCT (Discrete Cosine Transform)[16] is one of the most used transforms for compression. The reason is that this transform approximates the Karhunem-Loéve transform which de-correlates all the data values, but is computationally expensive because it is data dependent. To add more desired properties to the discrete cosine transform and avoid the introduction of discontinuity artifacts, we will use the Local Cosine Transform (LCT) described in section 2, which is a transform with a C^∞ orthonormal base of compact support and that generates a representation in scale for the data. Additionally, this transform has the advantage of adapting volume partition to data characteristics in the space$\times$frequency domain.

Figure 4 describes the general flow of the proposed method, which follows the main structure of a transform based compression method. The initial and principal stage is the application of LCT, followed by typical lossy coder steps: thresholding, removing insignificant coefficients, quantization of the remaining coefficients to restrict them to a small number of possibilities, and finally encoding the transformed data.

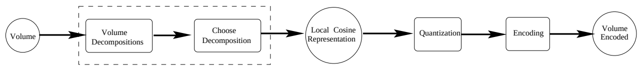

Figure 4: Overview of the Local Cosine Based Compression Method.

3.1 Volume Transformation

The initial stage of the compression method is the definition of a volume partition to apply the LCT. This partition can be achieved using the Fixed Level approach, which subdivides the volume into sub-volumes of equal size ($n_x \times n_y \times n_z$), or using the Best Basis approach, that find an adaptive representation for the volume. The description of the volume decomposition must be encoded in such a way that the decoder will be able to preserve the block arrangement. This is essential to retrieve blocks following any axis traversal order, an important characteristic to allow the development of visualization algorithms based on the compressed data structure.

Using the Fixed Level approach, we specify the dyadic decomposition level and the sobreposition extension (lap) that will be used. Based on this choice, the algorithm computes the associated volume partition. We have the volume decompositions restricted to the dyadic partitions of the three axis. Therefore a volume of size $N_x \times N_y \times N_z$ has 2^{3j} blocks in the decomposition level j, each block with size $\frac{N_x}{2^j} \times \frac{N_y}{2^j} \times \frac{N_z}{2^j}$. For each decomposition block we apply three one-dimensional local cosine transforms, folding the contribution of the six neighbors blocks. We can see this process as the application of a LCT in $N_x \times N_y$ one-dimensional arrays formed by the (i, j) voxel of each slice in the volume, using 2^j intervals. This is followed by the application of the same transform to the columns and lines of each resulting slice. The maximum dyadic decomposition level j_{max} is determined by the size (lap) of the overlapped portion of local cosine windows, and is given by $j_{max} = \log_2(2 * lap)$. In the implemented transform, we used the same value of lap for all blocks.

Transform efficiency is a measure of the energy retained in a particular number of coefficients. We measure the energy retained in the first m coefficients, and divide by the energy retained by all coefficients, given an indication of the energy retained in the lower frequency coefficients.

The second approach is based on the search for a volume partition that is adapted to the frequency characteristics of the volume. In this method we apply all the dyadic partition levels to the volume, generating an octree, and applying the LCT for each level. Then, the octree is traversed, using the best basis algorithm searching for the best volume partition.

The cost of adding adaptivity to the transform coding scheme is a header describing the decomposition of the volume into blocks. For example, for a volume of size 256^3 and lap size greater than 1, we need at most 21 bits per block to code the level and position of each block, so the overhead of the adaptive scheme, considering at most 262144 blocks, is of 5376 bytes, for a volume with 1 byte per voxel will have an overhead of 0.00032 bits per voxel.

3.2 Quantization and Encoding

We use the same quantization step for a sub-volume as in [17]. For all elements in a sub-volume we compute the dynamic range of coefficients, given by:

$$R_i = \|maxValue - minValue\| \tag{13}$$

where $maxVal$ and $minVal$ are the maximum and minimum transform coefficients values for the sub-volume. The quantization step for this sub-volume is set to $Q = \frac{R_i}{2^c}$, with c being a fixed constant, that is correlated to the number of bits necessary to represent the quantized coefficients. In this work we used $c = 16$ and $c = 8$.

To select the most significant coefficients, we follow the approach of Chen and Pratt ([18]), who suggested block scanning in a zig-zag order. The 3D zig-zag order through a block is the traversal order in which (u_1, v_1, w_1) precedes (u_2, v_2, w_2) if $u_1+v_1+w_1 < u_2+v_2+w_2$, and the (u, v, w) tuples on the plane $u + v + w = K$ follow a 2D zig-zag order. This approach is based on the assumption that a large section of the tail end of the scan will consist of zeros because, in general, higher order coefficients have smaller amplitude, which can be verified analyzing the transform efficiency computed for the test datasets using a normal block traversing (traversing each slice in a row order) and using the 3D zig-zag traversal. The results presented in section 5 show that this assumption is correct.

The linear sequence of transform coefficients resulting from the 3D zig-zag traversal of the blocks is fed into an entropy encoder. To encode the coefficients we retain a percentage of them and set the others to zero. Then, the retained coefficients are submited arithmetic coding to generate small entropy codes for them.

The paper [19] is one of the well-known papers that provided practical arithmetic coding algorithms. This coding method is based on the assumption that, it is more efficient to generate codewords for groups of symbols sequences rather than generating a separate codeword for each symbol in a sequence. The main advantages of arithmetic coding are its theorically optimality, great flexibility, and adequation to the where the probabilities are highly unbalanced.

On other hand, arithmetic coding tends to be slow, generally does not produce a prefix code, needs to indicate the end of the sequence and has poor error resistance.

The arithmetic coding implementation used in this work was made available by Alistair Moffat, and was based on [20].

Decompression is achieved by applying the inverse stages of the compression method, in the reverse order. The inverse LCT transform of the blocks must be performed following an fixed order, because the complete decompression of one block depends on the decompression of its six neighbors and on the addition of their contributions in the overlapped portion.

4 Results

4.1 Test Datasets

To evaluate the results obtained with the proposed compression scheme and the influence of parameter variations in the compression, we used the datasets described in the Table 4.1.

Dataset	Size	Bits/voxel	Modality
3DHead	256×256×64	8	MRI
Colt	256×256×64	8	SEISMIC
Broncho	512×512×128	8	CT
VH	512×512×128	16	CT

Table 1: Volume Datasets Description

The VH dataset is the same used in [6], [7]and [8], that was made available by Professor Insung Ihm. The dataset used is a clipped part of the original dataset with 512×512×128 voxels, rebuilt from the fresh CT slices of Visible Human, with 12 bits per voxel, stored in 16 bits.

4.2 Transform Results

First we analysed the influence of the zigzag traversal against the normal block traversal. To do this we compute the transform efficiency (TEF) for various datasets using the two traversal options. Transform efficiency is a measure of the energy retained in a particular number of coefficients. We measure the energy retained in the first m coefficients, and divide by the energy retained by all coefficients, given an indication of the energy retained in the lower frequency coefficients.

The results obtained show that the zigzag traversal is a best choice, giving up to 35% increase in the TEF. It was also observed that the increase in the TEF due to the zigzag traversal is greater when we use less coeficients to represent the volume blocks. These assumptions are shown in the Figure 5, where we present the computed TEF for the BRONCHO dataset.

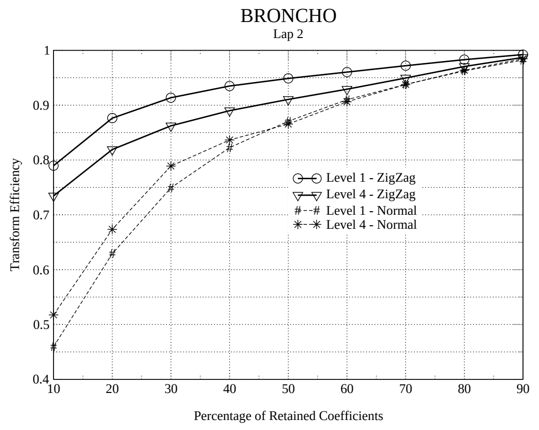

Figure 5: Comparison of the Transform Efficiency obtained by traversing the transformed blocks in zig-zag order and slice by slice.

From all datasets tested we conclude that the TEF is inversely proportional to the decomposition level, what represents that the TEF increases as we use less blocks to represent the volume. The lap size also affects the TEF, doubling the lap size decreases the TEF of approximately 2%.

Evaluating how the transform parameters affect the compression, we verified that the use of small transform blocks reduce the reconstruction error. This implies that we get better compression results using greater decomposition levels. Other parameter that affects significantly the compression is the number of quantization cells used to quantize the transform coefficients. In the Figure 6 we verify that the use of 2 aditionals bits to represent the quantized coefficients gives great reduction in the MSE. The lap size did not affect significantly the reconstruction error and compression rate, for a fixed level, the rate distortion curves generated for various lap size were almost coincident.

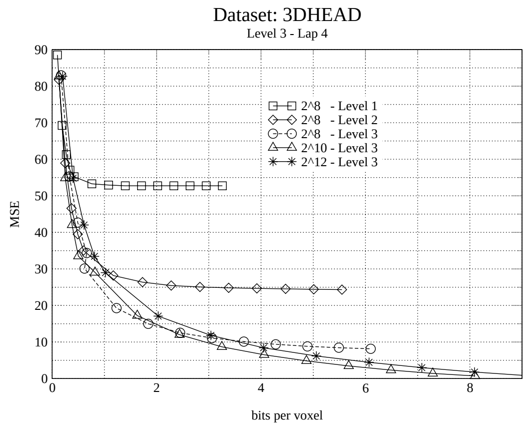

Figure 6: MSE comparison for different levels and number of quantizations cells for the 3DHEAD dataset.

The best basis results were more highlighted in the lower compression rate zone. The Figure 7 present the results obtained for the COLT dataset, where we can verify that for $bpv > 2.5$, the best basis approach gets an increase of 2 dB in the PSNR from the best fixed level result (level 3 - lap 4). This indicates that even with the overhead of blocks description coding, the best basis approach increases the compression capabilities. Aditionally, with this approach we did not need to find the level/lap combination more suitable to the dataset. We verifed too, that the cost functional used can modify the best basis result. In general, the L^{p}Norm was the most adequate cost functional.

Comparing the results obtained using the VH dataset we get result similar to that obtained in [6], using decomposition level 5 and lap equal to 2, being better than the result presented in [7]. As we compare the results with the work [8] we verified a diference from 0.25 to 0.6 bits per voxel,

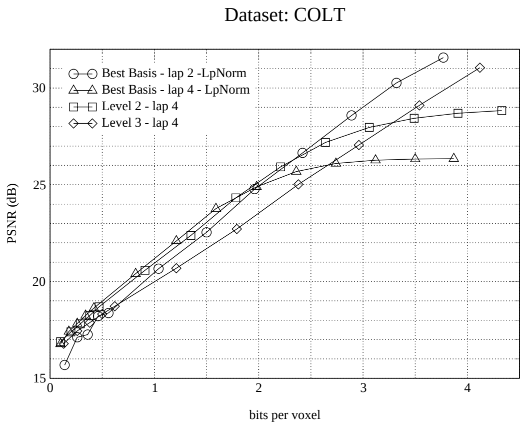

Figure 7: PSNR comparison of Best Basis and the Fixed Level approaches.

for the same PSNR level. But, this compression gain obtained in the Rodler work was due to the use of predictive code between volume slices, a characteristics, as mentioned in his work, that make his method not useful for an interactive visualization environment, because of its low decoding speed. These results are shown in the Figure 8.

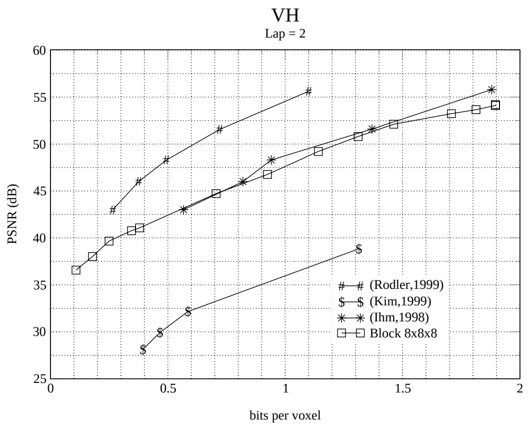

Figure 8: PSNR comparison of the datasets.

5 Conclusion

We have applied the local cosine transform to the compression of volumetric data, developing a promising algorithm for the compression of these data with the possibility of local volume decompression, and being adequate for further development of a visualization algorithm that allow a wider range of users to work with very large volume data. From the obtained results we verified that the best compression are related to the use of small volume blocks. In this way we explore the existence of volume homogeneous regions, what can be used to improve the compression. Also it was observed that the best basis algorithm present better results than the fixed level approach, even though its use for a visualization systems is more complex.

The quantization stage is a central problem in the compression scheme. For blocks with size greater than $32 \times 32 \times 32$ is necessary the use 2^{10} or more quantizations cells, to avoid the introdution of artifacts. To achieve high compression rates with small reconstruction errors we must use small blocks (e.g. 4^3 or 8^3 voxels) wich make possibe the use of less quantizations cells.

We are currently investigating quantization strategies that produce best compression ratios while maintaining the same reconstructed data quality, and the effects of the reconstructions errors in the visualization of the volume data. More investigation is needed also to evaluate the influence of the cost functionals in the best basis computation.

References

[1] M. Levoy. Efficient ray-tracing of volume visualization algorithms. *ACM Transactions on Graphics*, 9(3):245–261, 1990.

[2] J. Danskin and P. Hanrahan. Fast algorithms for volume rendering. In *ACM Workshop on Volume Visualization 1992*, pages 91–98, 1992.

[3] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics*, 28(3):451–458, 1995.

[4] P. Ning and L. Hesselink. Fast volume rendering of compressed data. In *Visualization 1993*, pages 11–18, 1993.

[5] S. Muraki. Aproximation and rendering of volume data using wavelet data fields. In *IEEE Visualization'92 Conference Proceedings*, pages 21–28, October 1992.

[6] I. Ihm and S. Park. Wavelet-based 3D compression scheme for very large volume data. In *Graphics Interface'98*, pages 107–116, 1998.

[7] T.-Y. Kim and Y. G. Shin. An efficient wavelet-based compression method for volume rendering. In *Pacific Graphics'99*, Seoul, Corea, October 1999.

[8] F. R. Rodler. Wavelet based 3d compression for very large volume data supporting fast random acces. Technical Report BRICS RS-99-34, Department of Computer Science, University of Aarhus, Aarhus C, Denmark, 1999. available at http://www.brics.dk.

[9] H. S. Malvar. The LOT: A link between block transforms coding and multirate filter banks. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 835–838, June 1988.

[10] R. R. Coifman, Y. Meyer, and M. V. Wickerhauser. *Wavelet Analysis and Signal Processing*, pages 153–178. Jones and Bartlett, Boston, 1991. In: Wavelet and Their Applications, edited by B. Ruskai et al.

[11] J.P. Princen and A. B. Bradley. Analysis/synthesis filter bank design based on time domain aliasing cancelation. *IEEE Transactions on Acoustics, Speech and Signal processing*, 34(5):1153–1161, 1986.

[12] I. Daubechies and J. C. Lagarias. Two scale difference equation: Existence and global regularity of solutions. *SIAM Journal on Mathematics Analysis*, 22(5):1388–1410, 1991.

[13] E. Laeng. Une base orthonormale de $l^2(\setminus)$, dont les Éléments sont bien localisés dans l'espace de phase et leurs supports adapté à toute partition symétrique de l'espace des fréquences. *Comptes Rendus de l'Académie des Science de Paris*, (311):677–680, 1990.

[14] H. S. Malvar. *Signal Processing with Lapped Transform*. Artech House, Norwood, MA., 1992.

[15] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2):713–718, March 1992.

[16] N. Ahmed and K. R. Rao. Discrete cosine transform. *IEEE Transaction on Computers*, C-23:90–93, January 1974.

[17] T.-C. Chiueh, C.-K. Yang, T. He, H. Pfister, and A. Kaufman. Integrated volume compression and visualization. Technical Report TR.94.01.04, State University of New York at Stony Brook, 1994.

[18] W.-H. Chen and W. K. Pratt. Scene adaptive coder. *IEEE Transactions on Communications*, (32):225–232, March 1984.

[19] J. J. Rissanem and G. G. Langdon. Arithmetic coding. *"IBM" Journal of Research and Development*, (2):149–162, March 1979.

[20] A. Moffat, R. Neal, and I.H. Witten. Arithmetic coding revisted. *ACM Transactions on Information Systems*, 16(3):256–294, 1998.