COMPRESSION OF FUNCTIONS DEFINED ON SURFACES OF 3D OBJECTS

Krasimir Kolarov, William Lynch
Interval Research Corporation, 1801 Page Mill Road, Bldg. C, Palo Alto, CA 94304

ABSTRACT

We present a technique to compress scalar functions defined on 2-manifolds. Our approach combines discrete wavelet transforms with zerotree compression, building on ideas from three previous developments: the lifting scheme, spherical wavelets, and embedded zerotree coding methods. Applications lie in the efficient storage and rapid transmission of complex data sets. Typical data sets are earth topography, satellite images, and surface parametrizations. Our contribution in this paper is the novel combination and application of these techniques to general 2-manifolds.

SECTION I INTRODUCTION

1.1 Modeling data as a function

We model the data to be compressed as a function. In this paper we will restrict our attention to the compression of (scalar) functions that map a 2-manifold into the real line. A 2-manifold is a space with the property that the neighborhood of each point is equivalent to a neighborhood of the origin of the plane, the half plane, or the quarter plane. Popular examples of 2-manifolds include the surfaces of 3D objects (e.g. a sphere).

Traditionally, data compression methods have been applied to functions defined on simple manifolds. Typical manifolds include: the real line (e.g., audio), a rectangle (e.g., images), or a three-dimensional open-ended box (e.g., video). However, many conventional data compression technologies, unmodified, are not suitable for compression of data defined on higher-dimensional or complex geometries such as spheres, general polytopes, etc. Accordingly, this paper seeks to provide a transform compression technique for addressing 2-manifold domains using second generation wavelet transforms and zerotree coding.

1.2 Structure of the paper

In what follows in Section II we will reference the wavelet transforms used in this paper (the second generation wavelets) and an efficient method for calculating such wavelets - the lifting scheme [9]. This methodology is used to build wavelet transforms of a function defined on a 2-sphere [7].

In Section III we will briefly discuss the scaling and quantization of the wavelet coefficients. The zerotree method provides for an embedded encoding with a set of thresholds differing by powers of one half. Appropriate scaling minimizes for the L_2 , L_1 or L_∞ norm.

Section IV will describe an adaptation of the zerotree coding method to coding the wavelet transform of a function on a general 2-manifold. We will describe in detail a novel organization for the zerotrees, called G-trees.

In Section V we report some compression results for applying the method to topographic data (elevation) defined on the surface of the Earth (a 2-sphere), as well as to a standard image (Lena) defined on the square (a 2-manifold with boundary). The latter example was chosen to enable comparison with traditional image compression.

SECTION II WAVELET TRANSFORMS ON 2-MANIFOLDS

2.1 Second generation wavelets

The basic ideas of wavelets and wavelet transforms were developed in the eighties by Yves Meyer, Ingrid Daubechies, and Stephane Mallat [2, 5]. A rather vague definition of wavelets is: "*Wavelet are* ***building blocks*** *that can* ***quickly decorrelate*** *data.*" This sentence shows three main features of wavelets that are the reason for their current popularity.

Classic wavelet constructions such as [1] always define the wavelet functions $\psi_i(x)$ as dyadic translates and dilates of one fixed function: "the mother wavelet $\psi_i(x)$." These wavelets are referred to as ***first generation wavelets***. They can be used in the compression of data defined on the real line (e.g. audio) or plane (e.g. images).

However, for the transformation of data defined on complex manifolds, first generation wavelets can no longer be used. Indeed translation and dilation only make sense in a Euclidean space. Therefore one needs to consider the generalization to *second generation wavelets*. These are wavelets which are not necessarily the translates and dilates of one function, but which still enjoy all the powerful properties of first generation wavelets, namely basis properties, decorellation power, and fast algorithms.

To construct the second generation wavelets we are using **the lifting scheme** [9], a new concept in the construction of wavelets that does not need the Fourier transform. It basically amounts to *splitting* the original data set into two subsets - data and detail (wavelet coefficients). At each level of resolution the data is compared to the predicted value for that data point by the transform and the difference between the two is stored in the wavelet coefficient - the stage of *prediction*. Afterwards the newly generated wavelet coefficient is used to update the predicted data value - the state of *update*. It has been shown that any finite classical wavelet can be generated as a result of splitting and a finite number of prediction and update steps. Thus the second generation wavelets are a superset of the finite classical wavelets.

A typical example of the use of lifting to construct second generation wavelets is on a sphere [7]. The sphere is a 2-manifold (i.e., a surface). The construction starts with the tessellation of the sphere into cells. This is done starting from a *base complex* (an icosahedron for this example) which is recursively subdivided. Each triangle is subdivided into four triangles and three edges, each bounded by a pair of midpoint vertices. The subdivision is iterated as many times as is required to support the desired wavelet approximation. Other subdivisions are possible but are not discussed here.

With two instances of lifting (prediction and update), one can obtain wavelets with the desired properties. These spherical wavelets can now be used to process scalar

functions defined on a sphere. Typical examples are spherical textures, environment maps and earth elevation data (topography).

2.2 Wavelets on general 2-manifolds

In fact there is very little of the sphere in the discussion of spherical wavelets [7] - only that the base complex is *homeomorphic* (i.e., topologically equivalent) to a sphere. To address a general 2-manifold we need to modify the construction by starting with a triangulated base complex that is homeomorphic to the 2-manifold of interest.

Examples are a photographic image (a function on the rectangle with x, y coordinates) and the ETOPO5 data set, which samples the Earth elevation with respect to sea level approximately every 5 arc minutes (a sampled function on the sphere with latitude-longitude coordinates).

In summary we have modeled a 2-manifold by a triangulation with subdivision connectivity, determined function values to attach to each vertex, and determined a method to calculate wavelet transforms on the structure. If the function is suitably smooth the norm of the wavelet transform will be mostly at a small number of vertices.

SECTION III SCALING, QUANTIZATION, AND APPROXIMATION RATE

3.1 Scaling

It is shown in [3, 4] that an optimal approximation for differing L_p norms requires differing scaling of the wavelet coefficients prior to any thresholding operation. (Equivalently, the threshold T could be made a function of the subdivision level of the coefficient.) Multiplication by unity optimizes for the L_∞ norm. To optimize for the L_2 norm the wavelet coefficients should be scaled by 2^{-j} where j is the subdivision level at which the coefficient is attached. To optimize for the L_1 norm the wavelet coefficients should be scaled by 4^{-j}. For the general L_p norm the scaling factor is $4^{\frac{-j}{p}}$. The contrast sensitivity curve for the human visual system is best approximated by using the L_1 norm and its associated scaling [3]. During decompression the wavelets are descaled by the inverse of the scaling factor used during compression.

3.2 Quantization

The function f that we want to compress is approximated by a function

(1) $$\hat{f}_T = \sum_{i \in \hat{I}_T \subseteq I} \hat{\gamma}_{Ti} \psi_i .$$

This function is a linear combination of the wavelet functions with coefficients:

(2) $$\hat{\gamma}_{Ti} = \begin{cases} 0 & \text{if } |\gamma_i| \le T \\ sgn(\gamma_i)\left(|\gamma_i| - (|\gamma_i| \bmod T)\right) & \text{if } |\gamma_i| > T \end{cases}$$ for a given threshold T.

We can define

(3) $$\hat{I}_T = \left\{ \hat{\gamma}_i \,\middle|\, |\gamma_i| > T \right\}$$ and

(4) $N = \left|\hat{I}_T\right|$, the cardinality of $\hat{I}_T$

It is clear that, for a given f, N can be considered a function of T and vice versa.

3.3 Approximation Rate

In [3] it is shown that there is an inverse relationship between N and the smoothness of f (the smoother f is the smaller N is). The precise relationship is

(5) $$\left\| f - \hat{f}_{T(N)} \right\|_{L_p(\Re^2)} \le CN^{-\frac{\alpha}{2}}$$

if and only if f is appropriately smooth. Appropriately smooth means that f is in the Besov space $B_q^\alpha\left(L_q\left(\Re^2\right)\right)$, which, in turn, means roughly that f has α derivatives in $L_q\left(\Re^2\right)$ where $\frac{1}{q} = \frac{\alpha}{2} + \frac{1}{p}$.

3.4 Embedded Encoding

This suggested to the authors of [3] that an embedded code could be achieved by using a sequence of T's which are powers of two. An *embedded* code is a code where the code for a coarser (larger T) approximation is a head of the code for a finer (smaller T) approximation. If the T's are powers of two then the code for the threshold T consists of the code for the threshold $2T$ followed by a code for one *bit plane* (i.e., one additional bit per coefficient, i.e. $\left(s_i\left|\gamma_i\right| \bmod 2^{n+1}T\right) - \left(s_i\left|\gamma_i\right| \bmod 2^{n}T\right)$), referred to as bit plane n.

With such an embedded code, quantization is performed, after scaling, by the above thresholding for thresholds at all 2^n. The minimum quantization is determined by the encoder and a maximum quantization by the decoder. The coefficients are coded out bit plane by bit plane for descending n. The scaling determines which bits are grouped together into bit plane n.

To review, an embedded encoding of the wavelet transform can be achieved by using descending powers of two as successively finer thresholds and sequencing the data bit plane by bit plane. Pre-scaling the data optimizes this approximation sequence for the desired norm.

SECTION IV ZEROTREE CODING FOR 2-MANIFOLDS

4.1 The Significance Function

So far the described coding achieves compression only by quantization (which also entails a corresponding approximation error). We have not really taken advantage of the concentration of the norm in a small number of coefficients. Most of the bits produced will be *leading zeros*, zeros which precede the most significant bit of the coefficient. Coefficients whose bit is *significant* (not a leading zero) are so much in the minority that the problem of designating which coefficients are significant far outweighs the problem of the value of those coefficients.

Shapiro [8] noticed that insignificant coefficients were typically clustered spatially and strongly correlated from between subdivision levels at the same spatial location. To make this concrete, he organized the coefficients into a tree structure with the property that one coefficient was below another if the former represented a subdivision refinement of the latter. He then formulated the *zerotree hypothesis* - that if a coefficient was insignificant then all coefficients below it are also insignificant. If the tree is constructed in a reasonable way the zerotree hypothesis will be true with very high probability. Thus, trees for which the hypothesis is true can be coded with very short codewords while long ones can be used in the rare cases where the hypothesis is false.

4.2 The G-tree

We now introduce a novel contribution of this paper, the *G-tree*, in order to apply the zerotree notion to the 2-manifold case. The cells (triangles, edges and vertices) of the tessellation of the 2-manifold are arranged into a tree, called a G-tree. Corresponding to the zerotree notion cells at coarser levels of subdivision are, in general, nearer to the G-tree root (which is the whole 2-manifold) than cells at finer levels of subdivision. Possible constructions for G-trees are described in detail later in the section. The original data and the coefficients of its wavelet transform are attached to a subset of the nodes of the G-tree. In this paper, data and coefficients will only be attached to vertices.

We need some notation and we will follow that of Said and Pearlman [6] (as well as using their coding variant of zerotrees).

The root of the G-tree (the whole manifold) is at the *top* of the G-tree. We will denote b below a in the G-tree by $a \supseteq b$.

The set of *descendants* of a is $D(a) = \{b \ni a \supseteq b, \quad b \neq a\}$ The set of *offspring* of a is $O(a) = \{b \ni (c \supseteq b \wedge c \in D(a)) \Rightarrow c = b\}$. These are the children of a. Following [6] the set $L(a)$ is defined as $L(a) = D(a) - O(a)$. We can now define the significance of a set A as $sig(A) = \max_{x \in A}(sig(x))$

We can make the following calculations recursively, proceeding from finer to coarser subdivision levels in the G-tree:

$$\begin{aligned} sig(D(a)) &= \max_{x \in O(a)}\left(\max(sig(x), sig(D(x)))\right) \\ sig(L(a)) &= \max_{x \in O(a)}\left(sig(D(x))\right) \end{aligned} \tag{6}$$

Shapiro's zerotree hypothesis can now be restated as

$$Prob(sig(D(a)) \leq sig(a)) \sim 1 \tag{7}$$

4.3 Structure of the coded file

We are now in a position to describe the overall structure of the coded file, using the method of Said and Pearlman [6]. We begin with a preamble that details the base complex, the subdivision method, the number of subdivision levels, and the scaling of the wavelet coefficients. The preamble information is sufficient to reconstruct the G-tree and the decoder initializes by reconstructing the G-tree from the preamble. Following

this are a sequence of bit planes, each representing the difference between the current approximation (with threshold T) and the next approximation (with threshold $T/2$) (see Fig 1). Each bit plane is a sequence of two sub-strings. The first of these are the *navigation* bits which describe the spatial location in the G-tree of the coefficients significant at this bit plane. The second are the bit plane coefficient bits (*refinement bits*).

The navigation bits for a bit plane result from a depth-first, left-to-right walk of the G-tree. However, terminal zerotrees (as identified by $sig(D(a))$ and $sig(L(a))$) are elided. This compactly encodes zerotrees, greatly reduces the encoding of leading zeros, and accounts for the bulk of the lossless compression of the method. The actual encoding method of the navigation bits proposed in [6] is quite clever. The tree walk proceeds by a sequence of binary decisions which guide the walk while avoiding zerotrees. The encoder outputs as navigation bits each binary predicate as the decision is made. The decoder then performs the same G-tree walk as the encoder by using exactly the same algorithm (except that it uses the next navigation bit as the next binary predicate).

The refinement bits for a bit plane contain one bit per coefficient significant at that bit plane. As a coefficient becomes significant at the bit plane corresponding to its significance, it is added to the end of a list of significant coefficients. The appropriate bit of each coefficient is read out by the encoder to form the refinements bits for the current bit plane. Coefficients are represented in sign-magnitude form, and since the position of the leading one is known it need not be read out. The sign bit is read out in its place.

The decoder reads the refinements bits and distributes one bit to each significant coefficient. When the decoder is finished the coefficients can be attached to their proper nodes in the G-tree.

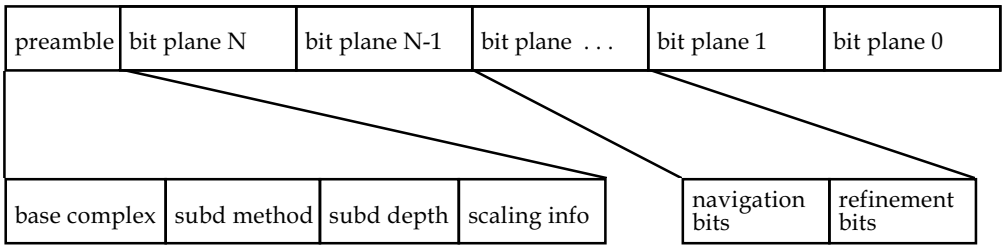


Figure 1 Structure of the encoded file.

4.4 Constructing the G-tree

In general, there are many reasonable ways of organizing into a G-tree the cells of recursively subdivided tessellation of a 2-manifold. We give here two such methods and utilize one of them for the remainder of this paper.

1) a) The root R of the G-tree is the 2-manifold itself
   b) $O(R)$ are the cells of the base complex
   c) $a \supseteq b$ if b arises in the recursive subdivision of a

$O(R)$ consists of the triangles, edges and vertices, all in the base complex. Using the subdivision scheme described in Section II (recall that other subdivision schemes are possible), we see that vertices are not subdivided (hence are leaves of the G-tree), edges subdivide into two edges and a vertex, and triangles subdivide into four triangles and

three edges. Hence a vertex is the offspring of an edge and a triangle is the offspring of another triangle. Edges may be the offspring of another edge or of a triangle.

2) a) The root R of the G-tree is the 2-manifold itself
   b) edges are omitted entirely from the G-tree
   c) $O(R)$ are the vertices and triangles of the base complex
   d) $a \supseteq b$ if b is a triangle arising in the recursive subdivision of a, another triangle
   e) if b is a vertex then b is a leaf node of the G-tree and $b \in O(a)$ for some uniquely chosen triangle a having the property that b is the midpoint of an edge of a

In other words, the G-tree initially consists of the triangles of the tessellation ordered by inclusion, with the vertices then assigned to be an offspring of some triangle of which the vertex is an edge mid-point. In general (except at the boundary of the 2-manifold), there will be two triangles with the vertex as an edge midpoint and the vertex will be arbitrarily assigned as the offspring of one of those two triangles. At the boundary there will be just one such triangle and the assignment will be unique.

For historical reasons, we use this second G-tree construction method in the remainder of the paper, although only the numerical results (not the algorithms) for the examples depend on this choice.

The actual algorithm that we designed is an adaptation of the SPIHT zerotree coding method that constructs the G-tree from the supporting 2-manifold.

SECTION V SIMULATION RESULTS AND APPLICATIONS

5.1 Experimental Setup

We have implemented the procedure described in the previous sections in C++ on a UNIX platform (SGI). The software, called "SW", extends a software package called "ttree" developed by Peter Schröder and Wim Sweldens at University of South Carolina[7]. The implementation requires that the entire data structure be kept in RAM. For simple base complexes (less than a couple dozen triangles) the 256 MB RAM of the SGI limits the subdivision level to seven or eight.

The compression algorithm follows closely the descriptions in Sections II, III, and IV. The user of "SW" can interactively select the type of wavelets to be used, the base complex , the number of levels of subdivision, the function to be compressed, and the desired compression. For the results below we have chosen two settings:

1) For comparison with existing data compression techniques (Said & Pearlman [6]) we used "SW" to compress the Lena image (the function being compressed is the gray-scale level). We specify a base complex consisting of two triangles, thereby forming a square. This was then subdivided nine times yielding a complex with 513^2 vertices and $2\text{x}512^2$ triangles. The function value at each of the 513^2 vertices was determined by interpolation from a 512^2 gray-scale Lena image. We used linear lifting (the new coefficients are the means of the two neighboring ones) and various compression ratios. Compression results are reported relative to the 513^2 data set.

• For the Earth example we compressed the topographic function that is the elevation (with respect to sea level) of the Earth. This function is initially approximated

by the ETOPO5 data set which samples the Earth every 5 arc minutes (the data file is 4.5 MB). This data set was 4:1 subsampled to the ETOPO10 data set which samples the Earth every 10 arc minutes (1.5 million points approximately 11 miles apart).

For the base complex we chose the icosahedron and subdivided it eight times yielding 655362 vertices and 1310720 triangles. The vertices of this complex were then radially projected onto a concentric geoid (vertices average about a 22 arc minute separation - about 25 miles). The topographic elevation at each vertex was then determined by interpolation of the ETOPO10 data set (about a 2:1 reduction in points).

This data was wavelet transformed using linear lifting and compressed at various ratios.

5.2 Compression Performance

Table 1 below summarizes the results for the peak signal-to-noise ratio (PSNR) for the two examples above for several different compression ratios. Scaling was chosen appropriate to the L_2 norm. The table reports the results relative to the interpolated vertex data. We can compare the performance curves in Figure 2. Figure 3 illustrate visually the quality of the compression. For calculating PSNR we are using the following formula which is similar to the one used in [6].

$$PSNR = 20 * \log_{10}\left(\frac{Max - Min}{L2}\right) \tag{8}$$

Here *Max* and *Min* are the largest and the smallest wavelet coefficients among all coefficients generated (for all levels), and *L2* is the L2-norm for the coefficients of the original image ($Orig_i$) and the ones of the processed image ($\Pr oc_i$), i.e. after compression and decompression of the original image.

$$L2 = \sqrt{\frac{\sum_i \left(Orig_i - \Pr oc_i\right)^2}{n}} \tag{9}$$

	LENA (9 levels, 263169 coeff.)			EARTH (8 levels, 655362 coeff.)		
bit planes	b/vertex	PSNR (dB)	Compression	b/vertex	PSNR (dB)	Compression
10	1.4029	38.83	6:1	0.5665	41.96	14:1
9	0.5428	34.25	15:1	0.1968	37.51	41:1
8	0.2533	30.39	31:1	0.0802	34.11	100:1
7	0.1177	27.11	68:1	0.0335	31.12	240:1
6	0.0505	24.18	160:1	0.0142	28.52	565:1
5	0.0211	21.64	382:1	0.0059	26.06	1356:1
4	0.0092	19.34	866:1	0.0027	23.98	2994:1
3	0.0032	16.71	2474:1	0.0014	22.03	5804:1
2	0.0014	15.09	5751:1	0.0008	19.85	9927:1
1	0.0007	13.97	10961:1	0.0007	18.65	11579:1

Table 1. PSNR data for the Earth (8 levels of subdivision, 1 vertex every 25 miles) and Lena (9 levels of subdivision) examples for different compression ratios.

The results that we obtained for the Earth data visually look exactly as what we would expect for good compression given the resolution that we are achieving. Since that is the first method that allows to do that type of compression for anything other than flat images (or sequences of such) it is difficult to judge our PSNR numbers.

On the other side the PSNR numbers for Lena are very similar to the ones reported for the embedded zerotree in [6] (there for compression 16:1 the PSNR is 33.79 dB). We should remember that the S&P method and algorithm from [6] is designed specifically for planar images to take advantage of that spatial structure. In addition there are some differences in the calculation for the PSNR for example, where we calculate the actual peak for all images used (rather using 255 by default as in [6]). Finally our method is specifically designed to deal with higher dimensional entities. In view of these considerations the numbers that we are reporting for comparison in the Lena case are surprisingly good. The code is run in a general setting without any adjustments or speed ups that might be included to take advantage of the flat geometry.

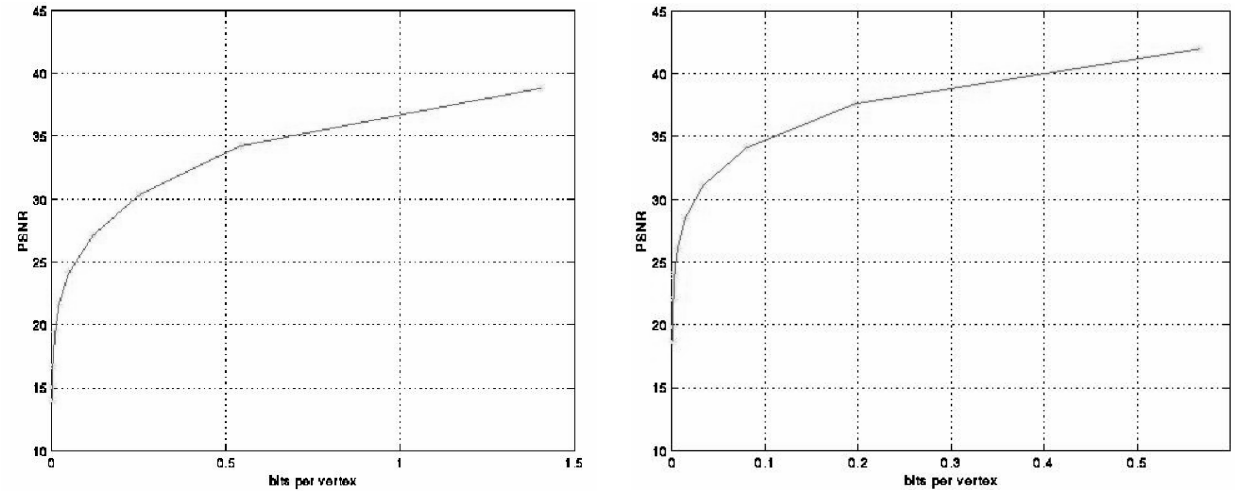


Figure 2. Performance curves: Lena with 9 levels of subdivision (left) and Earth with 8 levels (right).

5.3 Applications

Numerous applications for this technology exist. Those include Earth data compression; compression of data mapped on human heads; interactive multiresolution viewing of complex data sets; interactive multiresolution processing and editing of surfaces.

ACKNOWLEDGEMENTS

This paper is partially based on work done with Peter Schröder and Wim Sweldens during their visit at Interval Research Corporation in the summer of 1995.

REFERENCES

[1] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets, *Comm. Pure Appl. Math.*, 45:485--560, 1992.

[2] I. Daubechies. *Ten Lectures on Wavelets*, CBMS-NSF Regional Conf. Series in Appl. Math. Vol. 61. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[3] R. A. DeVore, B. Jawerth, and B. J. Lucier, "Image compression through wavelet transform coding", *IEEE Trans. Inform. Theory*, vol. 38, pp. 719-746, March 1992

[4] R. A. DeVore, B. J. Lucier, "Classifying the smoothness of images: Theory and applications to wavelet image processing", *Proc. of the 1994 IEEE Conf. on Image Processing*, IEEE Press, Vol. II, pages 6-10

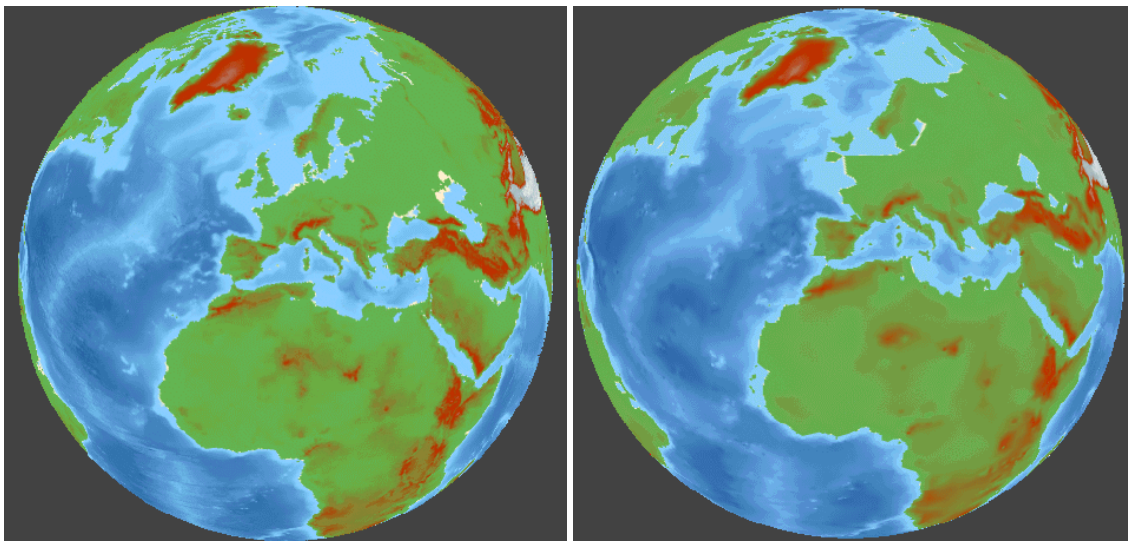
[5] S.G. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L_2(R)$, *Trans. Amer. Math. Soc.*, 315(1):69--87, 1989.

[6] A. Said and W.A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees, Submitted to the IEEE Transactions on Circuits and Systems for Video Technology.

[7] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere, *Computer Graphics Proceedings, (SIGGRAPH 95)*, pages 161--172, 1995.

[8] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients, 41(12):3445--3462, 1993, IEEE Trans. Signal Process.

[9] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets, Technical Paper 1994:7, IMI, Department. of Mathematics, University of South Carolina, 1994.



18 bit planes, 1:1 compression
PSNR = 83.83 at 9.26 bpv

9 bit planes, 41:1 compression
PSNR = 37.51 at 0.20 bpv

15 bit planes read
PSNR = 66.56 at 8.15 bpv

8 bit planes read
PSNR = 30.39 at 0.25 bpv

Figure 3. Compression results for the Earth topographical data and the Lena image.