WAVELET COMPRESSION FOR 3D AND HIGHER-DIMENSIONAL 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 of arbitrary topology. 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.

Keywords: compression, wavelets, manifolds, 3D objects, zerotree, multiresolution analysis, progressive coding

PART I INTRODUCTION

Digital computers have long been used to represent and predict aspects of the world around us. Such representations often take the form of geometric objects having interesting information associated with and distributed about that geometry. For example, the synthesis of real-world like scenes by computer graphics techniques often proceeds by approximating the geometry of the scenes by piecewise polygonal surfaces embedded in 3-space. These surfaces then have information about illumination, texture, etc. distributed about them. In this paper we focus on the representation of that distributed information, although it is clear that the techniques described herein are also applicable as well to the description of the geometry.

The growth in the capacity and computational power of computers over the years has enabled us to increase the scope and complexity of our computerized representations. Today modern computers are capable of dealing with quite complicated geometries having rich information distributed across them. However, our ambitions always exceed our capabilities, so we are always interested in techniques which substantially reduce the memory and computational loads required by a given situation (primarily so that we can deal with even more complex situations with the given computational resources).

Our computer representations are necessarily finitely generated, so we are always talking about representations that *approximate* the real world. Of course, we need to measure and control the degree of approximation appropriate to our specific application. A good approximation schemes will not only have appropriate and measurable fidelity, but will also be economical of both computational and memory resources.

Such approximations are often called *compression* schemes, because they greatly reduce storage requirements, but often also enable a great reduction in the computation required. It is generally under appreciated the extent to which a compressed representation will reduce computation just by the simple fact that the amount of data to be processed is greatly reduced.

1.1 Modeling data as a function

To be more precise, 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 (i.e., Euclidean 2-space), the half plane (an edge), or the quarter plane (a corner).

It is useful to consider the scalar function to be compressed as belonging to a normed vector space of functions. Such (Banach) spaces of functions are closed under linear combination and equip each function f with a norm $|f|$ satisfying the triangle inequality $|f|+|g|\geq|f+g|$.

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 Transform data compression

The transform compression of a function involves three steps, a transformation of the function, quantization, and entropy encoding. Only the quantization step actually discards information - the transformation and encoding steps have inverses and are reversible.

1) the function is subjected to a reversible linear transformation in order to concentrate most of its entropy (i.e., information) into a low dimensional subspace, thus simplifying its description. A wide variety of transformation techniques are currently in use, including the discrete cosine transform (DCT) and wavelet transforms.

2) the coefficients of the transformed function are quantized, i.e., scaled and truncated appropriately. Algebraically, this amounts to a projection of the function into one of the low dimensional subspaces in which the entropy is concentrated. Such a step discards the entropy (information) perpendicular to the projection - hopefully information that will not be required by an application. For example, if an image is being compressed for later viewing by the human perceptual system then information about high spatial frequencies can be attenuated or discarded on the grounds that such information will be imperceptible. Likewise in audio compression for human listening information at frequencies above 20 KHz. may be discarded as imperceptible.

3) the remaining coefficients after the quantization step are encoded using a suitable entropy encoding technique. Not all possible values of the coefficients are equally likely nor are all sequences of coefficients equally likely. Entropy encoding exploits statistics about the data to use shorter codes for the more frequent situations and longer codes for the less frequent situations. In particular, the transformation and quantization steps produce many zero values so that a major task of an entropy compression technique is to code the geometric location of these zeros. These locations are heavily correlated by location and scale and are compactly encoded by a quadtree-like technique known as *zerotrees* [8, 10].

A diagram of the compression and decompression processes is given in Figure 1.

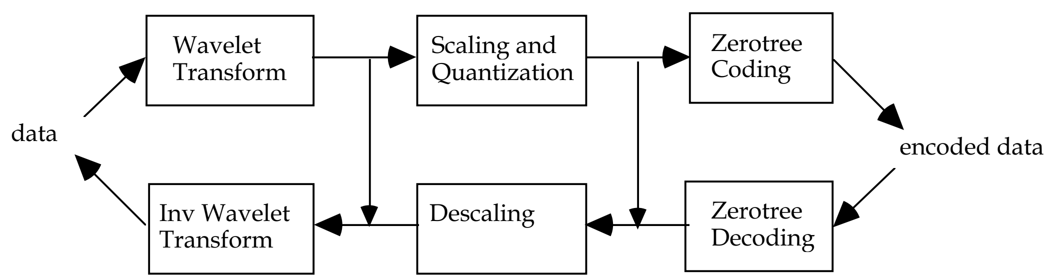


Figure 1. The compression - decompression process

1.3 Structure of the paper

In what follows in Part II we describe second generation wavelet transforms, as formulated by Sweldens [9], and discuss an efficient method for calculating such wavelets - the lifting scheme [11]. An illustration of this methodology will be demonstrated as wavelet transforms of a function defined on a 2-sphere [9]. In Part III we 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. In Part IV we review the zerotree coding method, focusing on an improved variant introduced in [8]. Adapting this method to coding the wavelet transform of a function on a general 2-manifold required a novel organization for the zerotrees, called G-trees, which we describe in detail. In Part 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. Finally Part VI describes some possible applications of this technology.

PART II WAVELET TRANSFORMS

2.1 Introduction

The basic ideas of wavelets and wavelet transforms were developed in the eighties by Yves Meyer, Ingrid Daubechies, and Stephane Mallat (see for example [1]). A rather vague definition of wavelets is: "*Wavelet are* ***building blocks*** *that can* ***quickly decorrelate*** *data.*" This sentence incorporates the three main features of wavelets.

2.1.1 Wavelets as a basis

First of all, they are **building blocks** for general data sets or functions. Mathematically we say that they form a basis. This means that each element of a general class can be written in a stable way as a linear combination of the wavelets. If we denote the wavelets by ψ_i and the coefficients by γ_i, we can write a general function f as

(1) $$f = \sum_{i \in I} \gamma_i \psi_i .$$

2.1.2 Wavelets decorrelate and efficiently approximate

Secondly, wavelets have the power to **decorrelate**. This means that the representation of the data in terms of the wavelet coefficients $\{\gamma_i\}$ is somehow more "compact" then the original representation. In other words the entropy in the wavelet representation is smaller then in the original representation.

The decorrelation power follows from the fact that the wavelets already in some way resemble the data we want to represent. More specifically, wavelets have the same correlation structure as the data. Many signals have both correlation in space and frequency. Indeed, samples which are spatially close are more correlated then ones that are far apart, and frequencies often occur in bands. Now wavelets precisely are local in space and frequency. Typically they have compact support (localization in space), are smooth (decay towards high frequencies), and have vanishing moments (decay towards low frequencies).

More precisely, the wavelet representation leads to rapidly converging approximations to functions that satisfy mild smoothness assumptions.

The function f can be approximated by

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

(3) $$\hat{\gamma}_{Ti} = \begin{cases} 0 & if\ |\gamma_i| \le T \\ sgn(\gamma_i)\left(|\gamma_i| - \left(|\gamma_i| \bmod T\right)\right) & if\ |\gamma_i| > T \end{cases} \qquad \text{and}$$

(4) $$\hat{I}_T = \left\{ i \middle| \hat{\gamma}_i \ne 0 \right\}$$

Note that $T_1 \le T_2 \Rightarrow \hat{I}_{T_1} \supseteq \hat{I}_{T_2}$. With mild smoothness assumptions on f (i.e., f is in some Besov space) the functions $\hat{f}_{2^{-k}}$ converge to f with increasing k and the sets $\hat{I}_{2^{-k}}$ are finite.

2.1.3 Wavelets are quick

Finally, the wavelet representation of a data set can be found quickly. More precisely, we can switch between the original representation of the data and its wavelet representation in a time proportional to the size of the data. The fast decorrelation power of wavelets is the key to compression applications.

2.2 Second generation wavelets

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 properties of first generation wavelets: basis properties, decorellation power, and fast algorithms.

In general, it has be shown that all finite classical wavelets may be represented using second generation wavelets. In other words, second generation wavelets are a superset of most all classical wavelets. Thus, the term as used herein refers not only to the specific second generation wavelets described, but also to those classical wavelets that may be represented using second generation wavelets.

The basic philosophy behind second generation wavelets is to build wavelets with all desirable properties (localization, fast transform) adapted to much more general settings than the real line, e.g., wavelets on manifolds. Such adaptations will depend on and vary with the local and global topological and geometric properties of the manifold. Adaptive constructions rely on the realization that translation and dilation are not fundamental to obtain the wavelets with the desired properties. The notion that a basis function may be written as a finite linear combination of basis functions at a finer, more subdivided level, is maintained and forms the key behind the fast transform. The main difference with the classical wavelets is that the filter coefficients of second generation wavelets are not the same throughout, but can change locally to reflect the changing (non-translation invariant) nature of the manifold.

A good example of adaptation to a local topological property, namely proximity to the manifold boundary, occurs even in the case of wavelets on the unit interval. The wavelets must be adapted (i.e., changed) near the endpoints [2].

2.3 The lifting scheme

First generation wavelet algorithms are usually constructed with the help of the Fourier transform. The underlying reason is that translation and dilation become algebraic operations on the Fourier transform. However, in the case of second generation wavelets, the Fourier transform can no longer be used as a construction tool.

Recently **the lifting scheme** [11], a new concept in the construction of wavelets has been introduced. It differs from the classical constructions in the sense that it does not need the Fourier transform. Therefore it is the perfect tool for the construction of second generation wavelets. We give here a short outline of the basic idea behind lifting.

A canonical case of lifting consists of three stages, which we refer to as: **split**, **predict**, and **update**. We describe here the basic idea behind each stage. Assume we start with an abstract data set, which we refer to as λ_0. We know this data set has some correlation structure and we would like to exploit it to obtain a more compact representation.

In the first stage we **split** the data into two smaller subsets λ_{-1} and γ_{-1}. We refer to γ_{-1} as the wavelet subset. The easiest possibility for the split is a simply brutal cut of the data set into two disjoint parts. This choice we refer to as the *Lazy wavelet*. Think for example of cutting an image into two parts with a pair of scissors.

In a second stage, we try to use the λ_{-1} subset to **predict** the γ_{-1} subset based on the correlation present in the original data. We construct prediction operator *P*, so that

(5) $$\gamma_{-1} \approx P(\lambda_{-1}).$$

We next replace γ_{-1} with the *difference* between itself and its predicted value $P(\lambda_{-1})$.

(6) $$\gamma_{-1} := \gamma_{-1} - P(\lambda_{-1}).$$

If the prediction is reasonable, this difference will contain much less information then the original γ_{-1} set. The wavelet subset now encodes how much the data deviates from the model on which *P* was built.

Often we want some global properties of the original data set to be maintained in the smaller versions λ_{-j}. For example, in the case of an image, we would like the smaller images λ_{-j} to have the same overall brightness, i.e. the same average pixel value. This is accomplished in the third stage. We here find a better λ_{-1} so that a certain scalar quantity *Q()*, like e.g. the mean, is preserved, or

(7) $$Q(\lambda_{-1}) = Q(\lambda_0).$$

To do so we use the already computed wavelet set γ_{-1} to update λ_{-1} so that the latter preserves *Q()*. In other words, we construct an operator *U* and update λ_{-1} as

(8) $\lambda_{-1} := \lambda_{-1} + U(\gamma_{-1})$.

At this moment we can replace the original data with the smaller set λ_{-1} and the wavelet set γ_{-1}. With a good prediction, the two subsets $\{\lambda_{-1}, \gamma_{-1}\}$ yield a more compact representation then the original set λ_0. We can now iterate this scheme. We split λ_{-1} into two subsets λ_{-2} and γ_{-2} and then replace γ_{-2} with the difference between γ_{-2} and $P(\lambda_{-2})$. After *n* steps we have replaced the original data with the wavelet representation $\{\lambda_{-n}, \gamma_{-n}, \ldots, \gamma_{-1}\}$. Given that the wavelet sets encode the difference with some predicted value based on a correlation model, this is likely to give a more compact representation. A nice properties of lifting is that once we have the forward transform, we can immediately derive the inverse. The only thing to do is to reverse the operations and toggle + and -.

2.4 Spherical wavelets

A typical example of the use of lifting to construct second generation wavelets are on a sphere [9]. The sphere is a 2-manifold (i.e., a surface). The construction starts with the tessellation of the sphere into cells (i.e., spherical triangles, their edges, and their vertices). This is done starting from a *base complex*, a triangulation of the sphere combinatorially isomorphic (and homeomorphic) to (the surface of) a pyramid, octahedron, or icosahedron.

This base complex is then recursively subdivided. Each edge is subdivided into two edges and a (midpoint) vertex and 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 subdivision schemes are possible.

Shown in Figure 2 is a once divided icosahedron containing a triangle 210 that represents one of the many triangles on the surface. Note that triangle 210 itself forms a part of the larger triangle 200 that is represented by triangles 211, 210, 212 and 213. It should also be noted that triangle 210 itself may be further subdivided according to the same scheme.

Each vertex has a data value (function value) attached to it. Other choices are possible (data values could have been attached to faces) but in this paper we will exclusively deal with values attached to vertices. This construction immediately allows a splitting of the vertices of a given level into an "even" and an "odd" subset as needed in the first stage of the lifting scheme. The even vertices are simply the ones from the coarser (i.e., pre-subdivision) level while the odds are the remaining ones. One thus has an asymptotic ratio of 1:3 for even vertices versus odd vertices. This gives a definition of a Lazy wavelet transform on the attached data values.

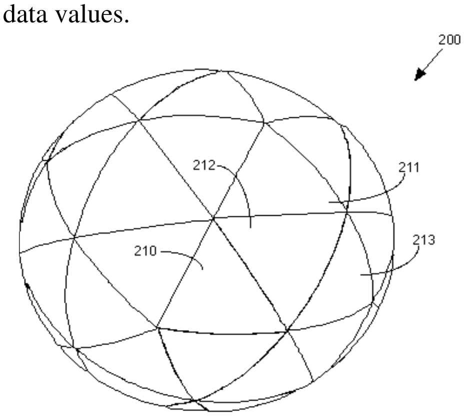


Figure 2. The first subdivision of an icosahedron

With two instances of lifting (prediction and update), one now obtains wavelets with the desired properties. We particularly consider two possibilities (out of many) for the prediction operator based upon Figure 3.

- Linear: Here we predict an odd value (denoted *m* in the figure) as the average of the two even values which reside at the endpoints of the edge (denoted v_1 and v_2 in the figure):

(9) $$m \approx \frac{1}{2}(v_1 + v_2).$$

• Butterfly: Inspired by [8], we predict the odd value here by a weighted average of 8 neighboring even values as follows:

(10) $$m \approx \frac{1}{2}(v_1 + v_2) + \frac{1}{8}(f_1 + f_2) - \frac{1}{16}(e_1 + e_2 + e_3 + e_4).$$

Many other prediction schemes can be constructed (for example quadratic, cubic, etc.). The actual wavelets and scaling functions can now be constructed by iterating the prediction operator. This process defines the scaling functions on a dense subset of the surface. This function is then extended to the whole surface by continuity. The wavelets are simple linear combinations of the scaling functions.

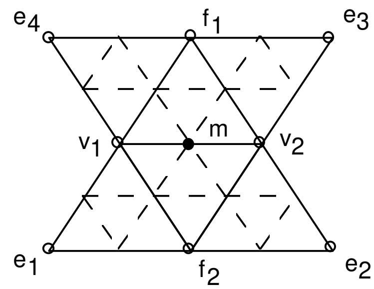


Figure 3. Odd vertex (m) surrounded by even neighbors $(\{v_1, v_2, f_1, f_2, e_1, e_2, e_3, e_4\})$ used in the prediction and update stages.

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). In [9] it is shown that with simple diagonal scaling of the wavelet coefficients, one can construct local smoothing and enhancement operators.

2.5 Wavelets on general 2-manifolds

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

2.6 Defining the function to be wavelet transformed

We have not yet addressed the issue of the origin of the data at the finest level of subdivision, the first stage of the lifting process. There are several ways that this data might be supplied.

2.6.1 As a function of manifold co-ordinates

The data might be supplied as a function of coordinates for the manifold. This process will be complicated by any singularities in the coordinate system. For example, the sphere is guaranteed to have at least two such singularities. It is for this reason that the sphere makes an interesting example, one that is inherently more complicated than a simple product space.

In any case the following steps are required to define a function by manifold co-ordinates:

a) The vertices of the triangulated manifold will have to be mapped to coordinates in the manifold.

b1) If the function is given on a dense subset it will have to be sampled at the coordinates corresponding to each vertex of the triangulated manifold.

b2) If the function is given only on a sample set, the coordinates of the samples may not be aligned with the coordinates of the vertices. In this case the function will have to be interpolated to the vertices.

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).

2.6.2 Attached to a complicated triangular mesh

A triangular mesh that is obtained by several levels of subdivision of a simple base complex is said to have *subdivision connectivity*. The function might be approximated by its sampled values on a complicated triangular mesh which does not have subdivision connectivity. Such a mesh does not lend itself to the lifting scheme. In [5] Eck et al use harmonic mapping techniques to interpolate such functions to a genus-equivalent triangular mesh that does have subdivision connectivity, enabling the lifting method described above.

2.6.3 By means of an oracle

The simplest approach is to assume that we have an external oracle which give us the data at the finest level. In this sense the previous two methods are possible oracles.

PART III SCALING, QUANTIZATION, AND APPROXIMATION RATE

3.1 Scaling

It is shown in [3] that an optimal approximation for differing L_p norms requires differing scaling of the wavelet coefficients prior to the thresholding operation above. (Equivalently, 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

Equations (2) and (3) defined an approximation to the function f. We can denote by

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

(12) $N = \left|\hat{I}_T\right|$ as 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

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

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 Ts 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 Ts 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, namely $\left(s_i\left|\gamma_i\right| \bmod 2^{n+1}T\right) - \left(s_i\left|\gamma_i\right| \bmod 2^n T\right)$), 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.

PART IV ZEROTREE CODING

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.

To deal with this situation, Shapiro [10] defined the *significance* of a coefficient γ_j as

(14) $$sig(\gamma_j) = \lfloor \log_2 |\gamma_j| \rfloor$$

A coefficient γ_j is significant at bit plane n if $sig(\gamma_j) \geq n$ and insignificant at bit plane n otherwise. The bit position of the leading one of γ_j is thus given by $sig(\gamma_j)$.

Shapiro [10] 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 codewords 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 in a later 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 nodes that are vertices.

We need some notation and we will follow that of Said and Pearlman [8] (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 [13] 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 now make the following calculations recursively, proceeding from finer to coarser subdivision levels in the G-tree:

(15)
$$sig(D(a)) = \max_{x \in O(a)} \left(\max\left(sig(x), sig(D(x)) \right) \right)$$
$$sig(L(a)) = \max_{x \in O(a)} \left(sig(D(x)) \right)$$

Shapiro's zerotree hypothesis can now be restated as

(16) $$Prob\left(sig(D(a)) \le sig(a) \right) \sim 1$$

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 and their algorithm SPIHT [8]. 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 is a sequence of bit planes, each representing the difference between the current approximation (with threshold T) and the next one with threshold $T/2$).

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 [8] 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 exactly the same G-tree walk as the encoder by using exactly the same algorithm, with the exception of using 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.

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 midpoint subdivision scheme (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 triangles. At the boundary there is just one such triangle and the assignment is unique.

We will use the first G-tree construction method and a subsequent arithmetic coding of the bitstream in the example below.

PART V RESULTS

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. The compression algorithm follows closely the descriptions in Parts 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 [8]) 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 and various compression ratios. Compression results are papered 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 that 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 then wavelet transformed using butterfly lifting and compressed at various ratios. The resulting bitstream for both data sets was further encoded using arithmetic coder [12].

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 4 and Figure 5. Figures 6 illustrate visually the quality of the compression. For calculating PSNR we are using the following formula which is similar to the one used in [8].

$$(17)\qquad PSNR = 20 * \log_{10}\left(\frac{Max - Min}{L2}\right) \quad \text{where} \quad L2 = \sqrt{\frac{\sum_i \left(Orig_i - \Pr oc_i\right)^2}{n}}$$

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.

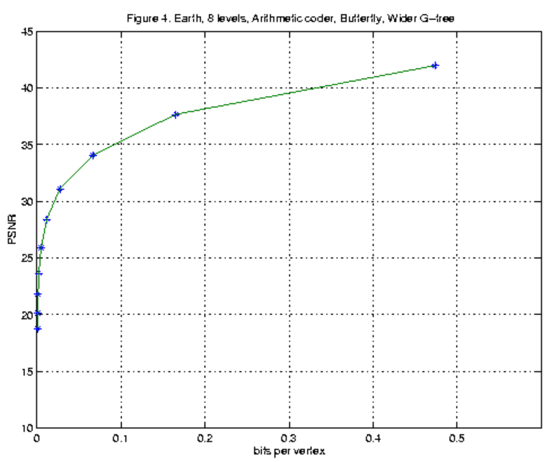
The results that we obtained for the Earth data visually look exactly as what we would expect for good compression given the resolution that we are achieving. Since that is the first method that allows to do that type of compression for anything other than flat images (or sequences of such) it is difficult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

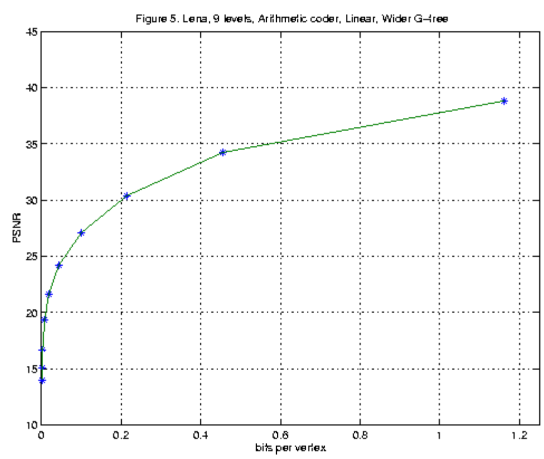
On the other side the PSNR numbers for Lena are very similar to the ones reported for the embedded zerotree in [8] (there for compression 16:1 the PSNR is 33.79 dB). We should remember that the S&P method and algorithm from [8] is designed specifically for planar images to take advantage of that spatial structure. In addition there are some differences in the calculation for the PSNR for example, where [8] explicitly uses 255 as the peak for all images calculated, as opposed to the actual peak that we calculate. Our method is primarily higher dimensional, thus accounting for all that the numbers that we are papering 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.

In addition the results reported below are about 12% better on the average than our previous results reported in [6] . The main difference is due to the use of the first G-tree structure from Section 4.4 (rather than the second one used in [6]). The addition of arithmetic coder brought only about 4% improvement, similarly to the use of Butterfly vs. Linear scheme for the Earth data (see [7]).

	LENA (9 levels, 263169 coeff.)			EARTH (8 levels, 655362 coeff.)		
bit planes	b/vertex	PSNR (dB)	Compression	b/vertex	PSNR (dB)	Compression
10	1.1612	38.83	6.4:1	0.4734	41.98	17:1
9	0.4557	34.25	16:1	0.1651	37.68	48:1
8	0.2139	30.39	33:1	0.0665	34.08	120:1
7	0.1003	27.11	71:1	0.0275	31.12	291:1
6	0.0434	24.18	162:1	0.0117	28.43	686:1
5	0.0184	21.64	387:1	0.0049	25.89	1623:1
4	0.0082	19.34	889:1	0.0022	23.66	3659:1
3	0.0029	16.7	2543:1	0.0013	21.81	6402:1
2	0.0014	15.09	5800:1	0.0008	20.12	10006:1
1	0.0007	13.97	11023:1	0.0007	18.76	11599:1

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.





PART VI APPLICATIONS

Numerous applications for this technology exists. Below we will touch upon a few of them.

- *Earth data compression*: Very large global and planetary datasets exist. A typical example is the ETOPO5 data set used as a test example. Even with powerful workstation manipulating, storing, and transmitting these data sets can be quite expensive. With our technology considerable speedup can be obtained.

- *Human head compression*: Current video phone technology takes advantage of the fact that mostly human heads have to be transmitted. One technique consists of first transmitting previously acquired 3D models of the persons heads and later only transmitting the moves of the head. Our technology can be used to considerably speed up the first stage. Other areas of application that use human head models include animation and multimedia.

• *Interactive multiresolution viewing of complex data sets*: When building an interactive viewer for complex volumes one can make use of the embedded nature of the encoding to only load into memory and display the information at the resolution actually required. For example, if the user needs a global view of the data a low resolution approximation can be used. When the user focuses on a particular area more local level of detail can be loaded and displayed in real time.

• *Interactive multiresolution processing and editing of surfaces*: The multiresolution structure of the embedded encoding allows very flexible ways of interactively building and editing complex 3D models. In case one wants to change a global feature of the model, one would modify a coarse level coefficient while keeping the finer resolutions unchanged [14]. On the other hand, if one wants to edit the finer details, this can be done by locally changing the finer level coefficients while keeping the coarser levels unchanged. This has applications in computer graphics and modeling.

REFERENCES

[1] 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.

[2] A. Cohen, I. Daubechies and P. Vial, Wavelets on the interval and fast wavelet transforms, *Applied and Computational Harmonic Analysis* 1, pp. 54-81, 1993.

[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] N. Dyn, D. Levin, and J. Gregory. A butterfly subdivision scheme for surface interpolation with tension control, *Transactions on Graphics*, 9(2):160--169, 1990.

[5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes, *Computer Graphics Proceedings, (SIGGRAPH 95)*, pages 173--182,1995.

[6] K. Kolarov and W. Lynch. Compression of functions defined on surfaces of 3D objects*, Proceedings of the Data Compression Conference*, Snowbird, Utah, pages 281--291, March 1997.

[7] K. Kolarov and W. Lynch. Optimization of the SW algorithm for high-dimensional compression, *Proceedings of SEQUENCES'97* , Positano, Italy, June 1997.

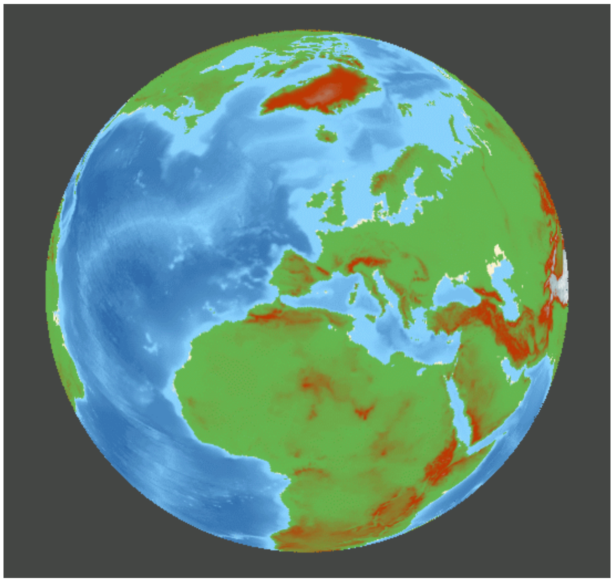
[8] A. Said and W.A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees, Submitted to the IEEE Transactions on Circuits and Systems for Video Technology.

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

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

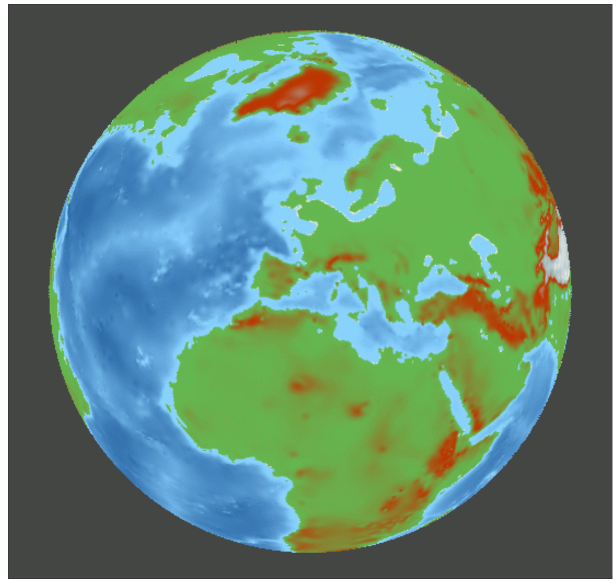
[11] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets, Technical Paper 1994:7, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, 1994.

[12] I. Witten, R. Neal and J. Cleary. Arithmetic coding for data compression, In Communications of the ACM, v. 30, n. 6, pages 520-540, June 1987.



19 bit planes, 1:1 compression

PSNR = 89.92 at 7.97 bpv



9 bit planes, 48:1 compression

PSNR = 37.68 at 0.17 bpv

17 bit planes read,

PSNR = 78.58 at 8.10 bpp

8 bit planes read,

PSNR = 30.39 at 0.21 bpp

Figure 6. Earth data (Butterfly base, 8 levels of subdivision, arithmetic coder) and Lena data (Linear base, 9 levels of subdivision, arithmetic coder)