Nonrectangular wavelets for multiresolution mesh analysis and compression

Kıvanç Köse, A. Enis Çetin, Uğur Güdükbay, and Levent Onural

Bilkent University, 06800 Bilkent, Ankara, Turkey

ABSTRACT

We propose a new Set Partitioning In Hierarchical Trees (SPIHT) based mesh compression framework. The 3D mesh is first transformed to 2D images on a regular grid structure. Then, this image-like representation is wavelet transformed and SPIHT is applied on the wavelet domain data. The method is progressive because the resolution of the reconstructed mesh can be changed by varying the length of the 1D data stream created by SPIHT algorithm. Nearly perfect reconstruction is possible if full length of 1D data is received.

Keywords: 3D Meshes, 2D Image like Mesh Representation, Wavelet Transform

1. INTRODUCTION

Meshes are widely used in computer graphics to represent and visualize 3D objects. The mesh representations of 3D objects are either created manually or using 3D scanning and acquisition techniques. Meshes with arbitrary topology can be created using these methods. The 3D geometric data is generally represented using two tables: *a vertex list* storing the 3D coordinates of vertices and *a polygon list* storing the connectivity of the vertices. The polygon list contains pointers into the vertex list.

Multiresolution representations can be defined for 3D meshes. In a fine resolution version of an object more vertices and polygons are used as compared to a coarse representation of the object. It would be desirable to obtain the coarse representation from the fine representation using computationally efficient algorithms. Wavelet-based approaches are applied to meshes to realize a multiresolution representation of a given 3D object.

There exist several mesh compression algorithms in the literature [1, 14, 16]. They can be classified as *progressive mesh compression* and *single-rate compression.* Progressive mesh representations [11] store a 3D object as a coarse mesh and a set of vertex split operations that can be used to obtain the original mesh. Progressive mesh representations use simplification techniques such as edge collapse and vertex removal to obtain the coarser versions of the original mesh. Subdivision technique can also be regarded as a progressive mesh representation [4, 12]. In [2, 13], multiresolution approaches for mesh compression using wavelets are developed. There also exists some compression algorithms that are designed only for dealing with geometry or connectivity of the mesh [17].

In this paper a wavelet-based mesh compression framework is proposed. The proposed mesh compression method is based on Set Partitioning In Hierarchical Trees (SPIHT) encoding of wavelet domain data [19, 20]. 2D planes $\mathbf{n}$ defined by $\mathbf{n_1}$ **x** $\mathbf{n_2}$ are regularly sampled with a user specified sampling rate in $\mathbf{n_1}, \mathbf{n_2}$ directions. These sampled planes are called grids. The vertex data are represented on grids in such a way that some nodes of the grid have a specific value related to the actual positions of the vertices in 3D space. The framework for mesh compression using SPIHT coding of wavelet domain data is given in Figure 1.

The value of a typical node of the grid is closely related to the perpendicular distance of the vertices to the plane of projection. The value taken by the grid node changes according to the chosen projection plane in 3D space. For example, if the projection plane is assumed to be on the x-y plane, the values taken by grid points are determined by the z-component of the nearest actual mesh node location.

Further author information:
Kıvanç Köse; E-mail: kkivanc@ee.bilkent.edu.tr, A. Enis Çetin; E-mail: cetin@bilkent.edu.tr,
Uğur Güdükbay; E-mail: gudukbay@cs.bilkent.edu.tr, Levent Onural; E-mail: onural@ee.bilkent.edu.tr
This work is supported by European Commission Sixth Framework Program with Grant No: 511568 (3DTV Network of Excellence Project).

Independent Component Analyses, Wavelets, Unsupervised Smart Sensors, and Neural Networks IV,
edited by Harold H. Szu, Proc. of SPIE Vol. 6247, 624704, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.666702

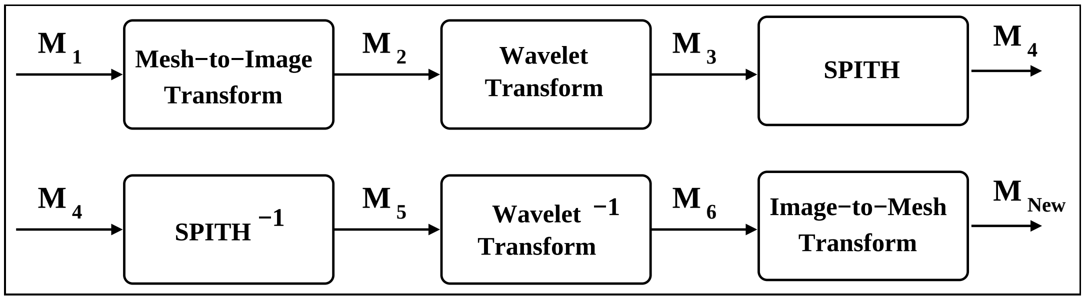


Figure 1. The proposed framework for mesh compression. M_i represents the mesh data at stage i.

Remeshing is a popular method for converting an irregular mesh to a semiregular or regular mesh. The idea in remeshing is representing the same object with more correlated connectivity and vertex position information while causing the least distortion. This makes the new mesh more compressible since many of the vertices can be predicted from its neighbors. It can be thought as regularizing the mesh data. However, remeshing is a complex task. Here we use an idea similar to remeshing. By *mesh-to-image transform* stage the positions of the vertices are quantized so that the data becomes more regular. During this transformation *rectangular* and *quincunx* or other grid structures may be used. In Figure 2 the sampling schemes of rectangular (a) and quincunx (b) grids are shown. Quantization using quincunx grids causes less error than quantization using rectangular grids.

The transformation steps are used for converting the 3D mesh to a representation on which signal processing methods can be applied. Two popular approaches to apply signal processing methods to surface meshes are parameterization of meshes and directly using modified versions of the signal processing algorithms, like relaxation. Parameterization of surface meshes is a complex task to accomplish for arbitrary objects because many linear equations should be solved. As the objects get more complex the parameterization operation becomes nearly impossible. In the proposed method the complexity of the transformation does not severely change for complex objects. The adaptation of all signal processing algorithms to surface meshes is also a challenging task although it is easier than parameterization. It is much easier to transform the data and apply any algorithm that is needed as compared to adapting signal processing algorithms.

This paper is organized as follows. The techniques used in the implementation of the mesh compression framework are explained in Section 2. A comparison of our approach with other approaches is given in Section 3. The simulation and comparison results are presented in Section 4. Conclusions are given in Section 5.

2. THE MESH COMPRESSION FRAMEWORK

A mesh can be considered as an irregularly sampled discrete signal $\mathbf{S}[\mathbf{d}]$ where $\mathbf{d} \in$ **vertex-list**. This means the signal is only defined at vertex positions. In our approach, the mesh is converted into a regularly sampled signal by putting a grid structure in space that correspond to resampling the space with a known sampling matrix and quantization. Then the data is wavelet transformed and the wavelet domain data is encoded using SPIHT. In the reconstruction stage the original mesh is recovered from the encoded data by using inverse operations, which are expansion and backprojection.

2.1. Initialization and Projection on the Grid Structure

First, we normalize the 3D mesh. In the normalization stage, the vertex positions are scaled to [-0.5, 0.5] interval in every axis so that the mesh fits into a unit cube centered at the origin. After this step, image sizes that are related to the number of vertices in the mesh are determined. Images are the planes on which the 3D mesh will be projected. The plane used is a unit square that is sampled with a sampling period *gridstep* in each dimension. At most N^2 vertices can be positioned in this image without coinciding with each other where N is the number of samples in each dimension. Each pixel represents a point on the grid structure. The vertex-to-pixel correspondence is determined by $\lceil \mathbf{p_p}/gridstep \rceil$ where $\mathbf{p_p}$ is the orthogonal projection of a normalized vertex point $\mathbf{p} = [p_x, p_y, p_z] \in R^3[-0.5, 0.5]$ on the selected image plane. Including a detail level factor, the *gridstep* value would be $1/N_{gridstep}$ where $N_{gridstep}$ is the number of gridsteps. Number of gridsteps is equal to $\sqrt{M}.detaillevel$ where M is the number of vertices in the mesh and the constant $detaillevel \in R > 1$.

All the pixels of the image, namely the grid points, can be appropriately indexed using the grid step value. The coordinates of the grid points $\mathbf{x_n} = \mathbf{Vn}$ where $\mathbf{x_n}$ is the representation of the grid points in the coordinate plane that uses $\hat{\mathbf{n_1}}, \hat{\mathbf{n_2}}$ as the basis vectors and $\mathbf{n} = [\mathbf{n_1}, \mathbf{n_2}], n_i \in Z$ and $\mathbf{V}$ is one of the 2D sampling matrices as seen in Figure 2 (a) and (b). The parameter $\mathbf{T}$ in sampling matrix is the gridstep value. A vertex position $\mathbf{p} = [p_x, p_y, p_z] \in R^3[-0.5, 0.5]$ is orthogonally projected to the plane $\mathbf{n}$ created by vectors $\mathbf{n_1}, \mathbf{n_2}$. The resulting vector $\mathbf{p_p} = \mathbf{p}.\mathbf{n}$ gives the orthogonal projection of the vertex position on the selected plane. The $\mathbf{n_1}, \mathbf{n_2}$ components of the $\mathbf{p_p}$ vector are $\mathbf{n_1}, \mathbf{n_2}$ component values of the nearest grid points. Every vertex is assigned to a grid point $\mathbf{p_p}$ on the selected plane with these operations. Multiple vertices can be assigned to the same grid point. This problem will be dealt with the reconstruction phase.

Since the grid points are pixels on the image, besides their position they should have some value. The values of the grid points also comes from the vertices assigned to them. The pixel values $\mathbf{p_p}$ is equal to $|\mathbf{p}.\mathbf{m}|$ where $\mathbf{m} = \mathbf{n_1}\ \mathbf{x}\ \mathbf{n_2}$. In Figure 2 (c) the plane of projection and the vector $\mathbf{m}$ is shown.

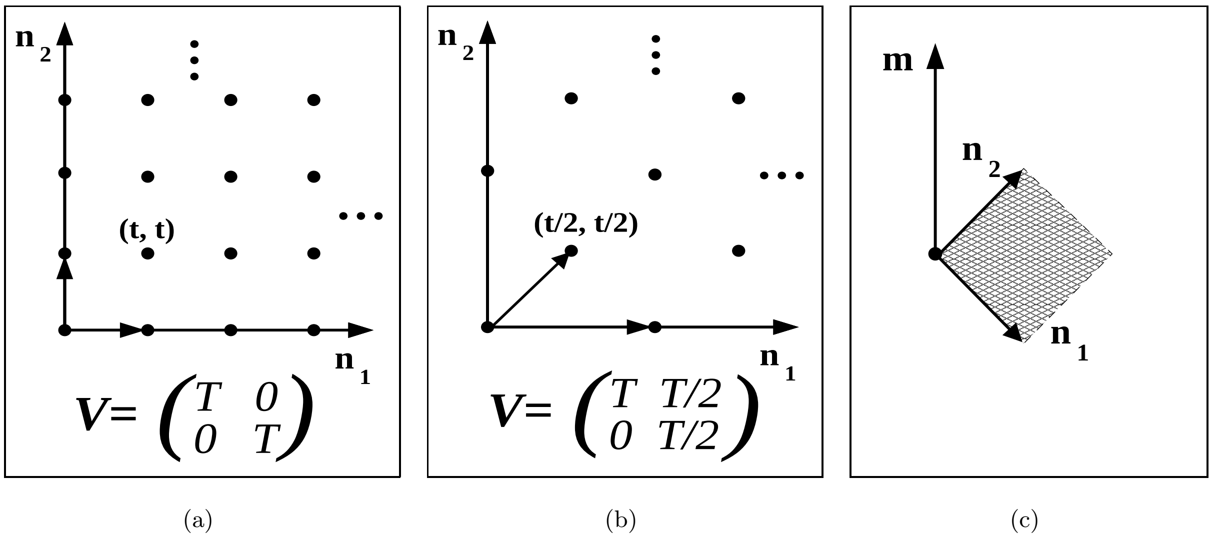


Figure 2. (a) 2D rectangular sampling lattice and 2D rectangular sampling matrix (b) 2D quincunx sampling lattice and 2D quincunx sampling matrix (c) m is the value of the grid point.

By applying the above operations on each vertex, we obtain a new image-like representation of the mesh. We find the orthogonal projection of the 3D mesh on user defines number of planes. Increasing the number of the projection planes lead to better reconstruction quality since lowers the need of interpolation of coincided vertices. However, it also decreases the compression ratio.

In addition, we have a second image for each projection plane that contains the information of which vertex is assigned to which grid point. Same construction technique is used to construct these images but instead of assigning the $|\mathbf{p}.\mathbf{m}|$ values to the grid points in the second image, the vertex number is assigned to them. In this way, each grid point stores the perpendicular distance to the specified vertex in the first channel and the index of the vertex in the second channel.

2.2. Image-like Representation of Meshes

After the projection phase, a new image-like representation of the mesh is formed. The pixels of the image are grid points of the new representation. The values of the pixels are related to the perpendicular distance between the pixel and the vertex assigned to it. The pixel-vertex matching procedure is closely related to the position of the vertex in space.

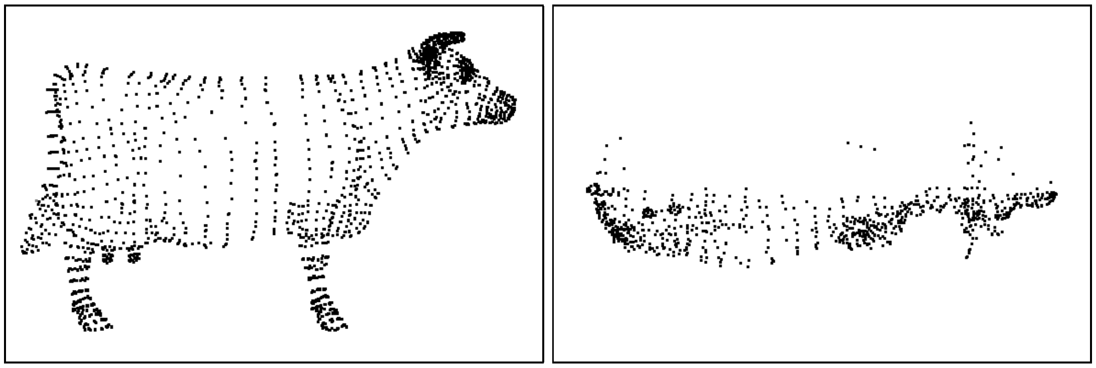

(a) (b)

Figure 3. Image-like representation of the mesh; (a) projected on XY plane; (b) projected on XZ plane.

The most crucial point in this representation is that most of the image pixels are empty that means they are assigned zero. At most M non-zero pixels may exist where M is the number of the vertices in the 3D mesh. This kind of representation is suitable for SPIHT coding because the multiresolution analysis of this representation will result in large zerotrees. Pixels that do not represent grid points are called "insignificant" pixels.

The image-like representation of a 3D object is shown in Figure 3. Black pixels correspond to the vertices of the mesh. Most of the image contains empty grid points.

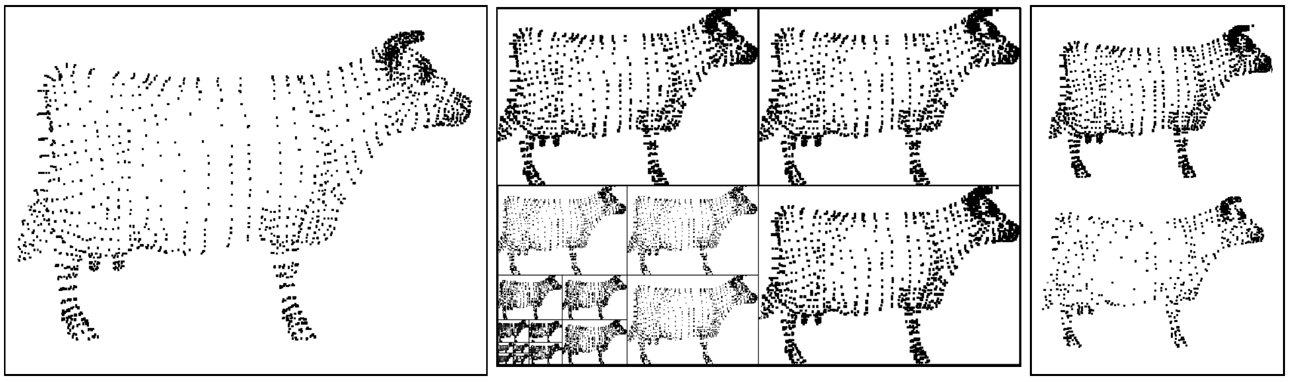
2.3. Embedded Zero-Tree Wavelet Coding and Set Partitioning In Hierarchical Trees

Multiresolution image analysis is widely used in image and video processing. Wavelets are very efficient in multiresolution analysis of signals because of their orthogonal or biorthogonal nature. In Discrete Wavelet Transform (DWT), a 1-D discrete signal is processed by a filterbank having a complementary half-band low-pass and high-pass filter pair. Outputs of the two filters are downsampled and two subsignals are obtained. The high-band subsignal contains the first level wavelet coefficients of the original signal corresponding to the normalized frequency range of $[\pi/2, \pi]$. The low-band subsignal is processed using the same filterbank once again and the second level wavelet coefficients covering the frequency range of $[\pi/4, \pi/2]$ and the low-low band subsignal covering $[0, \pi/4]$ are obtained. In a typical application, this process is repeated several times. Extension of the 1-D DWT to the 2-D signals can be carried out in a separable and non-separable manner. In the separable case, the 2-D signal is processed first horizontally row by row by the 1-D filterbank and two subimages are obtained. These two subimages are then filtered column by column by the same filterbank and, as a result, four subimages, low-low, high-low, low-high and high-high subimages are obtained (the order of filtering is immaterial). High-band subimages contain the first-level wavelet coefficients of the original 2-D signal. The low-low subimage can be processed by the 2-D filterbank recursively.

Extension of a 1-D DWT can be carried out into 2-D case in a non-separable manner as well and 2-D signals in quincunx grids can be analyzed using the fast wavelet transform [9].

Figure 4 shows an image (a) and its bands of the 4-level wavelet transform (b). The Embedded Zero-Tree Wavelet Coding (EZW) takes advantage of the zeros in multi-scale wavelet coefficients or a 2-D signal [20]. Wavelet coefficients corresponding to smooth portions of a given signal are either zero or close to zero. Wavelet coefficients only differ from zero around edges of a given image. Based on this assumption, zeroing some of the small-valued wavelet coefficients would not cause much distortion while reconstructing the image. EZW also takes advantage of the relation between multiresolution wavelet coefficients obtained using the 2-D wavelet tree. In Figure 4 (c), the image reconstructed after zeroing some parts of the bands of the image in Figure 4 (b) is shown. As it is seen, there is insignificant distortion in the reconstructed image.

A typical natural image is low-pass in nature. As a result most of the wavelet coefficients are zero or close to zero except the coefficients corresponding to edges and textures of the image. On the other hand the data in



(a) (b) (c)

Figure 4. (a) Original image (b) 4-level wavelet transformed image (c) Reconstructed image. Produced by Matlab Wavelet Demo.

image-like signals that we extract from meshes are sparse. Most of the grid points have zero values as there are very few vertices around smooth parts of the mesh. Filtering the mesh using a regular low-pass or a high-pass filter of a wavelet transform spreads an isolated mesh value in the wavelet subimages. One can use the lazy filterbank of the lifting wavelet transform shown in Figure 5 in which the filter impulse responses are trivial, $h_l[n] = \delta[n]$ and $h_h[n] = \delta[n-1]$. Therefore, the use of the lazy filterbank is more advantages than the regular filterbanks in mesh images. The use of the lazy filterbank only rearranges the mesh image data into a form that EZW or SPIHT can exploit. As in the case of a natural image most of the high-band wavelet coefficients turn out to be zero and there is high correlation between the high-band subimages.

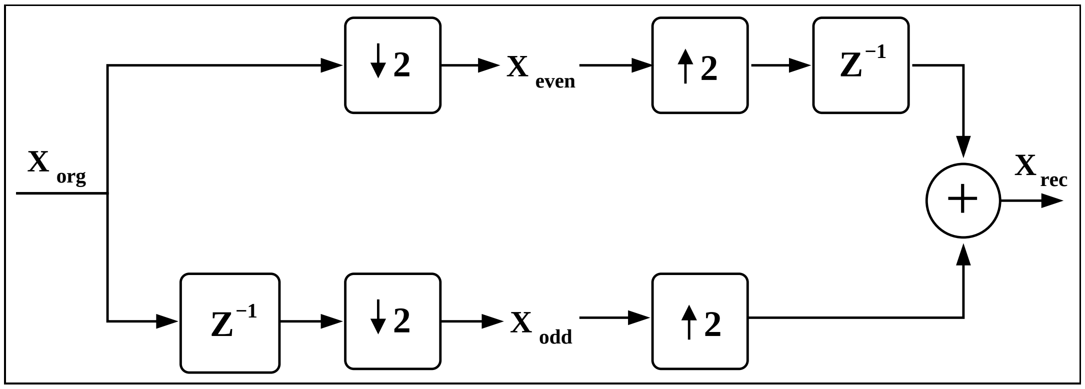


Figure 5. Lazy filter bank.

EZW, which is the predecessor of SPIHT, is an algorithm based on wavelet transform. An embedded code is a list of binary decisions. The original signal is obtained from an initial signal by those decisions. For example, think of an empty canvas as an initial signal. The performed operations will correspond to painting the canvas according to the decision list. The crucial part about embedded coding is the decisions are ordered according to their importance. This makes EZW a coder that can make a progressive compression-like coding.

The bitstream can be cut anywhere according to the users' needs. As the longer parts of the bitstream are chosen, the reconstructed initial signal gets closer to the original signal. Using the canvas analogy, the artist starts by painting the background objects and then adds the foreground objects. Finally, the details of the objects are added. The order that the painter uses is like an embedded code. If the painter do not paint the details of the image, it would be like a low-detailed, i.e., low-pass filtered image.

There exists a dependency between the wavelet coefficients, as shown in Figure 6 (a). This is the root-node hierarchy. The roots correspond to lower bands and the nodes correspond to higher bands. In most of the cases, the coefficients in the lower-level nodes are smaller than the coefficients in their roots. Figure 6 (b) shows the tree structure of the same hierarchy.

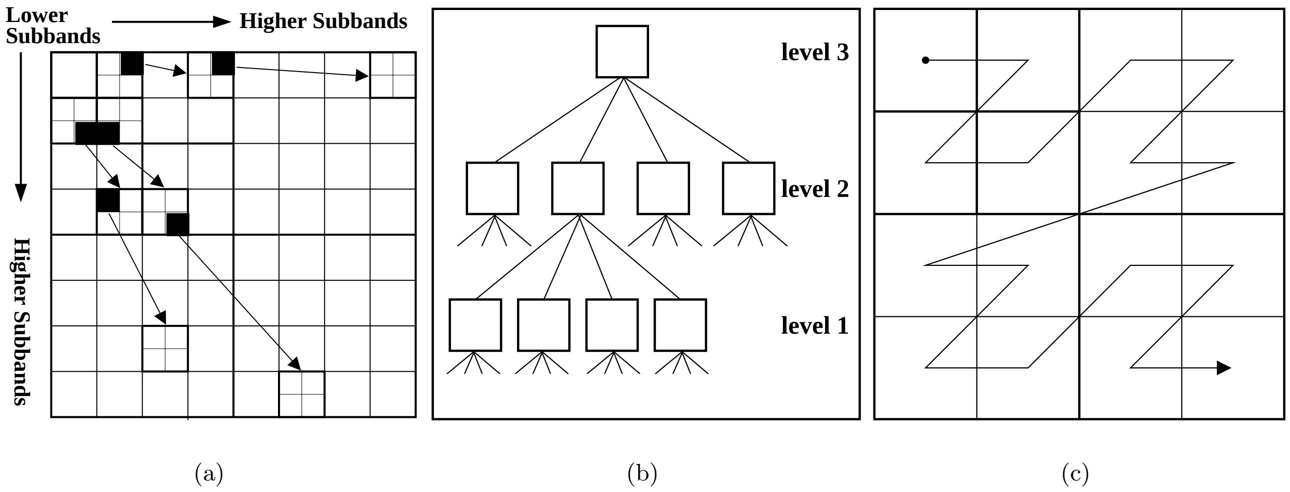


Figure 6. (a) Relation between wavelet coefficients in EZW; (b) tree structure of EZW; (c) scan structure of wavelet coefficients in EZW.

When a coefficient in the root is smaller than a threshold, its nodes contain even more smaller values. Thus, this branch of the tree, which is called a *zerotree*, can be pruned and coded easily. EZW checks for zerotrees in the transformed image and codes them with a special symbol.

In principle SPIHT carries out the same procedure. It takes the wavelet transform of the image until a user-defined scale is reached. It creates three lists by scanning the wavelet domain image in the order shown in Figure 6 (c). These lists are List of Significant Pixels (**LSP**), List of Insignificant Pixels (**LIP**), and List of Insignificant Sets (**LIS**). First, the coefficients in LIP are compared to a chosen threshold. If the value of the coefficient is bigger than the threshold it is put into LSP. The same procedure is carried out for the coefficients in LIS. This is called the *sorting pass*.

In *refinement pass*, n^{th} most significant bit of each entry in LSP is transmitted except those included in the last sorting pass [3]. The details of EZW and SPIHT can be found in [19–21].

2.4. Compression of the Images

A 3D mesh is represented by two image-like signals by applying the proposed mesh-to-image transform. As these signals are on a regular grid they can be compressed using the 2-D SPIHT. There are two issues that need to be decided while compressing these two image-like signals. The first one is choice of the length of the bit-stream and the second parameter is the number wavelet decomposition levels. Decreasing the length of the bitstream leads to more compression in the 3D mesh at the expense of higher distortion. Increasing the number of wavelet decomposition levels usually leads to higher compression ratios at the expense of more computational cost.

The distortion level of the reconstructed 3D mesh is measured visually or using some tools like METRO [5]. *Mean Square Error* (MSE) and *"Hausdorff Distance"* between the original and the reconstructed object are mostly used error measures in the literature.

The issue of how much of the bit-stream should be taken is closely related to the detail level parameter used in the orthogonal projection operation. If the detail level is low, the percentage of the used bit-stream must be increased to reconstruct the 3D mesh without much distortion. Pruning the bits at the end of the stream leads to less distortion than pruning the bits in the initial part of the stream.

Estimation of the number of levels of the wavelet decomposition is easier as compared to determining how much of the bitstream should be taken without introducing visual distortion. It is better to increase the number of scales in the wavelet decomposition as much as possible. This is because the data mostly contains insignificant pixels and as we get higher on the scales, the chance of having larger zerotrees is higher. Having larger zerotrees leads to better compression levels.

After the bitstream is obtained by the SPIHT encoder, it should be arithmetically coded. We used *gzip* software as an arithmetic coder [6]. It is an implementation of Lempel-Ziv coding algorithm [22]. Both the original vertex list and the SPIHT bitstream are compressed using *gzip* software for comparison purposes.

2.5. Reconstruction of a Mesh from the SPIHT Bitstream

The 3D mesh is reconstructed using the SPIHT bitstream and some other side information, such as the wavelet basis that is used, the vertex indexes (the second channel), the detail level used in the image-like representation. The volume of the side information is much smaller compared to the geometric data.

The bitstream is transformed to image-like representation using the SPIHT decoding. Then, using the projected vertex coordinates and the vertex indices, the image is back-projected to the 3D space. Since the only exact data available is the orthogonal component $\mathbf{m}$ of the 3D mesh vertices, the mesh can not be perfectly reconstructed. The $\mathbf{n_1}, \mathbf{n_2}$ components of the mesh are quantized to the grid point locations $\mathbf{x_n}$. The normalized 3D coordinates $\mathbf{p}$ of the mesh can be reconstructed without using any extra data.

Some of the vertices can be coincided while they are projected onto the $\mathbf{n}$ plane. Two methods are used to recover these vertices. One of them is to use a second plane to send the data of the coincided vertices and the second one is based on estimating the value of the vertex from its neighbors. The connectivity list is used to find the neighbors of the lost vertex. Figure 7 shows an example mesh (a) and the corresponding reconstructed meshes (b and c).

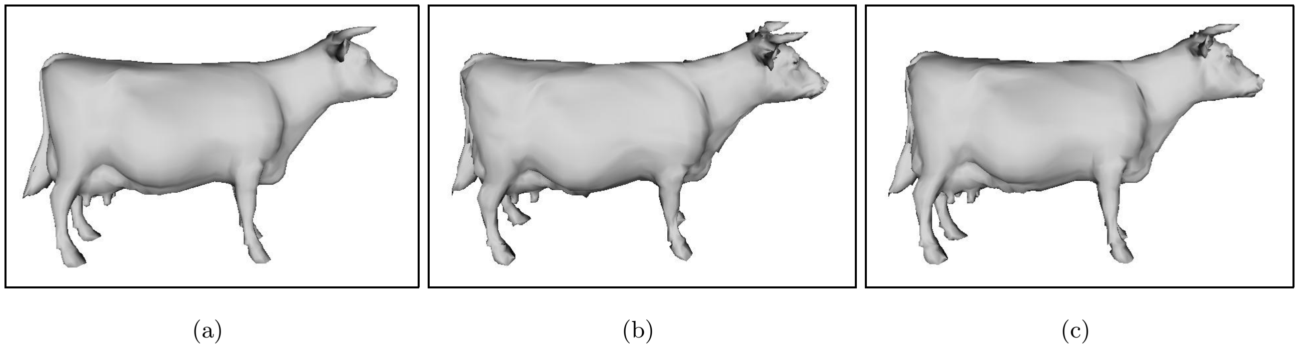


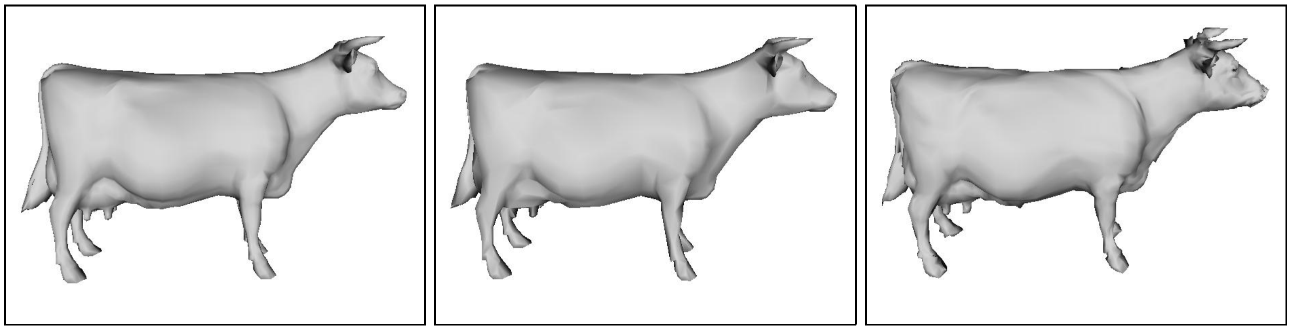
Figure 7. (a) original mesh (27KB); (b) Reconstructed mesh using 9.1KB bitstream; (c) reconstructed mesh using 10.4KB bitstream.

3. COMPARISON OF OUR APPROACH WITH OTHER APPROACHES

In single rate compression schemes, the mesh data is compressed before the transmission and then all the data is sent. In progressive compression schemes, the mesh data is compressed during the transmission and sent progressively. Therefore, first a low-resolution data is decoded and then the decoded model is updated to a higher resolution using new-coming data. In our approach, we use the beforehand compression property of the single rate compression and updatable decoded model property of the progressive compression.

The bitstream created by SPIHT coding has an hierarchical structure. Thus, after receiving the first l coefficients of the bitstream where l is smaller than the total length of the bitstream L, the decoder can reconstruct a lower resolution representation of the mesh.Once a mesh is formed using the first l coefficients then the decoder has the capability of updating the mesh using the remaining bitstream elements in a progressive manner. The mesh can be perfectly reconstructed when the whole bitstream is received. This property makes the proposed algorithm a member of the progressive class of mesh compression algorithms.

Most of the progressive mesh compression algorithms use edge collapses at the encoder and vertex splits at the decoder side. A typical low resolution version of a given mesh has a smaller number of vertices and larger triangles compared to the original mesh [7]. One cannot retrieve a lower resolution mesh structure from the bitstream in the proposed algorithm because the SPIHT encoder compresses the wavelet domain data by taking advantage of the multiscale correlation between scales. Because of this our compression algorithm is similar to single rate coding schemes. All the vertices can be recovered from a part of the bit-stream. The degradation in the quality comes from the distortion due to the inexact positioning of the vertices. These features of the proposed algorithm are demonstrated in Figure 8 for various resolution degradation levels.



(a) (b) (c)

Figure 8. (a) Original mesh; (b) simplified mesh; (c) mesh reconstructed by our algorithm using only 60 percent of the bitstream.

The difference between two resolutions in progressive mesh compression is represented by using a sequence of edge collapses or vertex splits [15]. In the proposed algorithm the difference between two resolutions is represented by using the parts of the bitstream that fine tune the positions of the mesh vertices.

The proposed algorithm is a geometry compression technique since it only compresses the vertex positions of the mesh, not the connectivity information. It can also be combined with a connectivity coder to become a complete mesh coder.

4. RESULTS

The proposed framework is tested by compressing the sandal and cow models (see Figures 9 and 10). The sandal and cow models are obtained from Viewpoint Datalabs Inc. and Matthias Müller (ETH Zürich), respectively. The original sandal model has a compressed data size of 31.7 KB and the cow model has a data size of 27.2 KB.

The filter banks that are used to compute the wavelet transform are given in the first column of Table 1. In the second column the percentage of the full length bitstream used for reconstruction is shown. The detail level parameters are given in column three. Size of the compressed data is shown in column four. The fifth and the sixth columns show the *"Hausdorff Distances"* between the original, quantized and reconstructed meshes. Meshes that are reconstructed from projection images on which no SPIHT encoding and decoding applied are called quantized meshes.

Filters	Bitstream Used (%)	Detail Level	Size (KB)	Org. - Quan. Hausdorff Dist.	Org. - Recons. Hausdorff Dist.
Lazy	40	3.0	6.39	0.022813	0.022830
Lazy	60	3.0	7.01	0.022813	0.0228224
Lazy	80	3.0	7.71	0.022813	0.022493
Lazy	60	4.5	7.84	0.021336	0.021347
Haar	40	3.0	8.51	0.022813	0.023225
Haar	60	3.0	10.2	0.022813	0.022827
Haar	80	3.0	12.1	0.022813	0.022825
Daubechies-10	60	3.0	13.0	0.022813	0.023813

Table 1. Compression results for the sandal model.

Filters	Bitstream Used (%)	Detail Level	Size (KB)	Org. - Quan. Hausdorff Dist.	Org. - Recons. Hausdorff Dist.
Lazy	40	3.0	9.51	0.0139930	0.014547
Lazy	60	3.0	10.4	0.0139930	0.014219
Haar	40	3.0	11.5	0.0139930	0.016085
Haar	60	3.0	13.8	0.0139930	0.014102
Haar	80	3.0	16.0	0.0139930	0.014011
Haar	100	3.0	18.0	0.0139930	0.014.28
Daubechies-4	40	3.0	12.0	0.0139930	0.018778
Daubechies-4	60	3.0	15.1	0.0139930	0.014661
Daubechies-4	80	3.0	18.2	0.0139930	0.014611
Daubechies-10	40	3.0	12.1	0.0139930	0.014194
Daubechies-10	60	3.0	15.2	0.0139930	0.045020
Biorthogonal-4.4	60	4.0	16.0	0.009446	0.014549
Biorthogonal-4.4	80	3.0	18.3	0.0139930	0.023555

Table 2. Compression results for the cow model.

5. CONCLUSION AND FUTURE WORK

In this paper, a new mesh compression framework using the SPIHT encoding is proposed. The 3D mesh data is first converted to two image-like 2D signals which are wavelet transformed. 2D signals can be defined on a rectangular grid or a quincunx grid. The wavelet domain data is compressed using a SPIHT encoder. Resultant bitstream is progressive in nature.

The SPIHT encoder provides better results when the values of signal samples are more correlated to each other. Hence, a mesh to image transform providing high correlation between the grid values will lead to higher compression ratios. The future work will include the use of a 3D grid representation of a given mesh and the use of 3D SPIHT for data compression.

An adaptive lifting structure [8] can be more advantageous than the lazy filterbank. One can use an adaptive predictor in the lifting structure. In sparse regions of the mesh image the predictor is useless as nonzero valued grid points are isolated. Therefore the predictor should not be used in such portions of the image. However, grid values are related to each other in detailed mesh regions (e.g, around the ears of the cow mesh the grid values are close to each other). In such regions a predictor will be useful, i.e., if the next pixel is non-zero then the predictor should be used otherwise it should not be used. The predictor scheme is shown in Figure 11. Results of this predictor with a switching structure will be presented during the paper presentation.

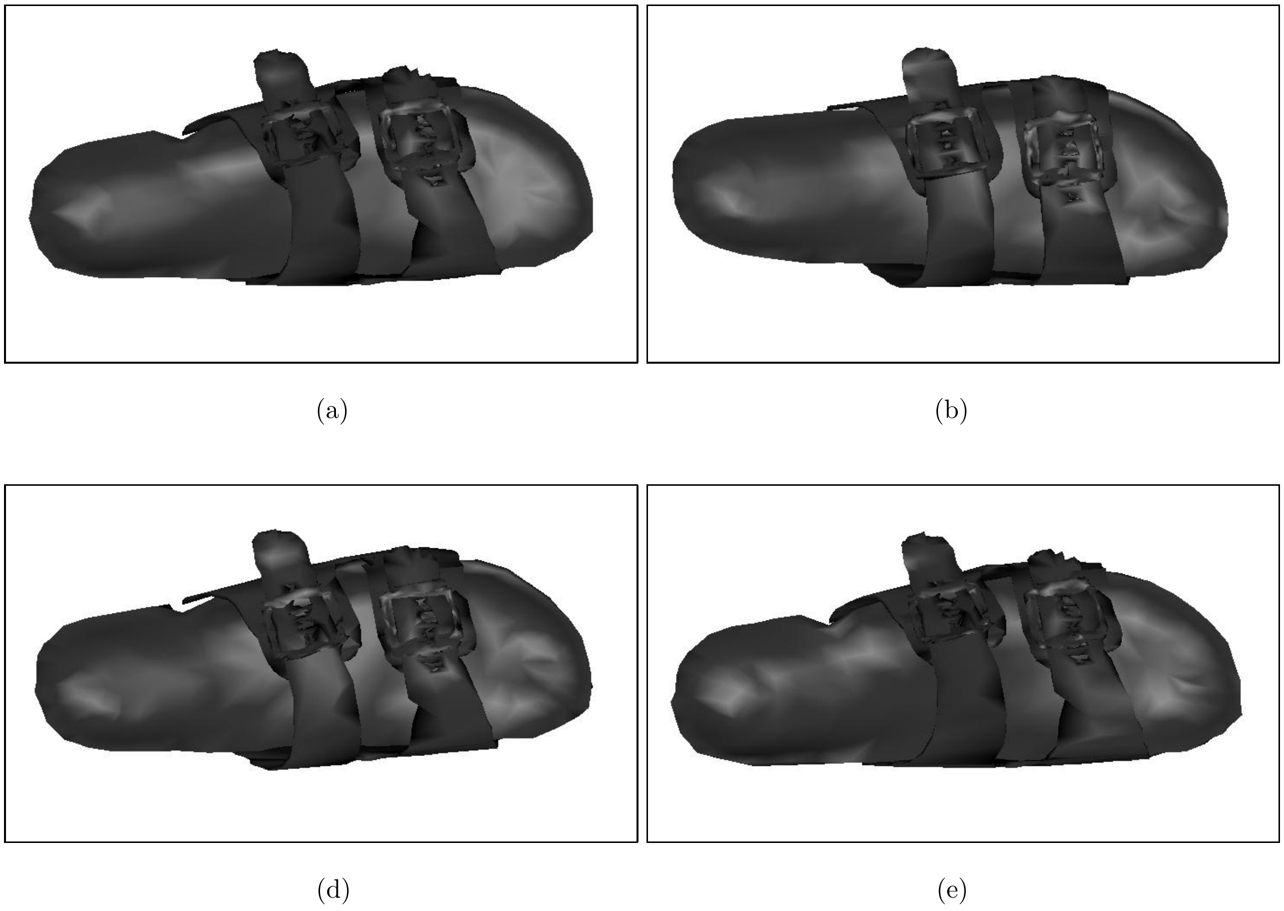
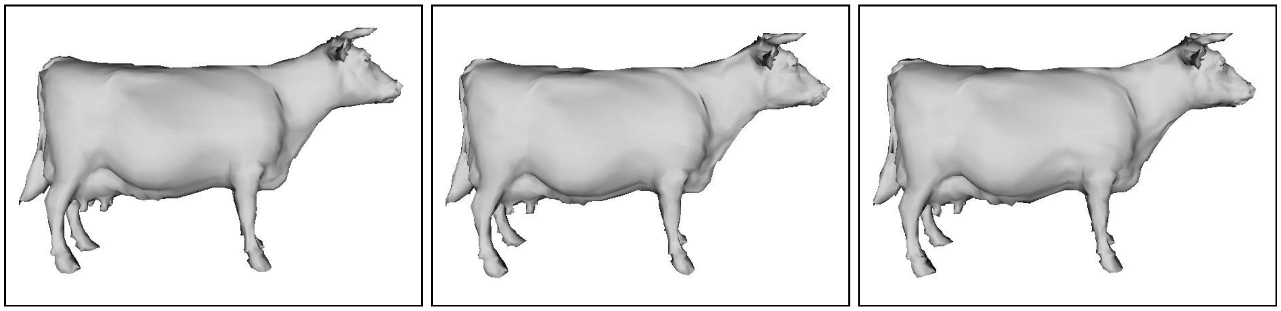


Figure 9. Reconstructed sandal meshes using parameters (a) lazy wavelet, 60% of bitstream, detail level=3; (b) lazy wavelet, 60% of bitstream, detail level=4.5; (c) Haar wavelet, 60% of bitstream, detail level=3; (d) Daubechies-10, 60% of bitstream, detail level=3.

APPENDIX A. NOTATION

$\mathbf{x_n}$: Grid Point
$\mathbf{p}$: The normalized actual coordinate of the mesh vertices
$\mathbf{n}$: is the plane that the 3D mesh is projected on
N: Number of the gridpoints on one edge of the projection plane $\mathbf{n}$
gridstep: The distance between to neighboring grid points
detaillevel: The measure of coarseness of the grid points. High detail level will lead more dense grid points and low detail level will lead coarse grid points
M: Number of the vertices in the mesh
$\mathbf{V}$: The 2D sampling matrix.
$\mathbf{p_p}$: Orthogonal projection of the position of the 3D mesh vertex on the selected plane
$\mathbf{m}$: Orthogonal vector to the plane $\mathbf{n}$


(a) (b) (c)

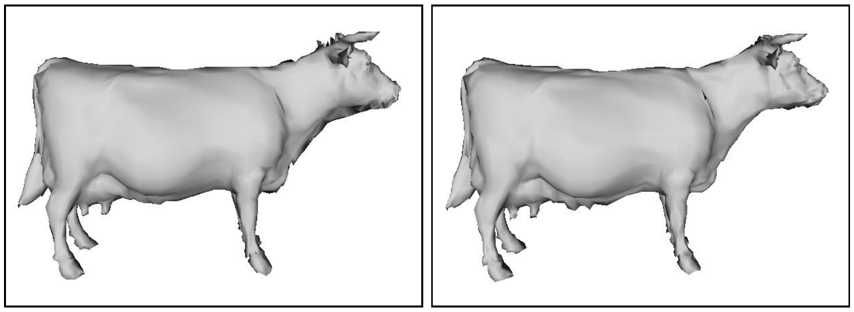

(d) (e)

Figure 10. Meshes reconstructed with a detail level of 3 and 0.6 of the bitstream. (a) Lazy, (b) Haar, (c) Daubechies-4, (d) Biorthogonal4.4, and (e) Daubechies-10 wavelet bases are used.

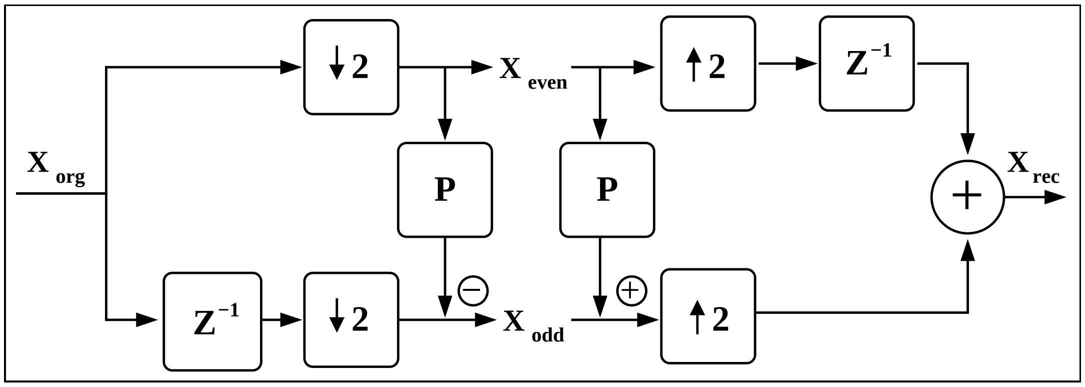


Figure 11. Lifting structure with an adaptive predictor.

REFERENCES

1. P. Alliez and C. Gotsman, “Recent advances in compression of 3D meshes”, In *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S Floater and M.A. Sabin (eds), Springer-Verlag, 2005.
2. R. Ansari and C. Guillemort, “Exact reconstruction filter banks using diamond FIR filters”, in *Proc. of Communication, Control and Signal Processing Conference*, 1990.
3. S. Arivazhagan, D. Gnanadurai, J.R. Antony Vance, K.M. Sarojini, and L. Ganesan. ”Evaluation of zero tree wavelet coders,”, *Proc. of International Conference on Information Technology: Computers and Communications*, p. 507, 2003.
4. E. Catmull and J. Clark, “Recursively generated B-spline surfaces on arbitrary topological meshes”, *Computer Aided Design*, Vol. 10, pp. 350-355, 1978.
5. P. Cignoni, C. Rocchini, and R. Scopigno, “Metro: measuring error on simplified surfaces”, *Computer Graphics Forum*, Vol. 17, No. 2, pp. 167-174, 1998.
6. J.-L. Gailly, gzip Compression Software.
7. M. Garland and P. Heckbert, “Surface simplification using quadric error metrics, *Proc. of ACM SIGGRAPH’97*, pp. 209-216, 1997.
8. Ö. N. Gerek and A. E. Çetin, “Adaptive Polyphase Subband Decomposition Structures for Image Compression,” *IEEE Trans. on Image Processing,* pp. 1649-1660, 2000.
9. C. Guillemot, A.E. Çetin and R. Ansari “M-channel nonrectangular wavelet representation for 2-D signals: basis for quincunx sampled signals” *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 4, pp. 2813-2816, 1991.
10. I. Guskov, W. Sweldens and P. Schröder, “Multiresolution Signal Processing for Meshes”, *Proc. of ACM SIGGRAPH*, pp. 325-334, 1999.
11. H. Hoppe, “Progressive meshes”, *Proc. of ACM SIGGRAPH*, pp. 99–108, 1996.
12. C. Loop, *Smooth subdivision surfaces based on triangles*, Masters Thesis, University of Utah, Department of Mathematics, 1987.
13. M. Lounsbery, T.D. DeRose, and J. Warren, “Multiresolution analysis for surfaces of arbitrary topological type”, *ACM Trans. on Graphics*, Vol. 16, No. 1, pp. 34-73, 1997.
14. F. Moran and N. Garcia, “Comparison of wavelet-based three-dimensional model coding techniques”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, No. 7, pp. 937-949, 2004.
15. R. Pajarola and J. Rossignac, “Compressed progressive meshes”, *IEEE Trans. on Visualization and Computer Graphics*, Vol. 6, No. 1, pp. 79-93, 2000.
16. J. Peng, C.-S. Kim, Kuo, and C.-C. Jay, *Technologies for 3D triangular mesh compression: A survey*, Technical Report, University of Southern California, 2003.
17. J. Rossignac. “Edgebreaker: connectivity compression for triangle meshes”, *IEEE Trans. on Visualization and Computer Graphics*, Vol. 5, No. 1, pp. 47-61, 1999.
18. J. Rossignac, “Surface simplification and 3D geometry compression”, Chapter 54 in *Handbook of Discrete and Computational Geometry*, 2004.
19. A. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees, *IEEE. Trans. Circ. Syst. Video Tech.*, Vol. 6, pp. 243–250, 1996.
20. J.M. Shapiro, “Embedded image coding using zerotrees of wavelets coefficients”,*IEEE Trans. Signal Processing*, Vol. 41, pp. 3445-3462, 1993.
21. C. Valens, EZW encoding, Available at http://perso.wanadoo.fr/polyvalens/clemens/ezw/ezw.html.
22. J. Ziv and A. Lempel. ”A universal algorithm for data compression.” *IEEE Trans. on Information Theory*, Vol. 23, No. 3, pp. 337-343, 1977.